Università di Pisa

Data Mining 2 Project

Ravdess Feature

THE RYERSON AUDIO-VISUAL DATABASE OF EMOTIONAL

SPEECH AND SONG (RAVDESS)

A cura di

Michele Dicandia 657494

A.A 2022/2023

Sommario

Sommario

1. Introduzione

The **Ravdess Feature** dataset is created by extracting basic statistics from the original audio data of the Ravdess dataset. After that, the following transformations are applied: differencing, zero-crossing rate, Mel-Frequency Cepstral Coefficients (MFCC), spectral centroid, and Short-Time Fourier Transform (STFT) chromagram. The features are extracted from 2452 WAV files. The features are also extracted by dividing each time series into four non-overlapping windows.

The variables (features) of the dataset include:

- Modality (audio-only)
- Vocal channel (speech, song)
- Emotion (neutral, calm, happy, sad, angry, fearful, disgust, surprised)
- Emotional intensity (normal, strong)
- Statement ("Kids are talking by the door", "Dogs are sitting by the door")
- Repetition (1st repetition, 2nd repetition)
- Actor (01 to 24)
- Sex (M, F)
- Filename (name of the corresponding audio file)
- Frame count (the number of frames from the audio sample)

Basic statistics are extracted for each audio waveform, including:

- Sum, mean, standard deviation (std), minimum, maximum
- Quantiles (q01, q05, q25, q50, q75, q95, q99)
- Kurtosis (kur) and skewness (skew)

Transformations are applied to each audio file, including:

- Lag1: Difference between each observation and the precedent (difference(t) = observation(t) - observation(t-1))
- Zero-crossing rate (zc)
- Mel-Frequency Cepstral Coefficients (mfcc)
- Spectral centroid (sc)
- Short-Time Fourier Transform chromagram (stft)

For each transformation, variables such as sum, mean, std, etc., are extracted.

The features are first extracted at the global level (i.e., for the entire signal), then divided into four equal-sized windows. The windows are indicated by the strings w1, w2, w3, or w4.

For example:

- "stft_skew_w4" indicates the skewness of the Short-Time Fourier Transform (STFT) chromagram of the fourth window of the audio signal.
- "stft_skew" indicates the skewness of the STFT chromagram of the entire audio signal.
- "skew_w1" indicates the skewness of the first window of the original audio signal.
- "skew" indicates the skewness of the entire original audio signal.

This notation provides a clear way to understand the meaning of the feature names and how they relate to different parts of the audio signal and its analysis.

2. Advanced Data Understanding

The Ravdess Feature dataset consists of two CSV files: "**RavdessAudioOnlyFeatures_TRAIN**" comprising 1828 records x 434 features, and "**RavdessAudioOnlyFeatures_TEST**" comprising 634 records and 434 features. The training set dataset includes all records whose actors belong to classes 01 to 18, while the test set dataset includes records of actors belonging to classes 19 to 24.

The datasets do not contain missing values.

The first step is to standardize all quantitative variables before proceeding with the analysis.

2.1 Dimensionality Reduction

Given the size of the dataset, a more focused analysis was conducted by selecting the most significant features. This was achieved through a **Recursive Feature Elimination** (RFE) approach. RFE systematically eliminates variables one by one based on their importance. Less significant variables are removed, resulting in a reduced set of features.

In the RFE process, a total of 9 features were selected. This selection was made by performing RFE for each number of features, ranging from 2 to 20. The Decision Tree classifier was employed, using the "emotion" variable as the target. The aim was to find the optimal number of features, denoted as "n," that maximizes accuracy and F1 score. In cases where accuracy and F1 score are equal, the smallest "n" is chosen.

The result of this process is the extraction of 9 features that contribute significantly to the dataset.

q25	kur	sc_q25	sc_q75_w2	sc_skew_w2	length_w4	skew_w4	mfcc_kur_w4	stft_q05_w4

These above are the 9 variables that provide a substantial marginal contribution and explain a significant portion of the dataset's variability. From now on, this will be the data matrix X used for conducting the analysis.

2.2 Outlier Detection

Regarding outlier detection, three different approaches have been discussed.

The first approach used to detect outliers is the graphical method, which is the most intuitive and straightforward. **Boxplots** were generated, as shown in Figure 1, and outliers were identified only in some of the 9 considered features.

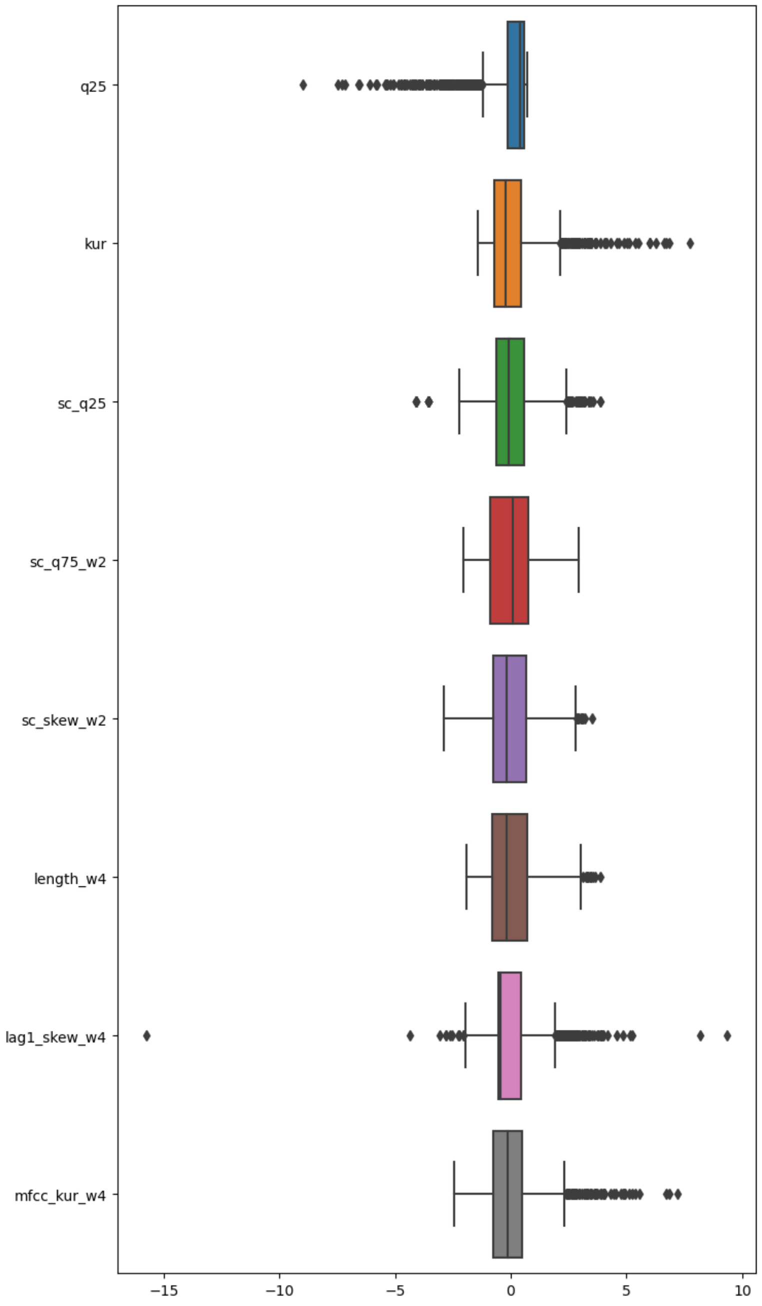


Figura 1-Outliers visual approch

Given that the data is standardized, the boxplots will all align around 0 on the x-axis, as the mean is 0. If they follow a normal distribution, the median will also be 0 because it coincides with the mean.

The second approach for outlier detection is based on density using the **Local Outlier Factor** (LOF). LOF detects outliers by measuring the deviation of a point from its local context based on the density of its neighbors, selecting a number of neighbors equal to 5.

The result of this outlier detection technique yields 104 outliers labeled as 1, while all other 1724 points are labeled as "inliers" with label 0. The output returned by Python is shown in the line below, and in Figure 2, the plot of the Local Outlier Factor outliers is depicted, with a line separating "inliers" (values on the left of the line) from "outliers" (values on the right).

```
(array([0, 1]), array([1724,  104], dtype=int64))
```

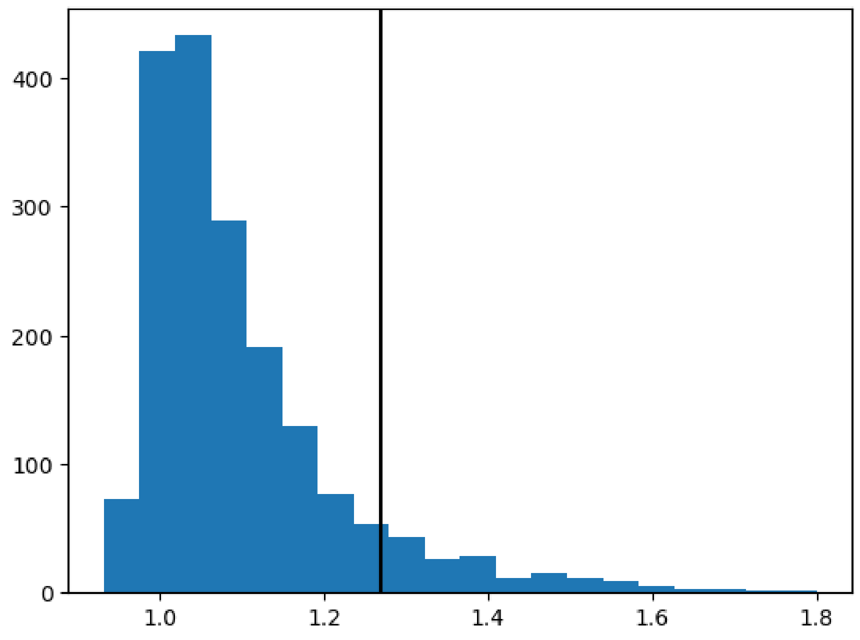


Figura 2- Local Outlier Factor

The third approach for outlier detection is based on the **Isolation Forest** model, which identifies outliers by exploiting the tendency of outliers to be more easily separated from normal points in the dataset. This method utilizes the Isolation Forest model that generates random decision trees.

Using this method, 231 outliers are identified and labeled as -1, in contrast to the inliers, which total 1597 and are labeled as 1. In the line below, we see the output result provided by Python for outlier detection using the Isolation Forest method. In Figure 3, the plot of the outliers is depicted, located to the left of the black line that separates the outliers from the inliers.

```
(array([-1,  1]), array([ 231, 1597], dtype=int64))
```

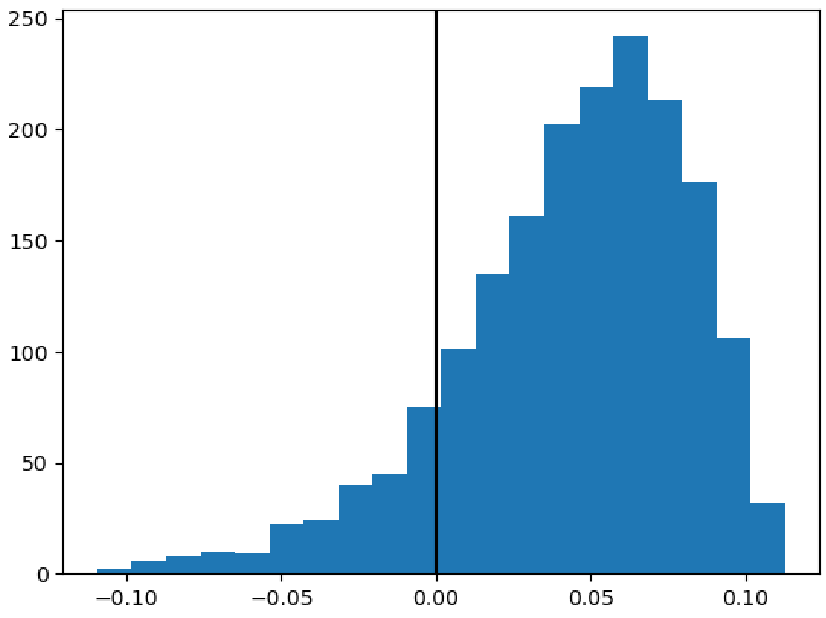


Figura 3- Model based aaproach: Isolation Forest

In Figure 4, the outlier detection is illustrated through a scatterplot using the first two principal components of the data matrix X. Observations are represented in blue, while outliers are represented in red.

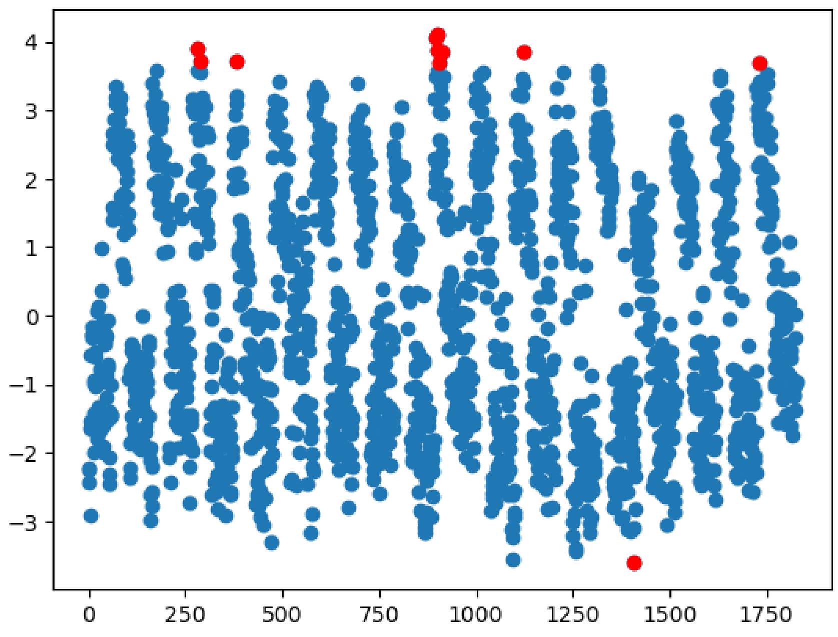


Figura 4-scatterplot outliers

Regarding the replacement of outliers, the approach was to identify the top 1% of outliers and replace them with the mean. A Shapiro-Wilk normality test was performed on the data matrix X containing the 9 quantitative variables. The resulting p-values were all found to be insignificant, indicating that they all come from a normal distribution. Subsequently, a function was created to replace values that are extreme and represent the top 1% with the mean, which is 0 in this case (since the data is standardized).

Values that are greater than 2.57 in absolute value (mean + or - 2.57 * std) are considered outliers. The value 2.57 corresponds to a point on the normal distribution where values higher in absolute value have a 1% probability of being observed. This approach ensures that the extreme values are replaced with a more representative value while considering the standard deviation and distribution characteristics.

Nelle figure 5 e 6 si mostrano i boxplot dei dati prima che gli outlier fossero rimossi e dopo:

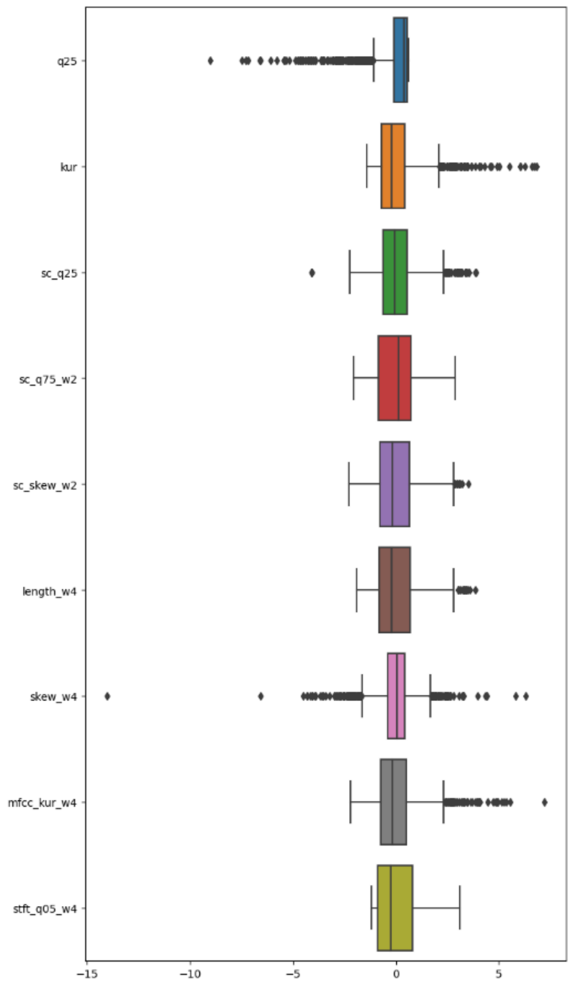


Figura 5-boxplot prima dell'eliminazione di outliers

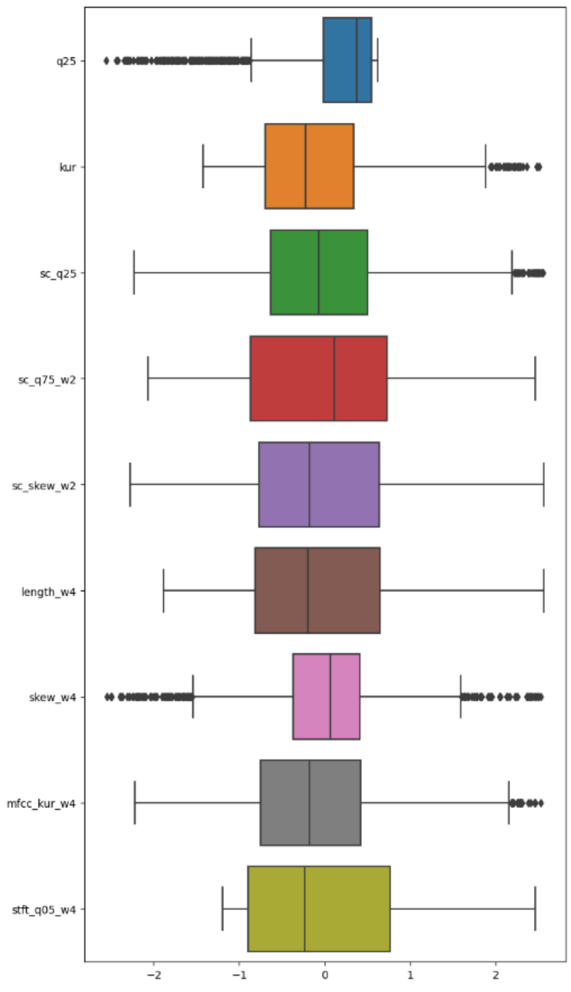


Figura 6-boxplot dopo l'eliminazione di outliers

2.3 Imbalance Learning

Regarding the issue of imbalanced learning, a decision was made to address the imbalance in the "vocal channel" variable. The goal was to achieve a proportion between the two modes of the variable, shifting from (59% - 41%) to (96% - 4%). Two different undersampling methods were employed: Random Undersampling and Condensed Nearest Neighbor, along with two oversampling methods, namely SMOTE and ADASYN.

The "vocal channel" variable has two modes: speech and song, with original absolute frequencies of exactly 1080 and 748, respectively. The scatterplot of the first two principal components of the data matrix X, segregating values of "speech" and "song," is depicted as follows (Figure 7):

(Note: As a text-based AI, I can't directly display images or plots. If you have specific questions about the scatterplot or need further assistance, please feel free to ask.)

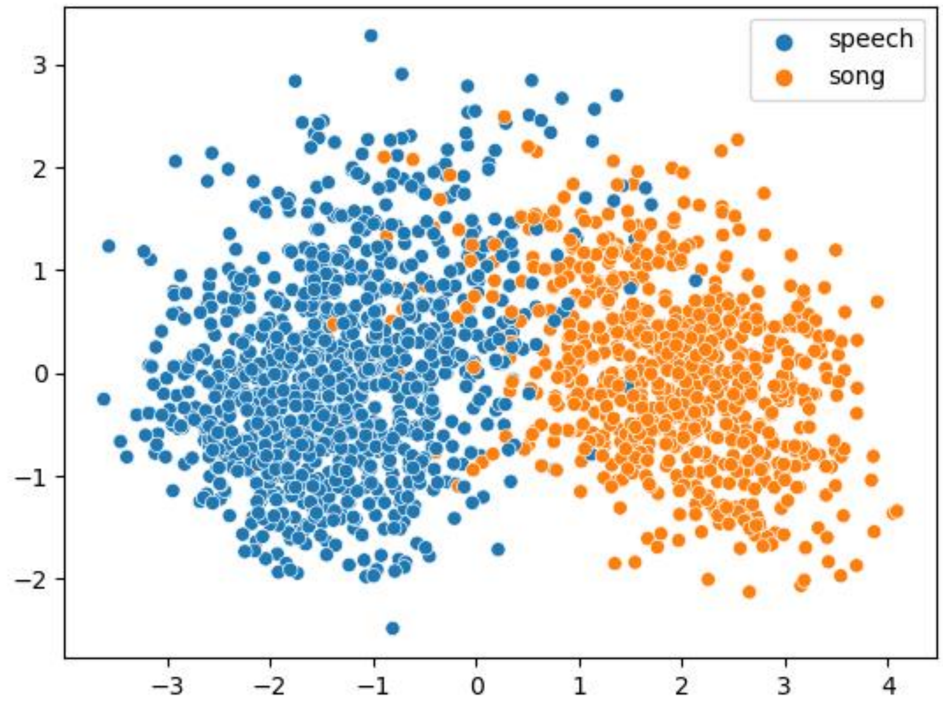


Figura 6.1-scatterplot originale

With the following decision tree classifier, which yields these output results:

```
              precision    recall  f1-score   support

        song       0.92      0.94      0.93       264
      speech       0.95      0.94      0.95       360

    accuracy                           0.94       624
   macro avg       0.94      0.94      0.94       624
weighted avg       0.94      0.94      0.94       624
```

So, the size of the 'speech' mode has been reduced from 1080 to 30, resulting in this scatterplot (Fig. 6.2)

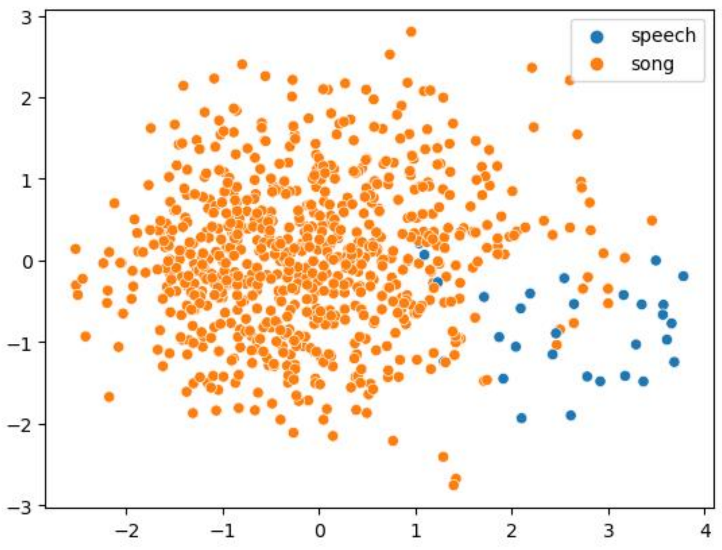


Figura 6.2-scatterplot classe sbilanciata

So, undersampling techniques were adopted. The first one is **Random Undersampling**, which reduces the number of instances in the 'song' mode, achieving a balance between the two classes. The decision tree classifier was trained using the dataset obtained after applying the Random Undersampling technique. The goal is to assess how well the model trained on a balanced dataset can accurately predict the test classes.

In Figures 6.3 and 6.4, the scatterplot and the classification report with the ROC curve are depicted. It can be observed that the achieved results for the minority class in terms of precision and recall are quite high, indicating that this Undersampling model is successful in correctly predicting the test classes. The precision of the 'speech' class is 0.92, the recall is 0.86, and the F1 score is 0.88. The AUC value is 0.89

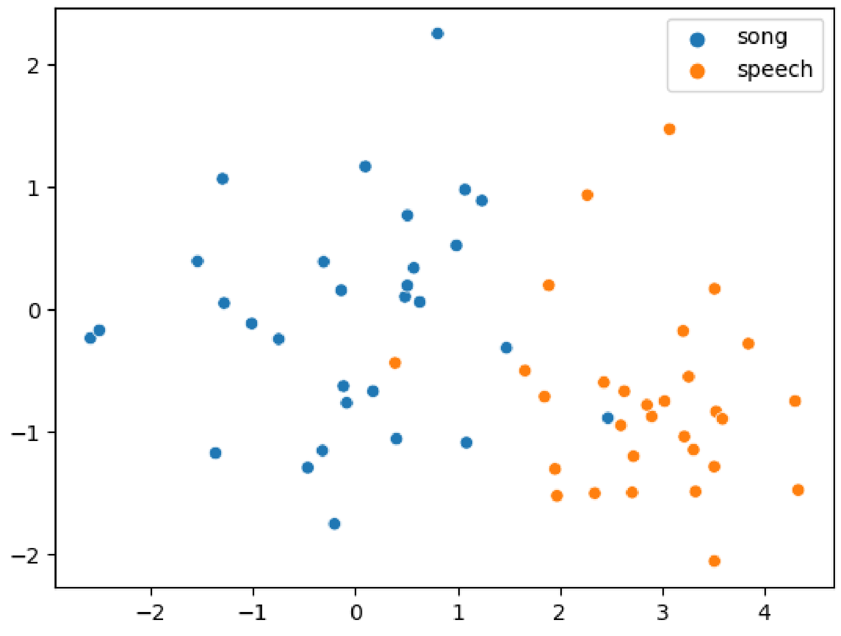


Figura 6.3-scatterplot Random Undersampling

```
Accuracy 0.8901515151515151
F1-score [0.89377289 0.88627451]
              precision    recall  f1-score   support

        song       0.87      0.92      0.89       264
      speech       0.92      0.86      0.89       264

    accuracy                           0.89       528
   macro avg       0.89      0.89      0.89       528
weighted avg       0.89      0.89      0.89       528
```

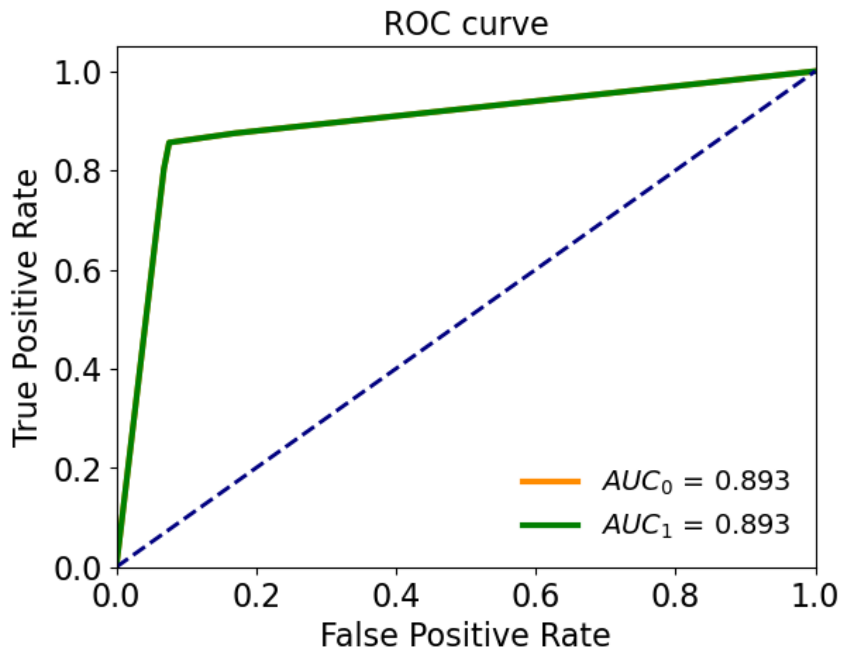


Figura 6.4-roc curve Random Undersampling

The second Undersampling method is **Condensed Nearest Neighbor** (Figure 6.5), which reduces the number of instances in the 'song' mode to balance the dataset. Similar to the first undersampling method, a decision tree model is then trained, and the scatterplot and classification report with the ROC curve are depicted below. In this case, the precision, recall, and F1 score values for the minority class ('speech') are slightly lower compared to the first undersampling method. The precision of the 'speech' class is 0.57, the recall is 0.48, and the F1 score is 0.52. The AUC value (Figure 6.6) is 0.74, slightly lower than the AUC of the random undersampling method. This outcome is predictable since the precision, recall, and F1 score values for 'speech' are all lower than the values obtained with random undersampling.

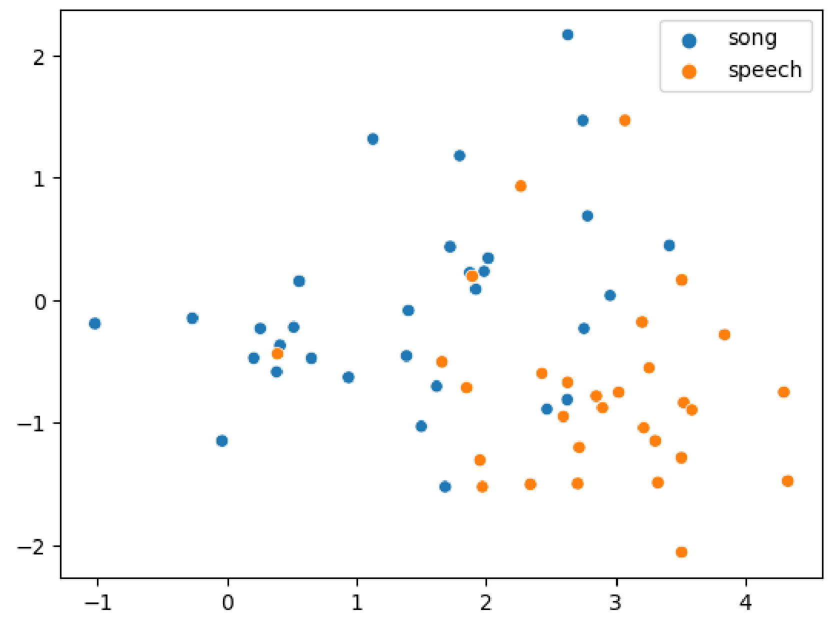


Figura 6.5-scatterplot condensed nearest neighbor

```
Accuracy 0.8544303797468354
F1-score [0.9141791  0.52083333]
              precision    recall  f1-score   support

        song       0.90      0.93      0.91       264
      speech       0.57      0.48      0.52        52

    accuracy                           0.85       316
   macro avg       0.73      0.70      0.72       316
weighted avg       0.85      0.85      0.85       316
```

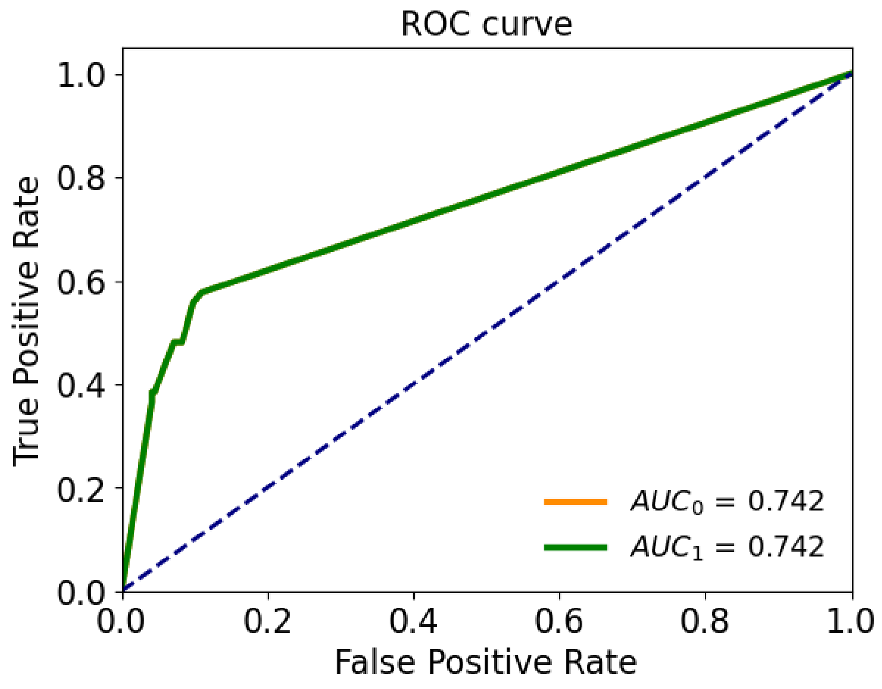


Figura 6.6-roc curve condensed nearest neighbor

As for oversampling, two important techniques, SMOTE and ADASYN, were employed. The objective of oversampling is to increase the size of the minority class, aiming to balance the two classes.

The first method, **SMOTE**, generates new instances in the minority class that lie between the selected instance and its neighbors. Here, the corresponding scatterplot (Figure 6.7), classification report, and ROC curve (Figure 6.8) are depicted. The precision value for the 'speech' class is 0.71, the recall is 0.57, and the F1 score is 0.57. The AUC value is 0.73

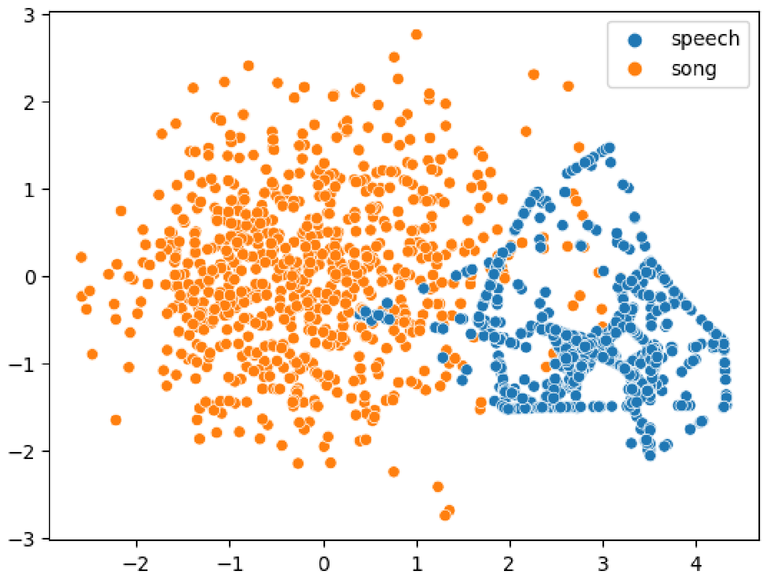


Figura 6.7-scatterplot smote

```
Accuracy 0.8829113924050633
F1-score [0.93211009 0.57471264]
              precision    recall  f1-score   support

        song       0.90      0.96      0.93       264
      speech       0.71      0.48      0.57        52

    accuracy                           0.88       316
   macro avg       0.81      0.72      0.75       316
weighted avg       0.87      0.88      0.87       316
```

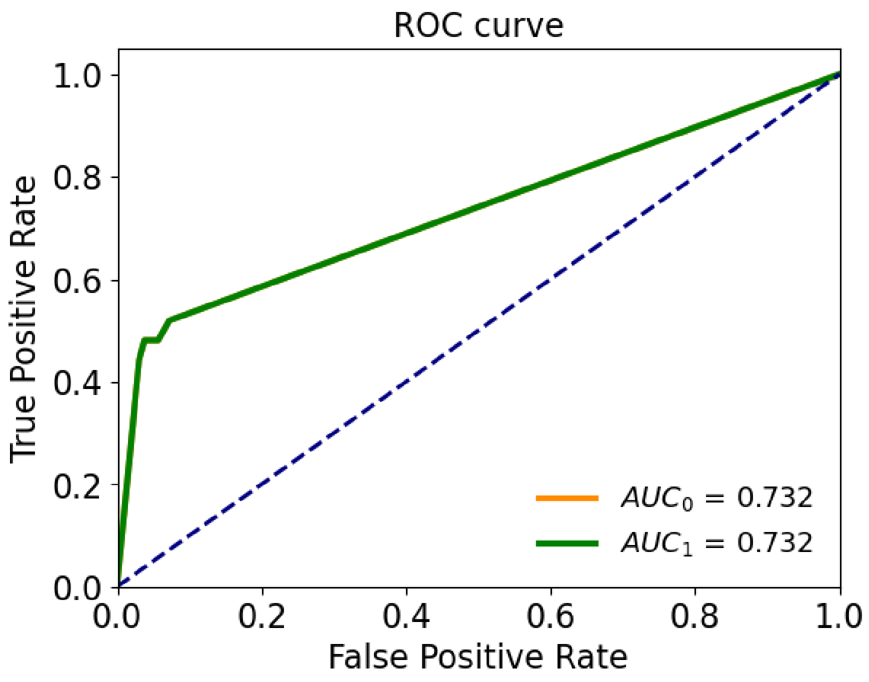


Figura 6.8-roc curve smote

The second oversampling method is **ADASYN,** which generates new instances in the minority class adaptively based on point density. As usual, the scatterplot (Figure 6.9), classification report, and ROC curve (Figure 6.10) are shown.

The precision value for the minority class is 0.6, the recall is 0.52, and the F1 score is 0.56. The AUC value is 0.75.

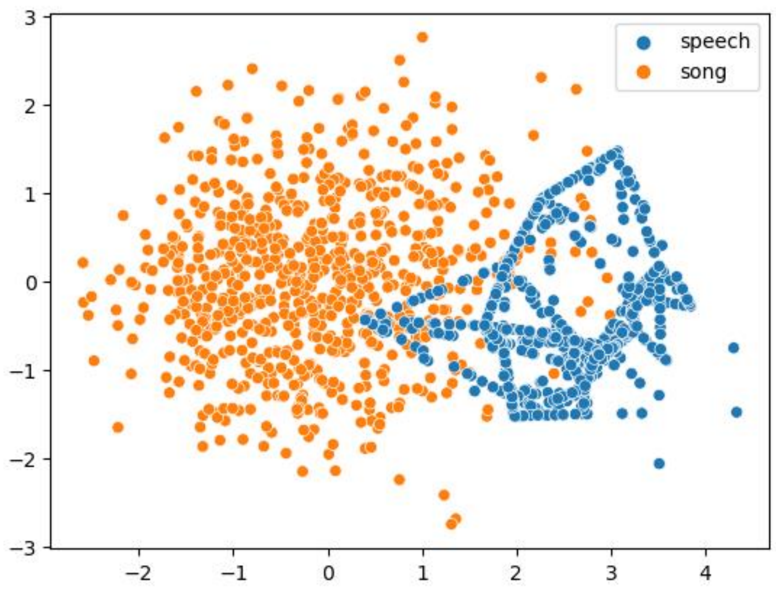


Figura 6.9-scatterplot adasyn

```
Accuracy 0.8639240506329114
F1-score [0.91962617 0.55670103]
               precision    recall  f1-score   support
```

```
        song       0.91      0.93      0.92       264
      speech       0.60      0.52      0.56        52

    accuracy                           0.86       316
   macro avg       0.75      0.73      0.74       316
weighted avg       0.86      0.86      0.86       316
```

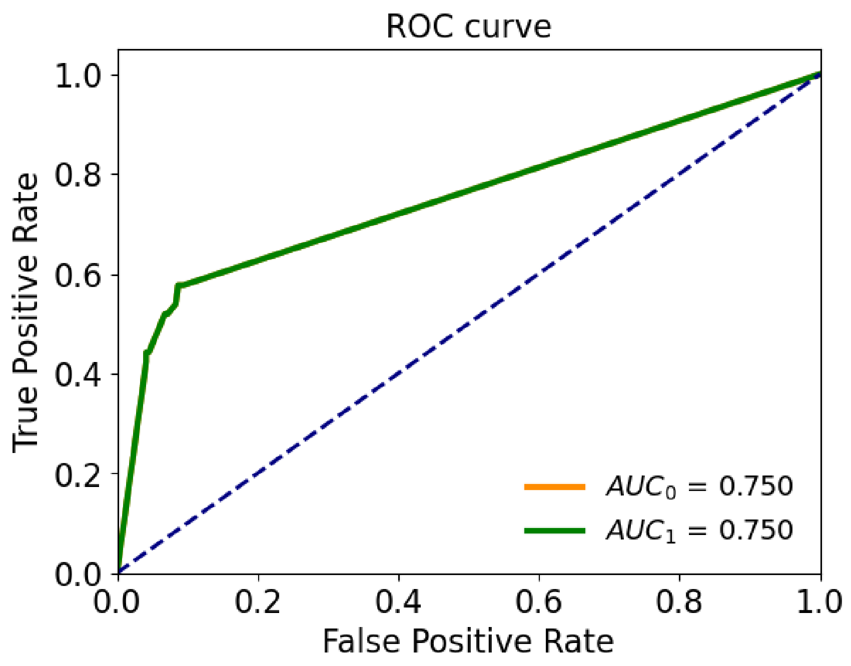


Figura 6.10-roc curve adasyn

3. Advanced Classification

In this section, the classification task will be implemented using the prepared dataset as presented in Chapter 1. For the purpose of classification, both basic and advanced algorithms will be utilized. The goal is to construct models that, based on the values assumed by various features, can predict the class of the target variable 'vocal_channel' as either 'speech' (1) or 'song' (0).

The classification methods to be employed include Logistic Regression, Support Vector Machine, Neural Networks, Ensemble Methods, and Gradient Boosting.

3.1 Logistic Regression

Logistic regression is a specific extension of generalized linear regression models that appropriately handles non-continuous dependent variables, particularly binary variables. In the context of the current analysis, where the target variable 'vocal_channel' is binary by nature, logistic regression allows us to perform the classification task appropriately. The data matrix X serves as our explanatory variable. With the LogisticRegressor model, the coefficient beta is estimated as 2.245, representing the effect of independent variables on the probability of the event of interest, while alpha is estimated as -0.007, which is the intercept or the output value when all independent variables are zero.

Here is the classification report: In summary, there is an accuracy of 83%, with an F1 score of 0.78 for the 'song' class and 0.87 for the 'speech' class.

```
Accuracy 0.8381410256410257
F1-score [0.78372591 0.87067862]
              precision    recall  f1-score   support

        song       0.90      0.69      0.78       264
      speech       0.81      0.94      0.87       360

    accuracy                           0.84       624
   macro avg       0.85      0.82      0.83       624
weighted avg       0.85      0.84      0.83       624
```

In figura 7 viene mostrato il plot della regressione logistica:

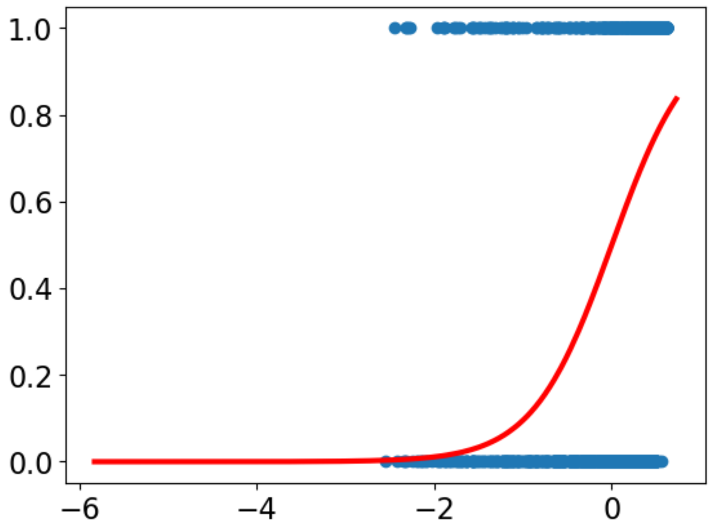


Figura 7-regressione logistica

3.2 Support Vector Machine

The **Support Vector Machine** (SVM) is a supervised learning model associated with learning algorithms for regression and classification tasks. In this analysis, the linear SVM has been implemented, which is a classifier based on the idea of finding a hyperplane that best separates a dataset into two classes. For this analysis, the **'LinearSVC'** function from the sklearn library has been utilized.

After running the algorithm multiple times with different combinations of parameters such as C (regularization parameter), penalty, and loss on the training set, the results obtained from the most performant configuration were chosen. This configuration was then applied to the test set, using the following parameters: **LinearSVC(C=1, penalty='l2', loss='squared_hinge')**. The accuracy achieved is 0.923, with an F1 score of 0.911 for the 'song' class and 0.932 for the 'speech' class. In general, we can conclude that this model accurately predicts the 'speech' and 'song' classes.

```
Accuracy 0.9230769230769231
F1-score [0.91111111 0.93220339]
                precision    recall  f1-score   support

song               0.89      0.93      0.91       264
speech             0.95      0.92      0.93       360

accuracy                               0.92       624
macro avg          0.92      0.92      0.92       624
weighted avg       0.92      0.92      0.92       624
```

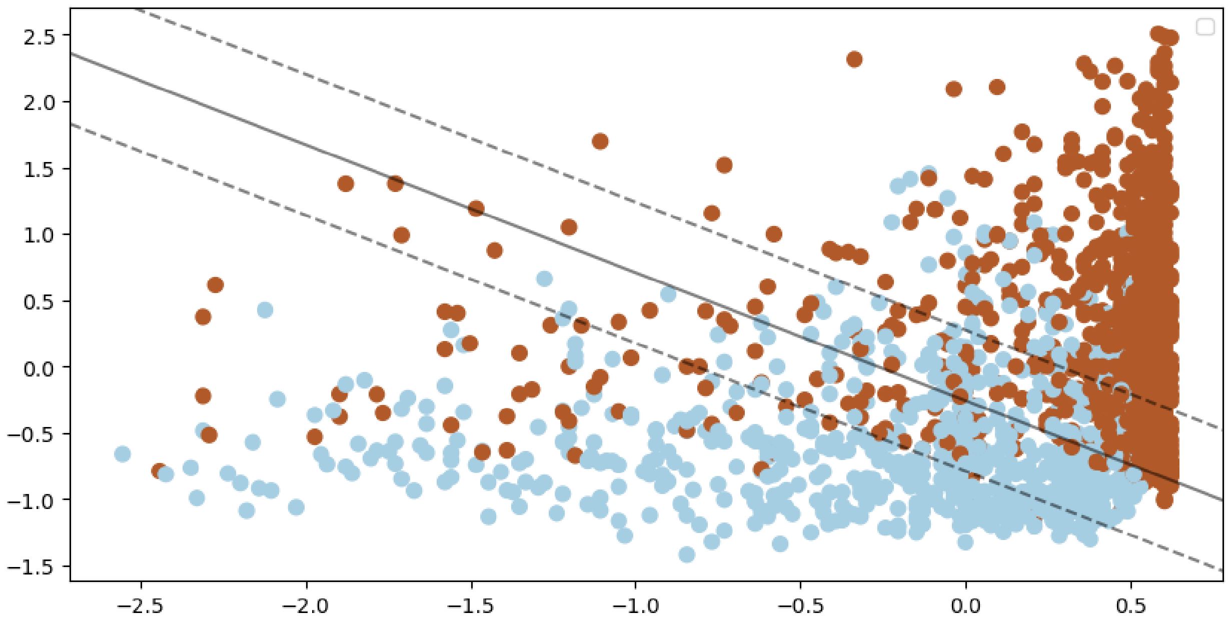
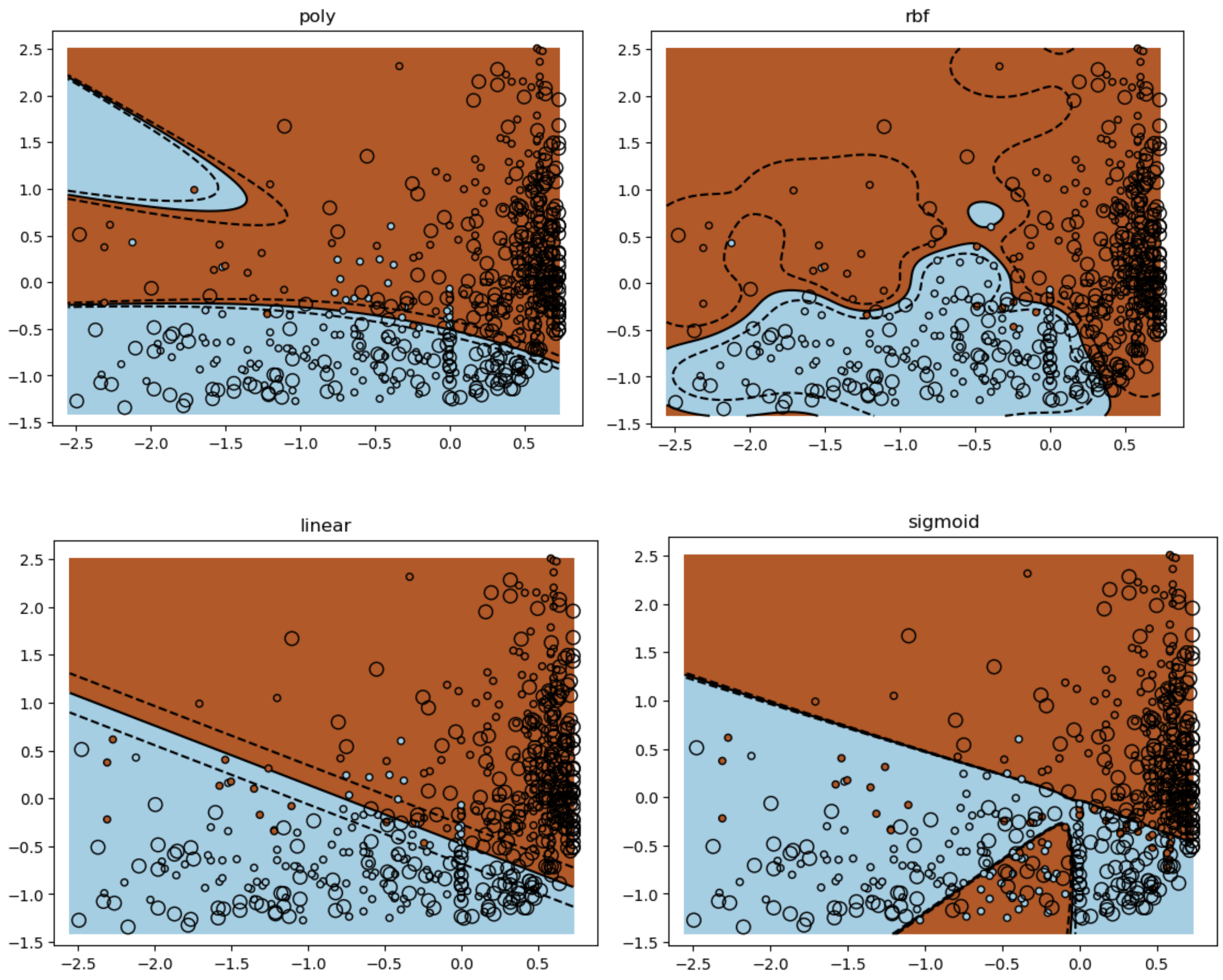


Figura 8-linear svm

In the case of non-linear classification using Support Vector Machine, kernel methods are employed to shape higher-dimensional non-linear models by implicitly mapping their inputs into a multidimensional feature space. Similarly, in this scenario, the algorithm has been tested with various parameters, particularly by varying different kernel methods (polynomial, RBF, linear, sigmoid).

The kernel that yielded the best results is the **'rbf'** (Radial Basis Function) kernel, achieving an accuracy of 0.94.



3.3 Neural Networks

For neural network modeling, the chosen model is the **Multilayer Perceptron Classifier** (MPC), implemented with various optimization algorithms (solvers) such as 'sgd' and 'adam'. This choice ensures the fair determination of relative connection weights and minimizes the level of loss. Different types of learning rates ('constant', 'invscaling', 'adaptive') were implemented, along with momentum values (0 and 0.9).

Multiple Multilayer Perceptron Classifier models were trained using various combinations of solvers, learning rates, and momentum. Training scores and loss functions were plotted, and the parameters with the highest training score and lowest loss were selected. The chosen parameters are solver='sgd', learning_rate='constant', and momentum=0.9, with training set score and loss of 0.998 and 0.016957, respectively.

Therefore, the trained neural network model **is MLPClassifier(random_state=0, solver='sgd', learning_rate='constant', momentum=0.9),** yielding the following classification report: The model's accuracy is 0.944, with an F1 score of 0.934 for the 'song' class (0) and 0.950 for the 'speech' class (1).

```
Accuracy 0.9439102564102564
F1-score [0.93482309 0.95077356]
```

```
              precision    recall  f1-score   support

           0       0.92      0.95      0.93       264
           1       0.96      0.94      0.95       360

    accuracy                           0.94       624
   macro avg       0.94      0.94      0.94       624
weighted avg       0.94      0.94      0.94       624
```

Using the MLPClassifier with hidden layers, models were trained with four different activation functions: identity, logistic, relu, and tanh. The model with the 'logistic' activation function exhibits an accuracy of 57%, with an F1 score of 0 for the 'song' class and 0.73 for the 'speech' class, making it a less favorable model. The models with the 'identity' and 'relu' activation functions achieve an accuracy of 94%, with F1-scores of [0.93656716, 0.95224719] for the 'identity' activation function model and [0.93632959, 0.95238095] for the 'relu' activation function model. In these F1 score results, the first value corresponds to the 'song' class (0), while the second value corresponds to the 'speech' class (1).

The model with the **'tanh'** activation function shows slightly higher accuracy and F1 scores, with an accuracy of 0.9487179487179487 and F1-scores of [0.94007491, 0.95518207]. Therefore, among all the models, the one with the 'tanh' activation function is preferred.

When performing hyperparameter tuning, the best combination of parameters is as follows:

```
        {'optimizer': 'adam', 'n_layers': 2, 'h_dim': 7, 'activation': 'relu'}
```

3.4 Ensemble Methods

Continuing with the classification task, we explored some ensemble methods, which combine predictions from multiple machine learning algorithms to make more accurate predictions compared to any single model. The end result is a 'meta-classifier' that exhibits better generalization performance compared to individual classifiers.

As a first model, we experimented with the random forest classifier, which combines a set of decision trees to enhance the overall system's performance. The final prediction of the **Random Forest Classifier** is determined through a combination of predictions from all trees in the Random Forest. The random forest also calculates feature importance, resulting in a bar chart of variable importance (Figure 9). The variable with the highest importance is 'length_w4', followed by 'sc_q75_w2' and 'sc_skew_w2'.

Next to it, permutation importance (Figure 10) of the variables is illustrated. This measure aims to quantify how much the model's performance degrades when observations of a particular variable are randomly permuted."

(Note: The figures mentioned, such as Figure 9 and Figure 10, are not provided in the text you provided. Please refer to the actual figures for a complete understanding of the content.)

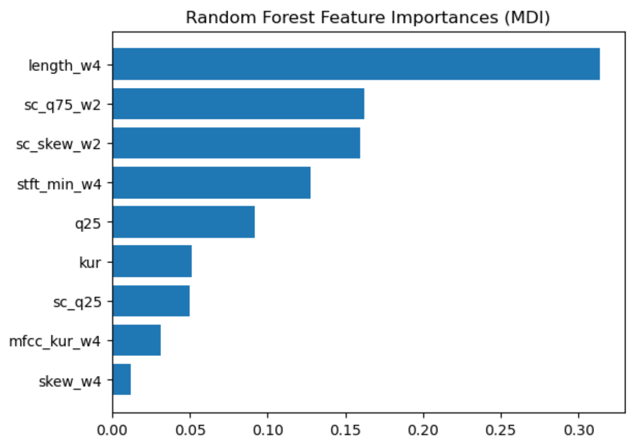


Figura 9

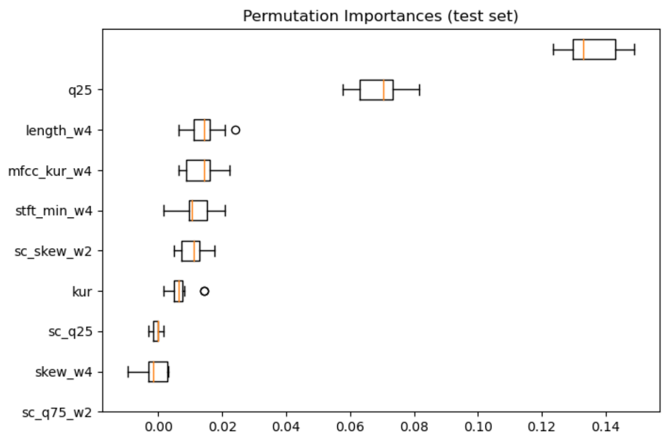
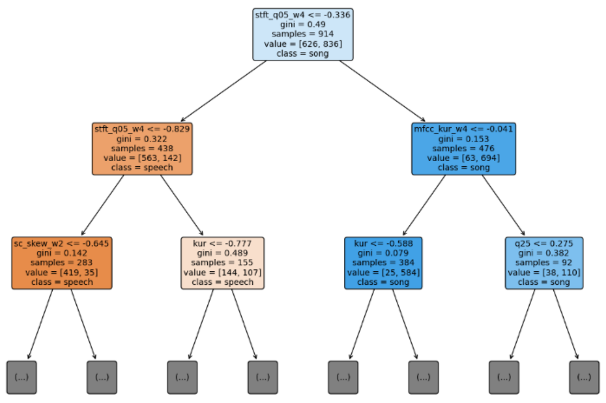


Figura 10

As for the decision trees, the first two decision trees trained by the random forest are depicted (in Fig 11 and 12) with a number of estimators equal to 100, separation criterion of Gini index, min_samples_split=2, and min_samples_leaf=1.



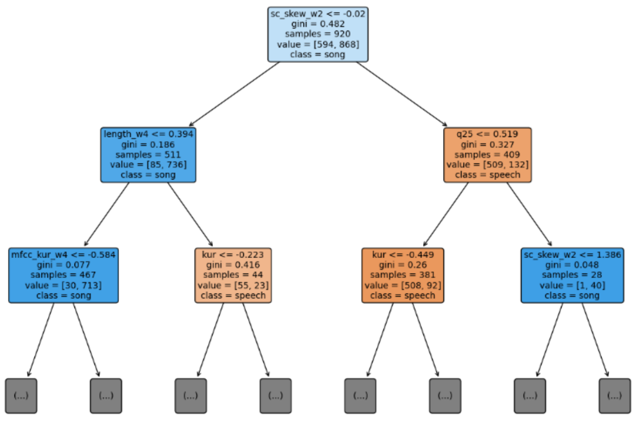


Figura 11

Figura 12

Having a classification report as follows:

```
Accuracy 0.9567307692307693
F1-score [0.9489603  0.96244784]
             precision    recall  f1-score   support

           0       0.95      0.95      0.95       264
           1       0.96      0.96      0.96       360

    accuracy                           0.96       624
   macro avg       0.96      0.96      0.96       624
weighted avg       0.96      0.96      0.96       624
```

With an accuracy of 0.956, the f1 score for class 0 (song) is 0.948, while the f1 score for class 1 (speech) is 0.962.

Regarding hyperparameter tuning, a GridSearch was conducted with parameters: max_depth=(none, 2, 3,...,20), min_samples_split=[2, 5, 10, 20, 30, 50, 100], and min_samples_leaf=[1, 5, 10, 20, 30, 50, 100]. The best parameter combination obtained was: **{'min_samples_split': 50, 'min_samples_leaf': 5, 'max_depth': 20}.**

Moving on to the bagging model, which aims to train multiple independent models on random subsets of the training set and combine their predictions to obtain a more accurate final prediction while reducing variance. Using various base classifiers and setting the number of estimators to 100, the bagging model with the base estimator **RandomForestClassifier** achieved the highest accuracy and f1 score. The accuracy is 0.961, with an f1 score of 0.955 for class song and 0.966 for class speech.

For boosting, the AdaBoostClassifier was used, and after training multiple models with different base estimators, the best-performing model was the one with the base estimator **RandomForestClassifier**.

3.5 Gradient Boosting

The tried models include Gradient Boosting Classifier, HistGradientBoostingClassifier, XGBClassifier, and LGBMClassifier, with the target variable always being 'vocal_channel' and the independent variables being the matrix of X obtained from RFE. All of them achieve an accuracy above 90%. Among these, however, the model that performs the best is the **HistGradientBoostingClassifier** with an accuracy of 0.956 and f1 scores for song and speech (0.945, 0.959), which are slightly higher compared to the evaluation metrics of the other models.

3.6 Advanced Regression

As for the regression problems, two models were chosen for representation: the Gradient Boosting Regressor and the Random Forest Regressor. Unlike classification problems, regression problems have different evaluation metrics, one of which is the Mean Squared Error (MSE). Other important metrics to evaluate are the training score and validation score, which assess performance during the training and validation process. The training score indicates how well the model fits the training data.

In the **Gradient Boosting Regressor**, the MSE is 0.04 with the validation score slightly higher than the training score (Figure 13). As the sample size increases, it appears that the training score and validation score become closer to each other.

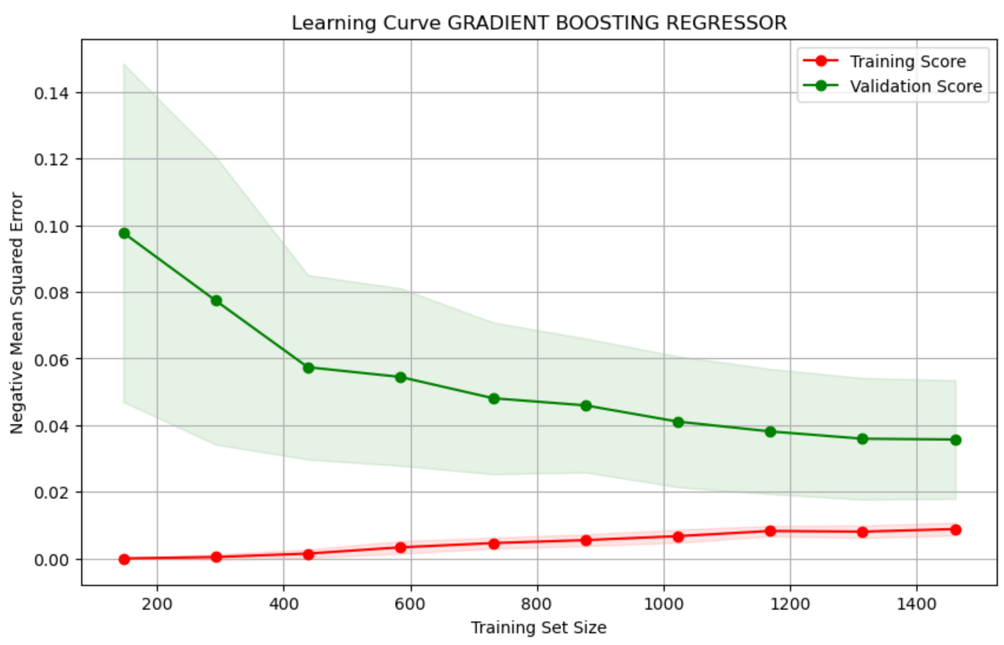


Figura 13

The second regression model is the **Random Forest Regressor** with the number of estimators set to 500. Here, the MSE is 0.041 with the MSE of the training score almost always at 0, while the MSE of the validation score appears to decrease as the sample size grows. The gap between the training score and the validation score is larger compared to the **Gradient Boosting Regressor**. It's likely that we are dealing with a situation of underfitting here. Between the two regression models, the better choice is indeed the Gradient Boosting Regression model, as shown in the learning curve, where the validation score and training score are less distant from each other compared to the Random Forest Regressor.

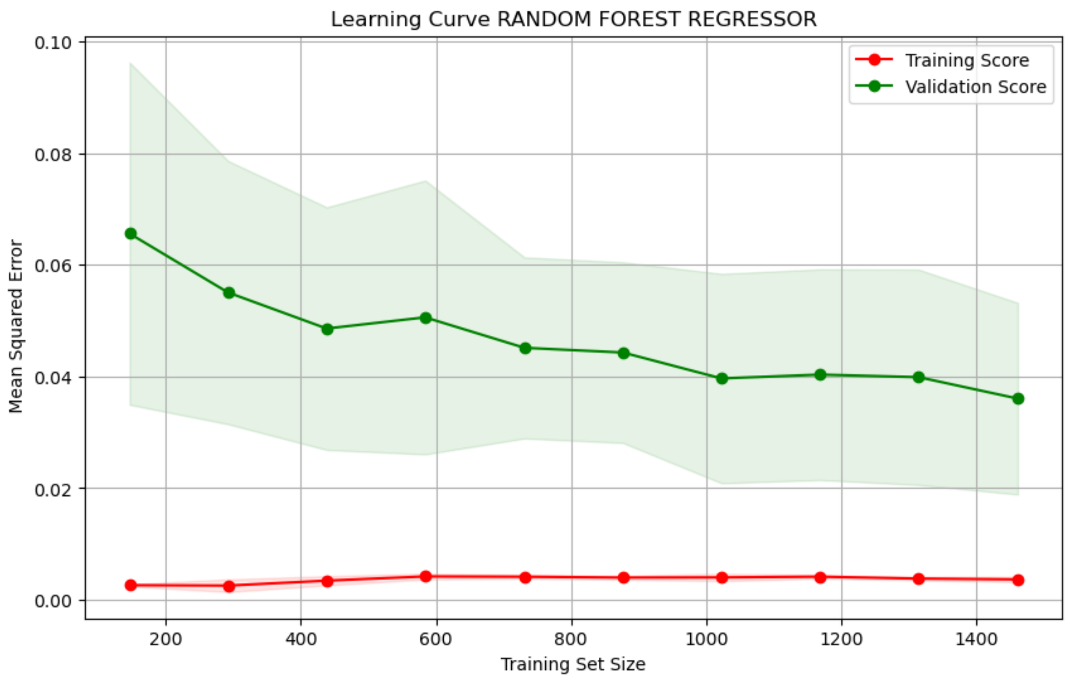


Figura 14

4 Time Series Analysis

In this section, the analysis of time series is conducted by considering 2452 audio samples of actors in WAV format, where they repetitively utter the phrases 'Dogs are sitting by the door' or 'Kids are talking by the door.' These audio samples are then transformed into time series data.

4.1 Data Understanding e Preparation

The extracted time series have a length of approximately 160,000 units, but the sound begins around observation 35,000 and ends around observation 130,000. Therefore, we extract all 2452 time series from unit 20,000 to unit 140,000 in order to have a consistent length, decimating it with a factor of q=24, resulting in 5,000 units per time series. For these 5,000-unit time series, a discrete Fourier approximation is applied with n units set to 500. This is done to reduce the length of the series while retaining its characteristics, allowing for clustering and classification analyses with lower processing time and memory usage compared to analyzing the series without approximation, even though some accuracy is sacrificed. After obtaining the matrix containing the 2452 time series approximated with a length of 500 units, offset transformations are performed. These transformations involve subtracting the mean of the series from all temporal observations, amplitude scaling to ensure a consistent amplitude of the data, detrending to remove the trend from the series, and smoothing to eliminate the series' seasonality and random oscillations (white noise).

4.2 Motif and Anomalies

Motifs are sequences that repeat within a time series, which are significant subsequences within a time series. In this example, the following record was utilized: {vocal channel = speech; emotion = neutral; emotional_intensity= normal; statement = kids are talking by the door; repetition = 1st; actor= 1} The matrix profile was calculated using the matrixprofile-ts package in Python, and subsequently, the top 3 motifs of the time series were computed.

Anomalies, on the other hand, are distinct anomalous patterns or subsequences that significantly differ from the rest of the time series. The top 3 anomalies were then identified in the series after deriving the matrix profile.

Figures 15, 16, and 17 represent the matrix profile of the time series, the plot with the top 3 motifs, and the top 3 anomalies, respectively. The top 3 motifs are highlighted in red (top 1 motif), green (top 2 motif), and blue (top 3 motif), while the top 3 anomalies are marked in black (top 1 anomaly), yellow (top 2 anomalies), and brown (top 3 anomalies).

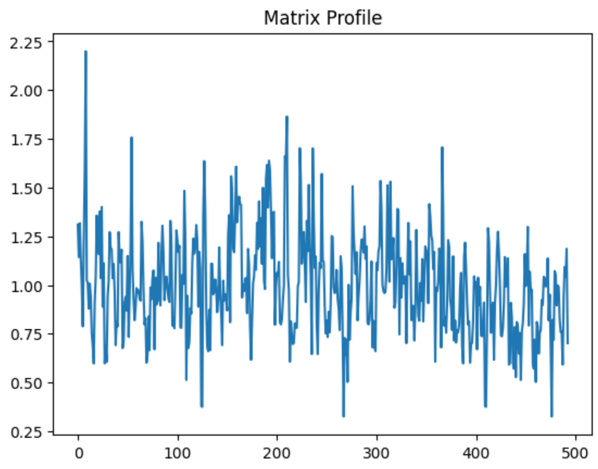


Figura 15-matrix profile

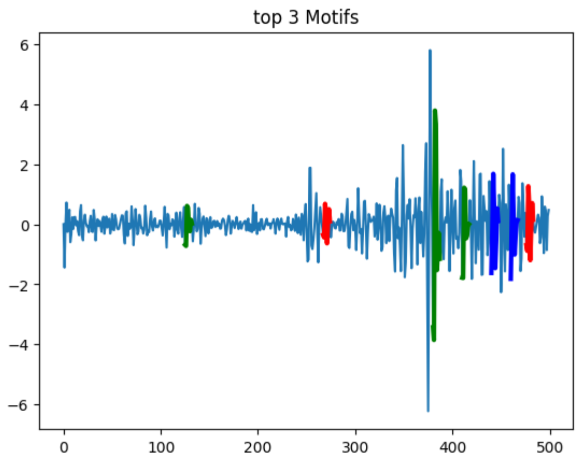


Figura 16-top 3 motifs

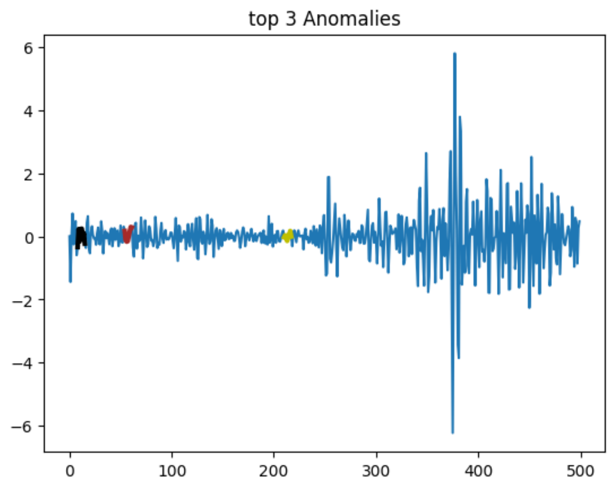


Figura 17-top 3 anomalies

4.3 Clustering

This section is focused on the study of clusters. All 2452 time series were considered, and two clustering techniques were applied: specifically, k-means and DBSCAN.

To execute the **k-means** clustering algorithm on the time series, it is necessary to predefine the number of clusters, denoted as k. To determine the optimal k value, a function was created that performed k-means on the time series matrix X for each value of k ranging from 2 to 10. The silhouette score was calculated for each k value, and the k value that yielded the highest silhouette score was chosen. In this case, the optimal value was found to be 3. For distance calculation in this algorithm, the 'dtw' metric was chosen as it showed lower distance between time series compared to other metrics.

Figures 18 and 19 display the graphs on the left with all 2452 time series, and on the right with the three distinct clusters highlighted in blue, green, and orange.

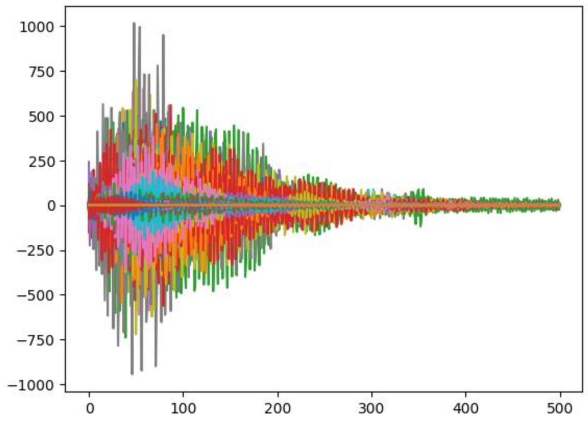


Figura 1-serie temporali

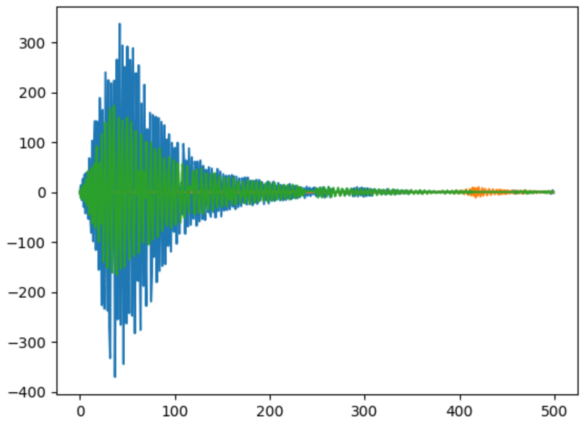


Figura 19-k=3-means clustering

The second clustering technique utilized is **DBSCAN**, which belongs to the family of density-based clustering techniques. Similarly, in this case, the number of min_samples and eps needed to be predefined. An arbitrary value of 5 was chosen for min_samples. To determine the optimal number of eps, experiments were conducted by calculating the silhouette score for all DBSCAN models with min_samples=5 and eps values ranging from 0.1 to 1 in increments of 0.1. It was observed that silhouette scores were 0 for eps values ranging from 0.1 to 0.6 and from 0.9 to 1. However, for eps values of 0.7 and 0.8, silhouette scores were found to be 0.16. Figure 20 depicts the graph of the silhouette score as a function of epsilon.

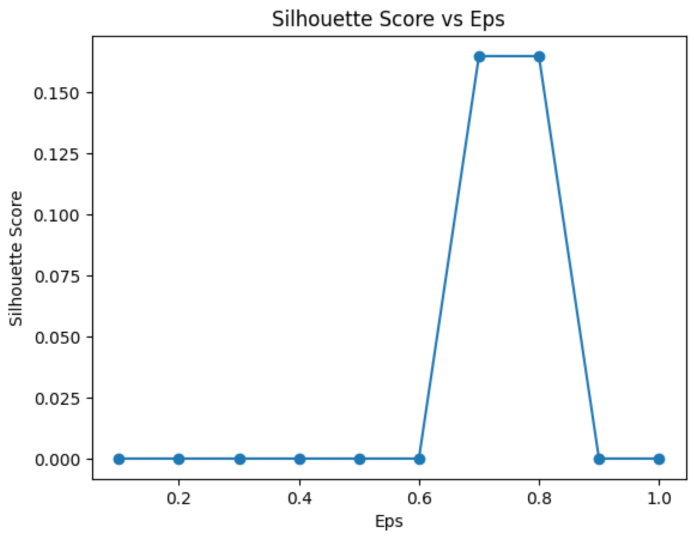


Figura 20

Thus, for the DBSCAN algorithm, `eps` was chosen as 0.8, and `min_samples` as 5.

Clusters can also be depicted using dimensionality reduction techniques. Both k-means and DBSCAN clustering algorithms were visualized using dimensionality reduction techniques. For the k-means method, **Principal Component Analysis** (PCA) was applied to the time series matrix X, extracting the first two principal components. As for DBSCAN, the **t-SNE** technique was selected, also extracting the first two dimensions. The values for k-means were kept constant at k=3, while for DBSCAN, `min_samples` were set to 5, and `eps` to 0.8.

Figure 21 illustrates a graph where the first clustering method, k-means, is displayed on the left after applying PCA, while the DBSCAN representation is shown on the right after applying t-SNE.

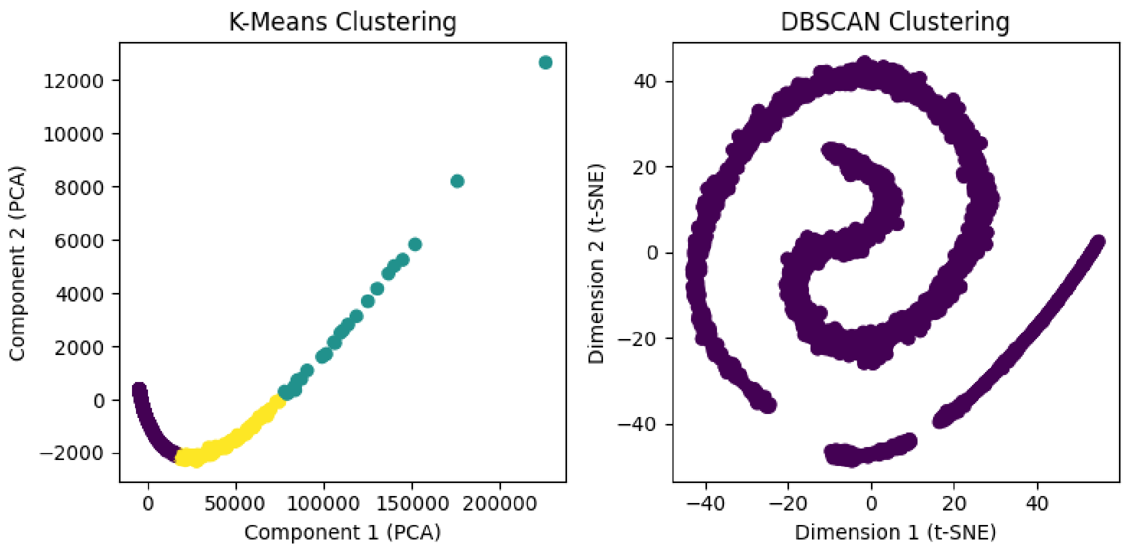


Figura 21

4.4 Classification

For the temporal series classification, the 'vocal_channel' variable with the categories 'speech' and 'song' was taken as the target variable. The data matrix of the time series was translated into a dataframe and split into a training set and a test set. The training set comprises records where actors fall within the range 01-18, while the test set consists of records where actors fall within the range 19-24.

The preliminary phase before classification involves extracting **shapelets**, distinctive patterns within time series that can be described as discriminative sequences of classes. For this purpose, the 'tslearn.shapelets' library was utilized. Based on the dimensions of the time series and the number of classes in the target variable, the number of shapelets was determined as 6, and their length was set to 50.

Subsequently, the shapelet model is created, trained, and evaluated using the accuracy score, which calculates the percentage of correct classifications by the model. The obtained accuracy score is 56%.

```
Correct classification rate: 0.5617128874388254
```

Thus, the visualization of shapelet correspondences (Figure 22) is performed for a specific time series with the following attributes: vocal_channel=speech, emotion=neutral, intensity=normal, statement=kids are talking by the door, repetition=1st, actor=11, sex=M.

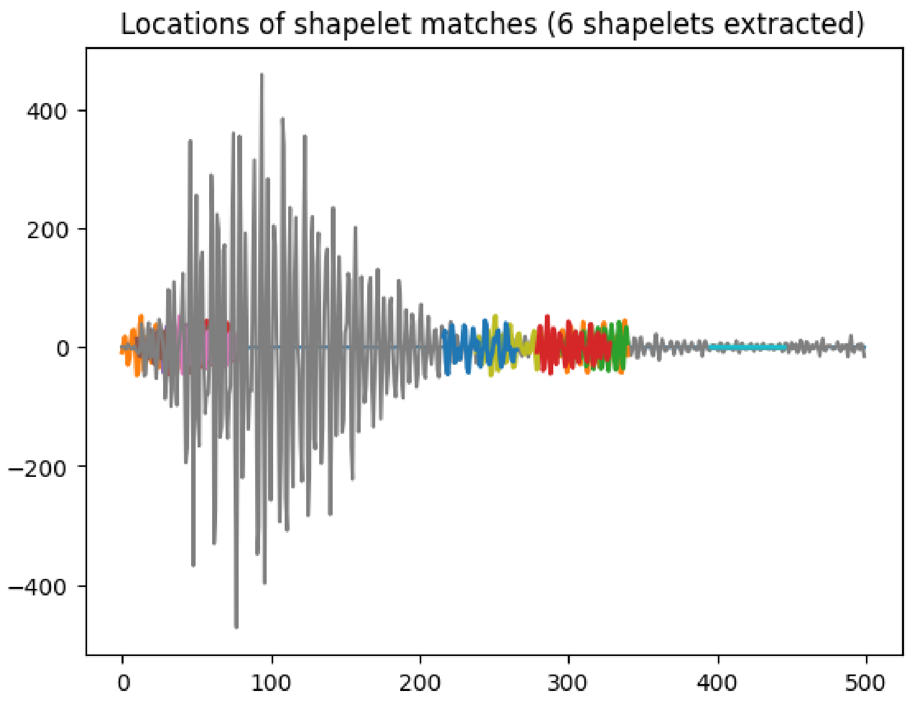


Figura 22

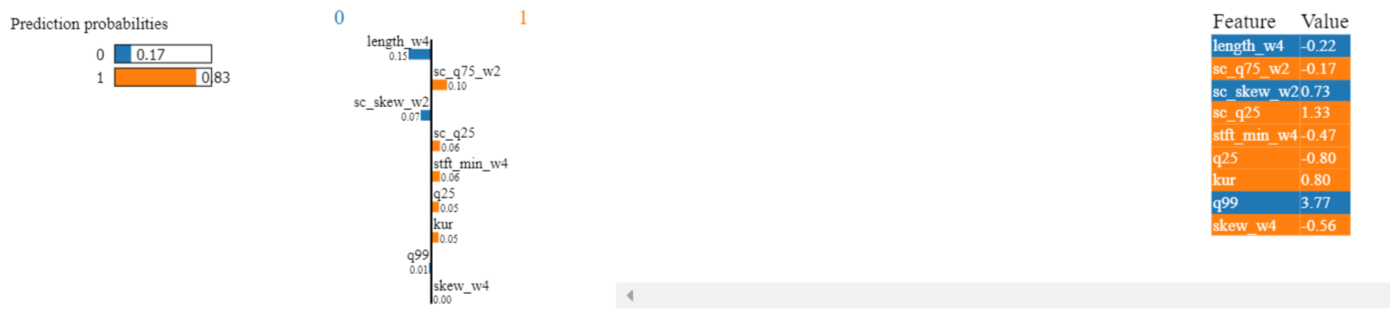
In terms of other classification methods, **both k-nearest neighbors** (KNN) with **Euclidean distance** and KNN with **Dynamic Time Warping** (DTW) distance were tested. The accuracy results were 0.575 for KNN with Euclidean distance and 0.640 for KNN with DTW distance. The F1 score for KNN with Euclidean distance was [0.33545648 0.68731269] for the 'song' and 'speech' classes respectively, while for KNN with DTW distance, it was [0.50834879 0.71596999].

A third classification method for time series, the **Canonical Interval Forest**, was also trained. This method selected a number of estimators equal to 30 and achieved an accuracy of 71% with an F1 score of 0.56 for the 'song' class and 0.78 for the 'speech' class.

Among all the classification methods tested, the **Canonical Interval Forest** yielded the highest results in terms of accuracy and F1 score. In general, the classification models struggle to predict the target variable accurately due to the significant reduction in the length of time series (from 160,000 units to 500) to allow Python to execute commands within 'reasonable' timeframes.

5 Explainability

Lastly, an **Explainability** task was conducted to make the classification performed in the interim of this paper more easily interpretable. The Random Forest classifier was used as a reference. The explanation methods employed were LIME and SHAP, which provide insights into the contribution of features towards a specific class. Both methods were applied to a randomly selected trace from the dataset. The results are depicted in Figures 23 and 24.



Feature	Value
length_w4	-0.22
sc_q75_w2	-0.17
sc_skew_w2	0.73
sc_q25	1.33
stft_min_w4	-0.47
q25	-0.80
kur	0.80
q99	3.77
skew_w4	-0.56

Figura 23-raffigurazione grafica del LIME

The first method mentioned above revealed that the features with the highest importance towards a 'speech' classification are respectively sc_q75_w2 with a contribution of 0.10, sc_q25, and stft_min_w4 with a contribution of 0.06. On the other hand, length_w4 is the feature that contributes the most to classifying the record as 'song' with a contribution of 0.15, along with sc_skew_w2 contributing 0.07.

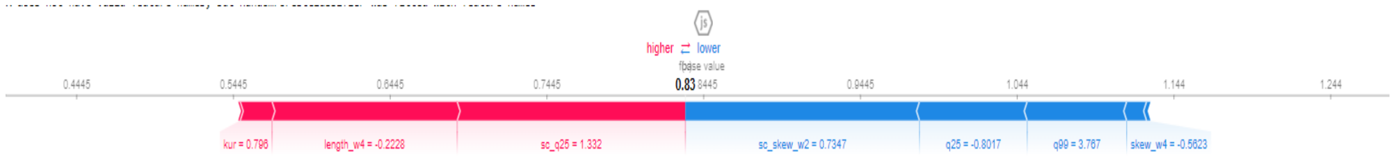


Figura 24- raffigurazione grafica dello SHAP

The second method, as evident from a comparison between the two graphs, confirms only a few results provided by the LIME method. Specifically, the feature length_w4 is depicted as the second attribute that contributes significantly to the 'song' class, whereas sc_q25 is considered the most influential feature for the same class. On the other hand, the values that drive the classification towards the 'speech' class are mainly sc_skew_w2 and q25.