UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

School of Telecommunications Engineering

Speech and singing voice classifier based on musical note classification and fundamental frequency estimation.

End of Degree Project

Bachelor's Degree in Telecommunication Technologies and Services Engineering

AUTHOR: Diana Sanchez, Santiago

Tutor: Ramos Peinado, Germán

External cotutor: LUZON ALVAREZ, CLARA

ACADEMIC YEAR: 2021/2022

Abstract

Speech processing is the study of speech signals and signal processing methods applied to their analysis and treatment. This processing is currently performed in the discrete domain, previously digitizing the speech signal. In this area, the distinction between the spoken voice and the sung voice is an essential task to help the achievement of various types of algorithms and their subsequent decision-making. This discrimination is sometimes difficult even for humans, depending on the type of music on which the sung voice is based. To achieve a good discrimination between both classes (spoken and sung), a deep and broad study must be carried out due to each type of voice may require different processing and decision algorithms. This work presents an automatic classifier between sung and spoken voice or speech, based on two main parameters: the tone derived from the classification of musical notes and the estimation of the fundamental frequency. This method obtains good results in the task of discriminating between silence, speech and sung voice. This work has a direct industrial application in the field of audio research, the application of filters or algorithms depending on whether there is sung voice or not, or the discrimination of musical styles from the parameters extracted from the sung voice.

Resumen

El procesamiento del habla es el estudio de las señales del habla y de los métodos de procesamiento de señal aplicados al análisis y el tratamiento de las mismas. Este tratamiento es actualmente realizado en el dominio discreto, digitalizando previamente la señal del habla. En este ámbito, la distinción entre el voz hablada y voz cantada es una tarea esencial para ayudar a la consecución de varios tipos de algoritmos y de su posterior toma de decisiones. Esta discriminación es a veces difícil incluso para los humanos, dependiendo del tipo de música en el que se basa la voz cantada. Para lograr una buena discriminación entre ambas clases (hablada y cantada), se debe realizar un estudio profundo y amplio debido a que cada tipo de voz puede requerir de un procesado y de algoritmos de decisión diferentes. Este trabajo, presenta un clasificador automático entre voz cantada y hablada o discurso, basado en dos parámetros principales: el tono derivado de la clasificación de las notas musicales y la estima de la frecuencia fundamental. Este método obtiene buenos resultados en la tarea de discriminar entre silencio, habla y voz cantada. Este trabajo tiene una directa aplicación industrial en el campo de la investigación del audio, la aplicación de filtros o algoritmos en función de si hay voz cantada o no, o la discriminación de estilos musicales a partir de los parámetros extraídos de la voz cantada.

To my parents. Thanks to them, this work, which encompasses my passage through the academic world, would not have been possible. I would also like to thank my grandparents, whose vital support makes this task undoubtedly easier. And to all the people who believe in me, what I do and what I transmit. Especially, to Voicemod, the company that has accompanied me throughout this process and has given me the opportunity to meet excellent professionals and people.

Contents

List of Figures

Part I

Memory

Chapter 1

Introduction

The field of speech processing has been strongly developed in the research and industrial fields in recent years. This work allows the study of the emerging technologies, which are used both by standard users in everyday tasks (e.g. interaction of people via verbal language with intelligent devices) and by powerful industrial applications of speech synthesis, speech conversion or natural language processing achieved through complex artificial intelligence models.

What could be the most well-known example of speech analysis and recognition is "Alexa", from Amazon or "Siri" from Apple [1].It includes onboard microphones to record and detect speech using an Automatic Speech Recognition system. Other applications can be considered and explained regarding to the speech processing analysis, as "Call-centers" or "Chatbots" [2], translation programs between languages in real-time [3], or automatic text generation [4], among others [5].

For the achievement of the above mentioned applications, several types of algorithms and techniques can be used. Two of them are the most used: Artificial Intelligence applied to audio and Classical Digital Signal Processing Techniques. In this work, both approaches will be combined depending on the model or part of the algorithm that is going to be implemented. These deep learning techniques are usually used in the most basic modules (speech recognition and synthesis) common to many applications, but the upper layers, those that implement the system's understanding and intelligence capabilities of the sys-

tem, usually still use a lot of expert knowledge, which is difficult to reuse, so it is difficult to implement with Artificial Intelligence. Also classical digital signal processing is used in order to pre-process and feed the neural networks [6].

Speaker style recognition, whether someone is singing or speaking, is one of the multiple applications in this field. A good discrimination allows us to apply another filters or algorithms depending on when the user is singing. For example, voice synthesis is a well known field in which Speaker Style Recognition could be used, being applied in different applications such as live concerts voice synthesis or addition of filters and electrical instruments to the voice while singing.

1.1 Objectives

Taking into account all that has been stated in the introduction, the purpose of this work is the implementation of an open source algorithm for a sung and spoken voice classification. Having studied the state of the art of this type of voice processing, a Reference Paper was found, owned by the University of the Basque Country, Spain (UPV/EHU) [7] [8]. The second paper mentioned is an extension of the Reference Paper subsequently published. This paper was published in November 2018, as a presentation of a disruptive method in the field of speech segmentation and sung voice, which was part of the following doctoral thesis, from the University of the Basque Country.[9]

The aim of this work is the implementation of the algorithm proposed in this Reference Paper in an off-line mode and then to perform a novel real time implementation of this classifier.

For the development of this algorithm, several sub-algorithms must be previously investigated and implemented before, like VAD; a pitch extractor; and a musical note detection algorithm.

This project and its possible applications are part of an internship at VOICEMOD [10] S.L. This company has created an application that applies effects on the human voice in real time. The objective of this thesis is the generation of an algorithm that can be applied in certain aspects of the Voicemod application, for example, in the application of

certain effects on the voice only when the user is singing such or the synthesis of musical instruments on the voice when the user is singing.

1.2 Methodology

In this section we will briefly describe the methodology used to obtain the presented algorithm.

First of all, the programming language used was Python, version 3.10. This decision was made because it has an environment (strong open source libraries and frameworks, multipurpose, portable and usable in many well-known IDES) that allows the quick and easy to understand creation of signal processing prototypes because of the ease with which this language makes it possible to use machine learning libraries.[11]

Following in this line, during the realization of the algorithm, several Python free license libraries have been used, such as NumPy, SciPy, librosa, and especially Sklearn[12], [13], [14], [15].

This last library has allowed the generation of two ML (Machine Learning) models that have been implemented in two fundamental parts of the algorithm, the voice and silence detection and the decision function. For this, supervised Machine Learning algorithms such as GMM (Gaussian Mixture Model) or SVM (Support Vector Machines) have been possible to implement thanks to this library.

A GMM (Gaussian Mixture Model)[16] is a probabilistically-grounded way of doing soft classification of different classes. Each cluster corresponds to a probabilistic solution, that in this case is a gaussian. The SVM (Support Vector Machine)[17] is a really similar model, which analyzes data for classification and regression analysis. These models will be extensively explained in the following sections concerning the voice detection and decision function.

Finally, it must be commented that, in Python, there are several libraries that could achieve part of the work done , but that do not allow the integration of the algorithm in a real time platform. The reason why this happens is because they add a lot of delay, even

though they are implemented with less code lines. A study about this topic was done at Voicemod before the implementation of this algorithm, and that is why these resources were not used. Among these libraries is PyWorld, or Speaker Verification Toolkit among others, which perform voice detection or timbre detection work with fewer lines of code but with worse performance in real-time, which is the final objective of this work. [18]

1.3 Structure of the thesis

This section aims to briefly outline the general structure of the thesis:

- In chapter 2, the state of the art of the different technologies that implement similar algorithms to the one implemented will be investigated and explained, comparing different techniques that are used in order to achieve this discrimination and the reason why finally the one implemented was chosen. State of the art of sub algorithms such us Voice Activity Detection, Note Detection and Decision Function will be also investigated in this chapter.
- Later, in chapter 3, the general implementation of the algorithm will be explained, and later, a more detailed explanation of each part will be presented, step by step through its theoretical explanation and its implementation.
- After the algorithm explanation, in chapter 4, the experiments done and the results achieved will be presented and compared to the Reference Paper results.
- Throughout chapter five, conclusions drawn will be described after the analysis of the metrics obtained. Also possible further works will be presented.

Chapter 2

State of the art

In order to understand the current available technologies regarding to this topic and explain the reason why this algorithm was chosen, a brief summary of the state of the art techniques of speech and sung voice recognition and discrimination will be presented.

During the last years there have been several studies dealing with the analysis of technologies both for speech and singing [19] [20] . The interest in analyzing and treating the characteristics of the singing voice has increased exponentially lately. The singing voice is an essential part of music, serving as a communication channel for lyrics and rich emotions, so its discrimination is useful for research areas related to emotions of speech, automatic lyrics alignment, singing melody transcription, vocal melody extraction, singer identification or singing voice synthesis [21].

The classification done in this work between sung voice and speech is based on two parameters derived from the pitch: the percentage of notes detected and the percentage of voiced frames. These parameters are derived from short term features (MFCC) of the audio and f0 (fundamental frequency) estimation, which are the two technologies in which the algorithms presented are based. As a brief introduction, the fundamental frequency of an audio refers to the approximate frequency of the "quasi-periodic" structure of voiced speech signals [22]. In the case of MFCC, they are coefficients for the representation of speech based on the perception of human hearing. They show the local characteristics of the voice signal associated with the vocal tract.

These technologies will be explained in detail in the following sections of this work. Here are also presented other methods that are based on other types of features, that show worse metrics in comparison to the algorithm implemented.

1. The first approach known about this technique was proposed in 2001 using statistical features and the Hidden Markov Model (HMM) as a classifier [16] [23]. It was able to train an effective SVD (Singing Voice Detector) that was 80% accurate.

2. MFCC (Mel Frequency Cepstral Coefficients) and GMM (Gaussian Mixture Model) based algorithm . Another possibility was to base the whole decision only in a GMM trained with the MFCC of the audio and their $\triangle$ (which is another implementation of MFCC)[24] with a short frame period [7]. These calculations give a good performance but not enough compared to the method presented in this work.

3. DFT of F0 distribution (Discrete Fourier Transform of the Fundamental Frequency Distribution). F0 provides useful information to discriminate between speech and singing. One possible method is the analysis of the histogram of the f0 curve. It gives information about the distribution of f0 values, that for singing voices are surrounding a frequency that corresponds exactly to a musical note, with a possible little deviation. This method generates worse metrics than the one presented in this work, and that is why it was discarded [7].

4. Deep learning models. Convolutional Neural Networks and Recurrent Neural Networks. There exist different amounts of data which could be used in order to train these networks: MFCC features, FFT (Fast Fourier Transform) features, among others. This way to achieve the discrimination of speech and sung voice is one of the best found, but it lacks a note detection algorithm, which could help us in other aspects of the sung voice research field. Also the metrics shown in the reference papers in any case, surpass the algorithm implemented in this work [20]. A deep neural network is shown in figure 2.1, which shows the structure of neurons and layers that make up the complete neural network [25].

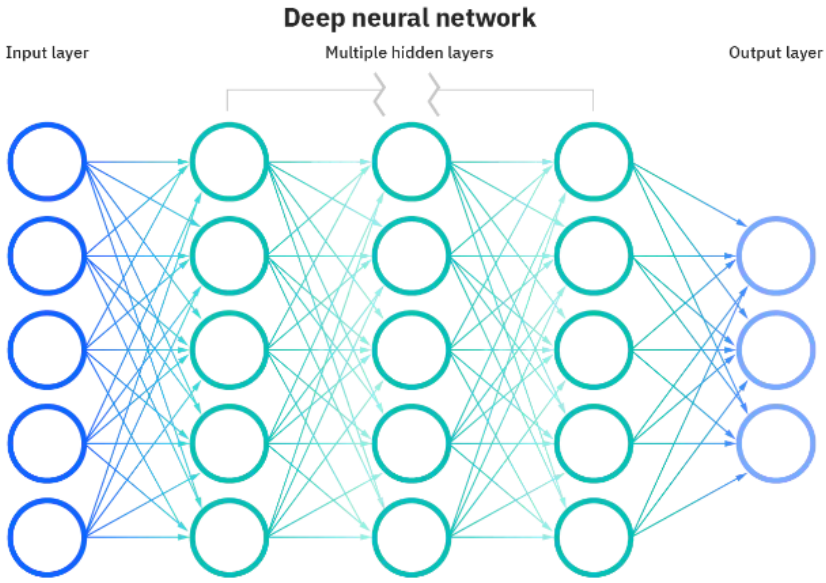

Figure 2.1: Deep Neural Network (extracted from [25])

5. Different features. ZCR (zero-crossing rate) is the measure of sign-changes along a signal . The interesting observation of this rate is that the standard deviation of this feature is higher for speech signals than for sung signals [20], [26].

   This feature is used to train a SVM (Support Vector Machine) to make the decision. It hasn't been used because of the lack of metrics and information related to the topic. In the figure 2.2 the ZCR of an audio signal can be observed. The larger the curve, the more sign changes there are in the audio signal.

6. Phoneme-alignment based classifier [27]. The time boundaries for each phoneme could be a feature of discrimination. It was not proved and studied, so that is why it was discarded.

As it can be seen in the presentation of the state of the art methods to develop a good discrimination of sung voice and speech, MFCC are usually used characteristics in this task and that is why the model presented is based in part on this features. Also GMM's and SVM's are very common models used in almost every proposal of this segmentation, and they are implemented in this work too.

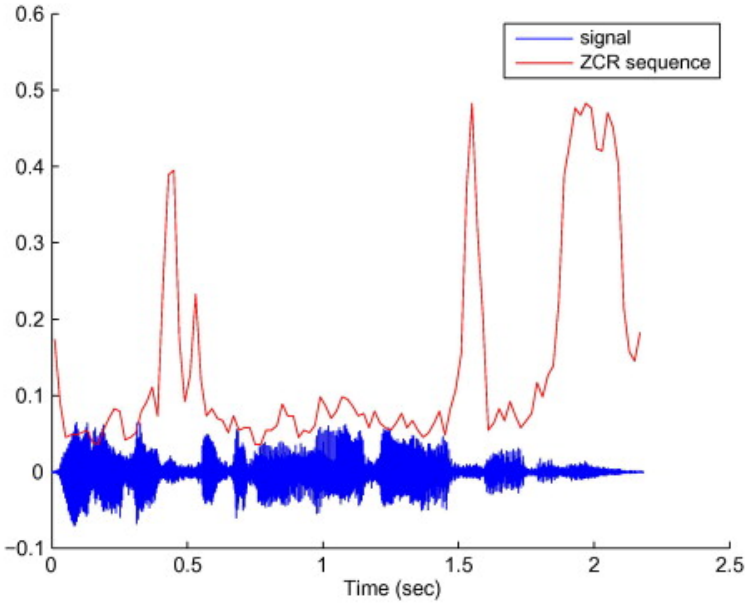

Figure 2.2: Zero Crossing Signal of an audio file (extracted from [26])

None of the above commented novel methods presents better metrics than the Reference Paper [7] that is implemented in this work. Moreover, here an important Note Detection Algorithm has been created, and it allows us to apply the algorithm implemented in other applications directly related to music synthesis, which is the state of the art application of this type of method.

2.1 Sub-Algorithms' State of the art

For the complete realization of this algorithm, a chain of sub-algorithms is created to allow the execution of all the functions required by this work. Among these sections are: Voice detection; Note Detection Algorithm and the Decision Function, among others.

In this chapter, the techniques that represent the state of the art will be discussed, and the reason why the version that has been implemented was finally chosen will be given.

2.1.1 Voice Activity Detector State of the art

The speech detection algorithm is the technique that detects the presence or absence of speech in an audio signal.

There are many configurations of a VAD because it is a technique that has been widely used for a long time for many applications in the field of speech processing[28].

The classification of the audio signal as speech or silence is achieved after the classification of certain audio characteristics, which can range from the energy divided by frames to the MFCCs discussed above. In the figure 2.3, the VAD flow chart can be observed. The type of feature extracted depends on the version of the algorithm used.

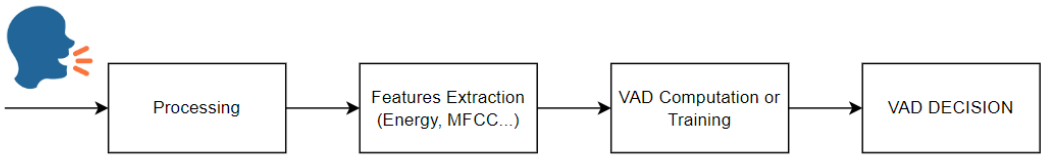

Figure 2.3: Voice Activity Detection process

One of the most basic configurations, and therefore discarded in this project, is the energy-based VAD. The idea, as the name suggests, is to perform a classification based on the energy of the signal. It is easy to understand that the energy in the segments that are spoken is much higher than the silent segments.

However, there are certain assumptions that an optimal speech detection algorithm should meet that are not possible with energy-based VAD. For example, the VAD should be adapted to non-stationary background noise. This is an important problem for this type of VAD, because in many occasions the energy value can be very even between a spoken segment and a segment with background noise [29].

Another possible approach is a SVM based VAD which, given a training data set, builds a model that assigns new examples to a category, generating a hyperplane that optimally separates one class from another, seeking to maximize the distance between classes.

This machine learning model is very interesting and it does better support situations with background noise, but having been studied a posteriori, once the whole algorithm was created, it offered a worse performance than the GMM-based VAD that was finally implemented. Therefore, a VAD based on SVM is also a very good option, and depending on the application it could even be better than the one implemented in this work, depending on the data used and the dataset environment [30].

After having studied both previous models, and also following the implementation of the Reference Paper, the chosen model was the GMM based VAD. As explained above, GMM is a probabilistic model that assumes that all data are generated by a finite mixture of Gaussian distributions with unknown parameters.

The performance that this model has offered in the Reference Paper was one of the reasons for applying this model. In addition, it is widely applied globally in the creation of speech detection algorithms [28].

2.1.2 Note Detection Algorithm State of the art

The Note Detection Algorithm, as its name suggests, is responsible for detecting musical notes on audio frames classified as voice according to the Voice Activity Detector.

The state of the art of this algorithm is less prolific than that of the Voice Activity Detector. Moreover, there is a software called "Tony"[31] that could be considered as the most innovative implementation regarding to this topic right now.

Tony Software is a software created by Queen Mary University of London in 2014. It is a tool for the interactive annotation of melodies from monophonic audio recordings, and evaluate its usability and the accuracy of its note extraction method.

Tony software performance is excellent in all aspects. In addition, it contains a graphical interface that shows not only that the note has been detected, but also what note it is and how long it has lasted.

The implementation of an algorithm very similar to Tony is the goal of this work. The detailed explanation in the corresponding section will explain how it has been developed to try to achieve a performance similar to this state-of-the-art software.

2.1.3 Decision Function State of the art

The decision function is the last step of the classification algorithm between sung and spoken voice. This function receives data from the segments categorized as speech by the VAD and classifies them as sung voice or spoken voice. Segments categorized as silence do not pass through this function, as they are directly categorized by the VAD.

The application of the decision function was possible with 2 options: a function made by hand at the frame level that made decisions without prior training or the implementation of an SVM that was previously trained and that, by passing certain parameters, made the decision.

The decision function made by hand has a clear problem. The limits to be designed between both classes (sung voice and spoken voice) are integers that in a 2D graph are only capable of generating a straight line, in no case a curve.

This is a problem, since on many occasions the dataset used has a distribution in the plane that is not uniform and cannot be well classified with a straight line.

This is where the SVM comes into play, which is able to perform an accurate classification with polynomial distinction functions. This is the main reason why SVM was finally chosen for the creation of the algorithm's decision function [17]. The possible curves and configurations of the SVM will be explained in more detail in the corresponding section.

As it can be seen in the figure 2.4, with this distribution of both classes, it could not be possible to differentiate them with a straight line. That is why the model that does not allow polynomial curves is not the best one to be used, as it will cause so much errors.

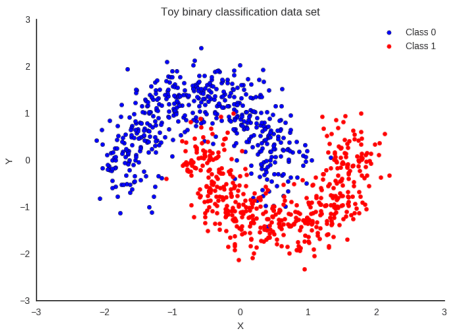

Figure 2.4: Binary classification (extracted from [32])

Chapter 3

Explanation of the Algorithm Implementation

3.1 General Explanation of the Algorithm

The implemented algorithm consists of five clearly differentiated parts. The objective to be achieved is the discrimination between sung and spoken voice. For this purpose, it is necessary to calculate the two parameters mentioned above: the percentage of frames classified as musical notes (PN) and the percentage of frames classified as voice (PV).

In order to calculate both parameters, the first step is to implement a voice and silence segmentation algorithm, also called Voice Activity Detector. In the case of this algorithm, and following the dynamics of the Reference Paper, a GMM based VAD has been implemented. This first step of the algorithm distinguishes between voiced frames or silent frames, so that those that are classified as speech are still analyzed in the rest of the algorithm while the silences are already passed to the final decision vector. A 10ms frame size has been used all around this algorithm.

The next step is the calculation of the f0 (fundamental frequency) of each frame with voice. For this purpose, an algorithm that calculates the fundamental frequency of each of the frames separately and obtains a curve has been implemented. This will allow the calculation of musical notes under certain conditions.

These conditions are applied in the Note Detection Algorithm. This is the most complex, complete and usable section of the algorithm, because it can be applied in many fields of speech processing and sung voice [20]. The Note Detection Algorithm is responsible for detecting musical notes in the audio signal. To do this, the frames containing speech must meet two conditions: be longer than 150ms and not vary more than 100 cents on the f0 curve in cent scale [33]. A larger explanation of this scale will be done in the proper section.

Once the vector that detects musical notes has been obtained, all that remains is to calculate both parameters and implement the decision function of the algorithm. To do

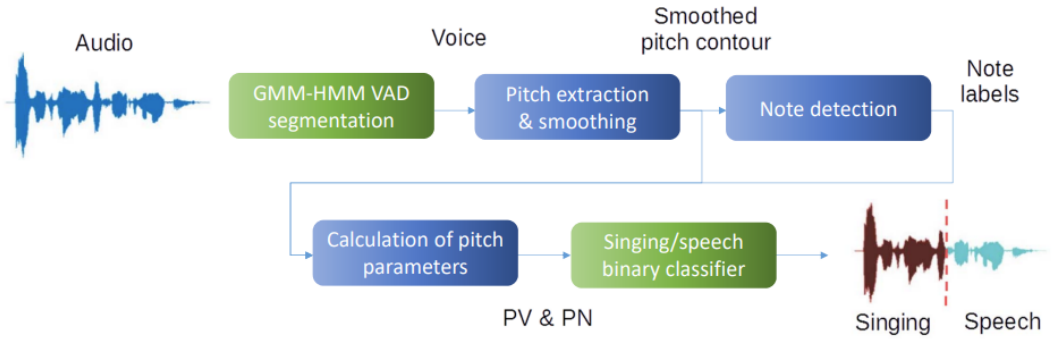

Figure 3.1: Flow chart of the algorithm (extracted from [8])

this, a window of 200 ms is applied. It moves around the entire audio and calculates both parameters: the percentage of frames cataloged as voice and the percentage of frames cataloged as voice. As can be seen, for a correct calculation of both parameters, a frame level analysis is necessary from the beginning of the algorithm in the voice detection or the calculation of the f0.

Once both parameters have been calculated, they are embedded into the decision function. A SVM (Support Vector Machine) has been implemented, which is a supervised machine learning model that is able to discern between both classes (spoken or sung), being previously trained with a labeled dataset and with both parameters previously obtained. The labeling that has been performed will be explained in the next sections regarding the datasets used.

The result extracted from the SVM is the decision vector that yields the result of the algorithm which classifies as zero the silences, as one the spoken segments and as two the sung segments. The decision is made at the frame level, i.e., each frame is classified into one of the three possible classes.

In the figure 3.1 the flow chart of the algorithm is shown. As it can be seen, every part of the algorithm commented above is contained in there, reaching the solution taking into account both PV and PN parameters.

3.2 GMM Based Voice Activity Detection

Voice Activity Detection (VAD) is the task of identifying the presence or absence of speech in an audio signal. Many versions of this algorithm are widely known, but in this project will be implemented a GMM based VAD, as [7] and [8] propose for the first part of the algorithm. GMM Voice activity detection is widely used in the "Speech and signal processing" field because it is a must in clustering and segmentation tasks and it has a very accurate performance due to the structure of the GMM model itself.

3.2.1 Gaussian Mixture Models

The Voice Activity Detector implemented is based on the Gaussian Mixture Model (GMM), which is a probabilistically-grounded way of doing soft classification. Each cluster corresponds to a probabilistic solution, that in this case is gaussian .

K-means clustering is an algorithm of non supervised classification which groups objects into k groups based on their characteristics. It assigns each object into a group depending on which centroid is the nearest to that object [34].

GMM is a generalization of k-means algorithm but incorporating more information about the covariance structure of the data and the center points of the gaussians curves. Instead of assigning each observation to a single cluster, a probability distribution of belonging to each cluster is obtained.

Then a better calculation of the parameters of the gaussians can be achieved.

What might be discovered by applying a GMM algorithm are the parameters of the mean and covariance of each gaussian, as it can be seen in the figure 3.2.

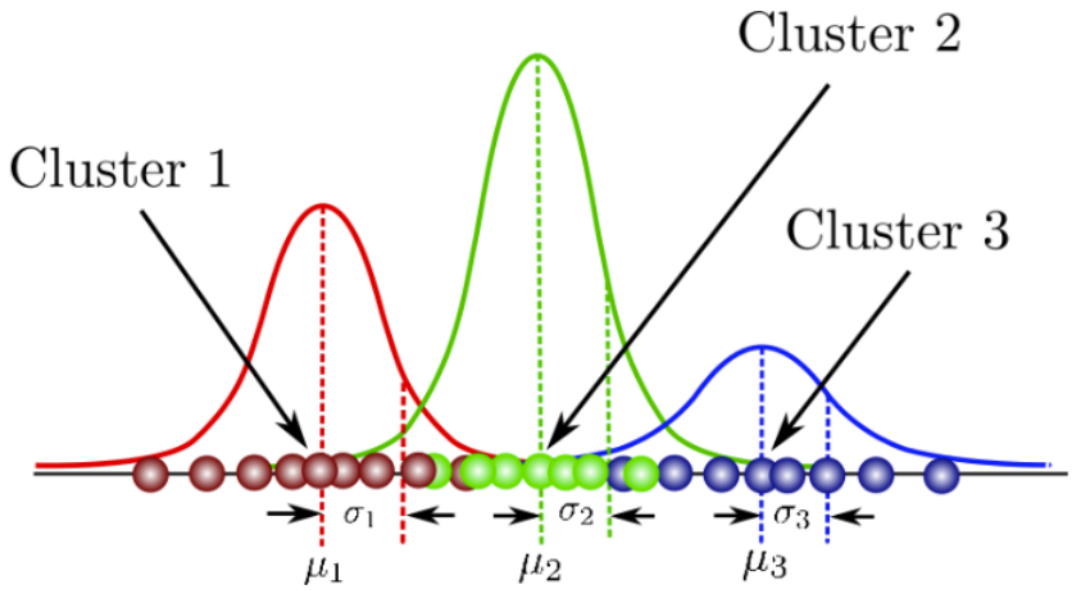

Figure 3.2: Clustering of the Gaussian Mixture Model (extracted from [35])

3.2.2 Expectation-Maximization Algorithm

The Expectation Maximization algorithm is the way to find the two parameters that need to be calculated from each gaussian: variance and mean. It starts by placing the gaussians randomly in the space domain where the points are located with a random mean and variance. For each point, the probability of coming from each Gaussian must be calculated.

Later, both mean and variance of each cluster must be adjusted depending on what points were assigned to which Gaussian, to try to fit those points in the Gaussian that was assigned to them.

That is the way of doing soft clustering, because it is taking into account the probability of belonging to each Gaussian, instead of only being able to belong to one, as k-means method would do.

The GMM implementation can vary between different configurations, because the number of Gaussians that can be applied to the algorithm is variable. As it is explained in the Reference Paper, it has been completed an intense study of how the number of gaussians affects the performance of the algorithm. It can be concluded that the number of Gaussians does not significantly affect to the metrics of the GMM based Voice Activity Detection.

That is the reason why 32 gaussians have been chosen, in order to reproduce as faithfully as possible the algorithm of the Reference Paper. Using 32 gaussians means that 32 different gaussian distributions are calculated, although this classification is only between 2 classes. In this case, each class (silence or speech) belongs to a lot of gaussians. If there were two gaussians only, each class would have only one gaussian, and perhaps some points located far from both classes would be misclassified. That is why a big amount of distributions is used, to be able to locate more distant points.

3.2.3 Implementation of the GMM by SCIKIT-LEARN

The implementation of the Gaussian Mixture Model has been done using ”sklearn” from Python. Scikit-learn [13] is an open-source machine learning library for Python that includes several algorithms related to classification, regression and groups analysis. In this case, the class sklearn.mixture. In figure 3.3 the most important elements of the library can be found.

The best way that was found to achieve a good performance of the Voice Activity Detector was to implement two GMM's, one for voice and another for silence. The creation process was:

1. Training of both GMM classes with the corresponding number of gmm components, the audio MFCC's and the audio labels. The audio labels were scripts manually created, that label each frame of the audio file with the proper solution, which in this case is '0' for silence and '1' for voice. In the training dataset there were 164918 frames and in the testing dataset there were 223150 frames. With both MFCC and labels, the GMM is able to learn to discern between silence and voice with a high level of accuracy.

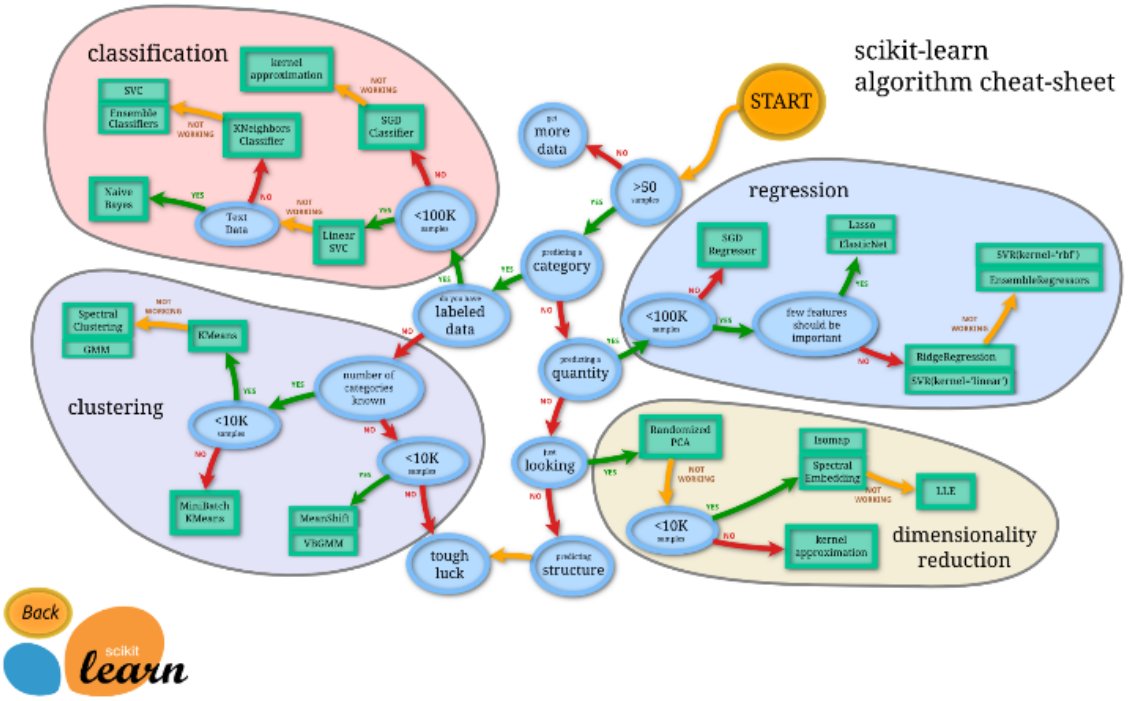

Figure 3.3: Sklearn Library Graph (extracted from [13])

2. Once the model is trained, the method "score samples" from the class is used in order to calculate the likelihood of belonging to silence class or voice class, having as parameter the MFCC of the audio dataset being trained.

3. Test of both classes with the test dataset, that will be exposed in the next sections. Due to the structure of the GMM, each training with the same database yields different results in the testing phase. That is why, in this stage, a repetitive training of the GMM was performed until the test metrics were the best possible.

4. Once the best training was found, the configuration is saved.

Now that the best possible configuration of the VAD having been tested with the proper dataset is available, any audio can go through the VAD and it can be analyzed. If the audio labels are available, some metrics such as accuracy, f-score or confusion matrix can be calculated through the python script.

The configuration of the VAD can be found in 2 files ".sav": "Parameters_silence_VAD.sav"; "Parameters_voice_VAD.sav"; that later are called from the main function when performing the Voice Activity Detection.

The flowchart of the GMM based VAD can be found in the figure 3.4. Labels and MFCC are used in order to train the classifier. Then, when an unkwown audio comes up and its frames want to be predicted, MFCC's are extracted and go trough the already trained classifier.

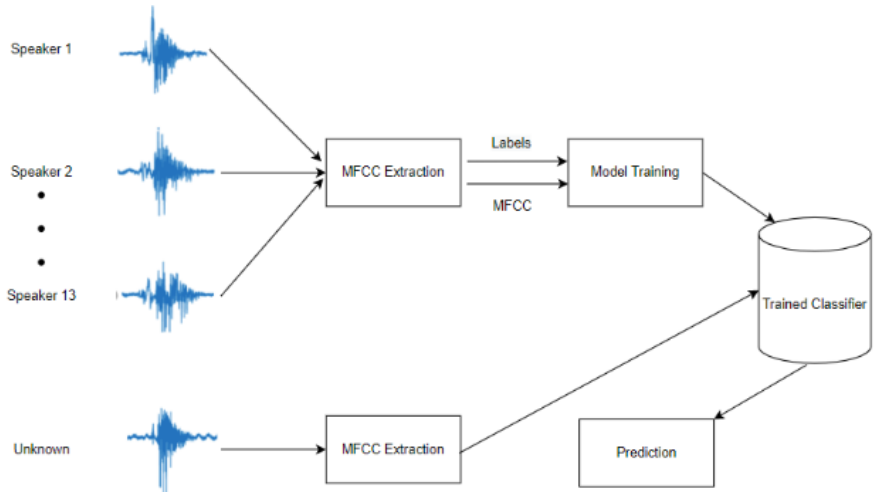

Figure 3.4: Voice Activity Detection training and test graphic

3.2.4 Mel Spectrogram Cepstral Coefficients

For the VAD training, MFCC are used. These coefficients are a representation widely used in audio processing techniques[36].

MFFC's are coefficients for the representation of speech based on the perception of human hearing. They show the local characteristics of the voice signal associated with the vocal tract. Basically, MFCC are calculated by applying the Discrete Cosine Transform to the Mel Spectrogram [37].

A spectrogram is a visual representation that makes it possible to identify the different variations in frequency and intensity of sound over a period of time. Studies have shown that humans don't perceive frequencies on a linear scale. That is why the Mel scale

was created, because it is a unit of pitch such that equal distances in pitch sounded equally distant to the listener. This scale is a logarithmic representation of the frequency domain. Therefore, the Mel Spectrogram is a spectrogram where the frequencies have been converted to the Mel scale. The equation 3.1 converts to "mels" the Hz scale, in order to be able to configure the Mel Spectrogram, which uses the Mel scale.

$$m = 1127 * \ln(1 + \frac{f}{700}) \tag{3.1}$$

In the figure 3.5, the relation between Hz and mels is shown, and that defines the relation between the standard spectrogram and the Mel spectrogram.

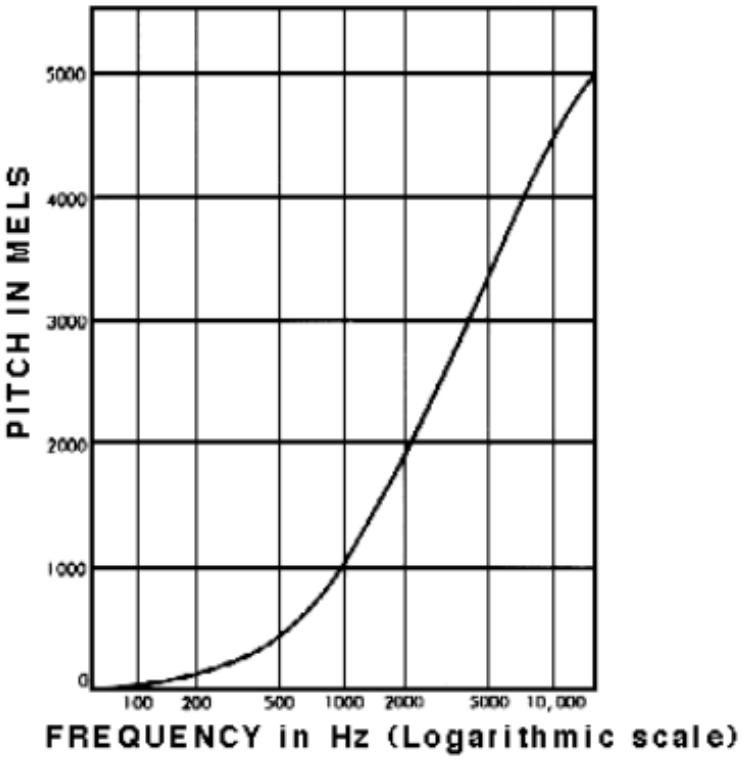

Figure 3.5: Hz to Mel scale function [37]

In the figure 3.6 there is a comparison between both spectrograms, in where the clear difference is the y-axis with a logarithmic representation in the Mel spectrogram case.

MFCC's are calculated frame by frame, using the feature.mfcc from librosa [15]. Librosa calculates for each frame a vector of 13 characteristics that, together with the labels, are

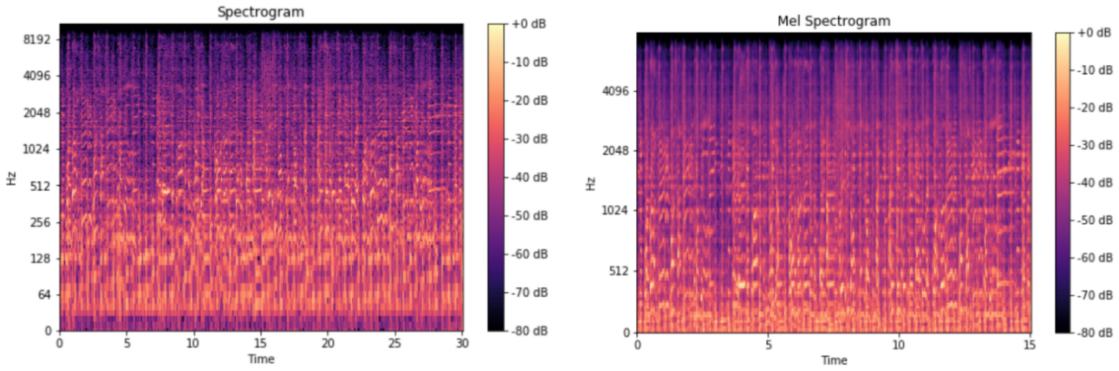

Figure 3.6: Spectrograms (extracted from[37])

embedded in the fit method [38] of sklearn.GaussianMixture in order to train the VAD. The fit method is the one that estimates the model parameters using the Expectation Maximization algorithm. The method sets the parameters with which the model has the largest likelihood or lower bound.

In the figure 3.7, the process of calculation of the MFCC is shown. The Mel Frequency Filter Bank is a bank of filters based on the Mel Scale that are applied to the audio already windowed. After that, the logarithm is applied and the DCT to be able to perform the MFCC.

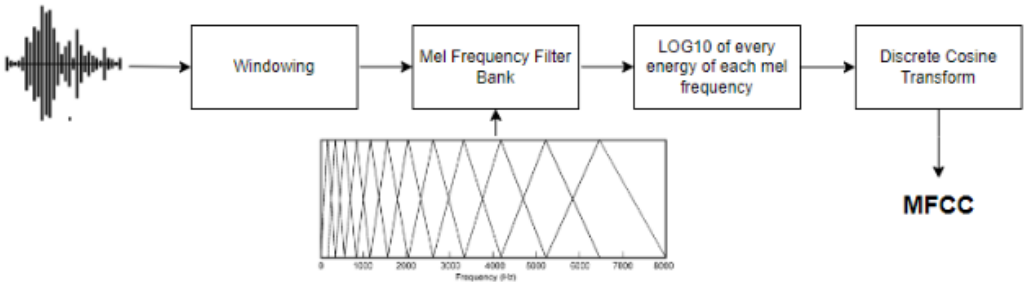

Figure 3.7: MFCC calculation

3.3 F0 Extraction

The human voice can be analyzed by means of spectrograms, and thanks to them, we can extract different parameters that characterize the sound waves, as for example, the frequency or the intensity depending on time.

Of the whole set of simple waves emitted by the human voice when it makes an oral production, the wave with the lowest frequency of all those emitted is known as the fundamental frequency (f0).

The fundamental frequency value of an audio signal is widely used in the field of speech processing . It is a value that provides a lot of information for analyzing what kind of person is delivering that speech, whether it is a female or male voice, among other characteristics. It is also capable of analyzing the characteristics of the speech itself in the sense of knowing the duration of the spoken segments, the pronunciation of vowels or consonants, and, with respect to this work, the analysis of musical notes [39].

The fundamental frequency of the audio signal entering the algorithm implemented in this work serves to discern between musical notes or not in the subsequent note detection algorithm. In the following section, the application of the fundamental frequency in this algorithm will be fully explained.

A comparison of different f0 values, depending on the gender of the speaker is performed in the figure 3.8. It can be seen that the man's f0 curve is clearly slower than the women's.

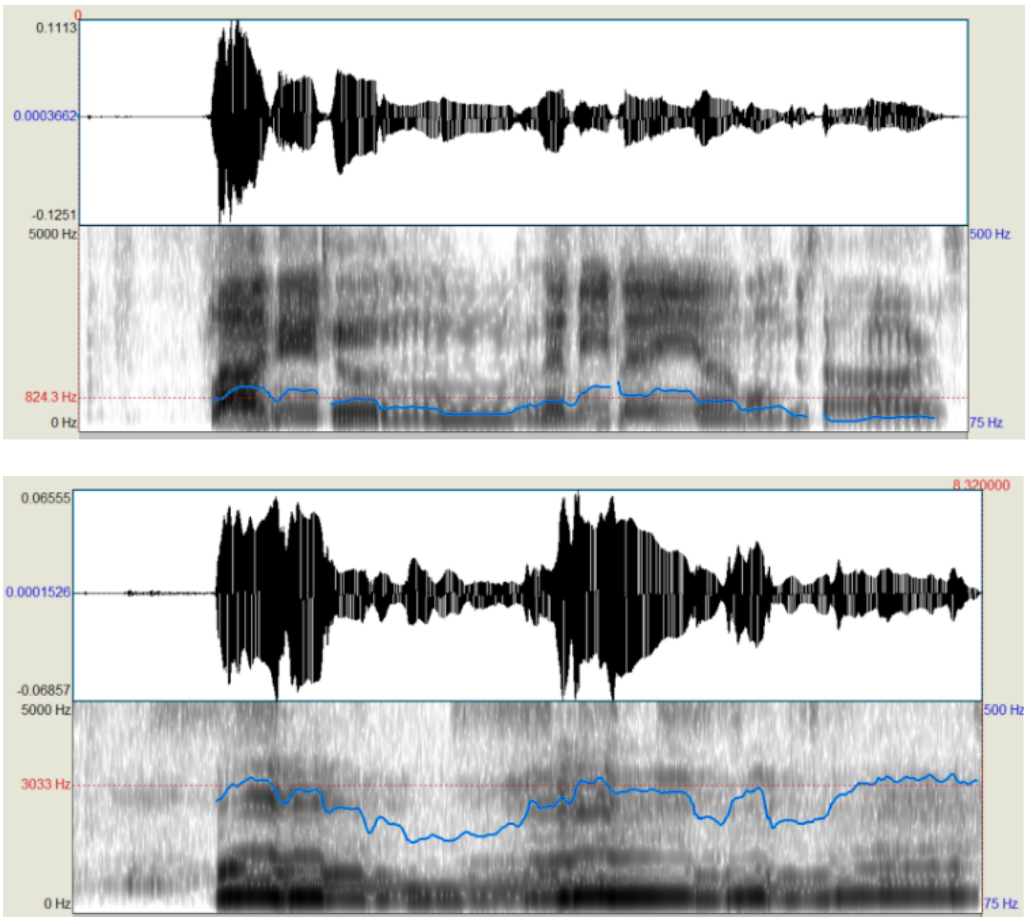

Figure 3.8: F0 comparison between man's and woman's voice (extracted from [40])

3.3.1 F0 implementation

This project, as commented before, is carried at the context of an internship at VOICEMOD. Due to the activity that this company performs, pitch extraction algorithms are common and recurrent. That is why it was offered to this project their own pitch extraction algorithm based on DNN [25].

A study among many possible algorithms was performed within VOICEMOD, and for this project the DNN based algorithm was selected, which cannot be explained in more depth due to confidentiality issues.

What can be commented, and what can be found in the official repository of this project, is that the implementation of the pitch extraction algorithm is done in C++, due to certain limitations that Python offered for its creation. Once created in C++, a Python wrapper was used that allows using C++ code from Python. All these files are inside the extractors folder of the project. At this reference more information about python wrappers can be found. [41]

3.4 Note Detection Algorithm

As explained in the previous sections, the Note Detection Algorithm is the most important element of the whole process. As its name indicates, this algorithm is in charge of detecting musical notes in the audio file, in order to classify the sung voice according to the percentage of notes that are detected.

Compared to state-of-the-art note detection algorithms, the present one is simpler. This is because annotated data was not available and therefore a method requiring minimal supervision was required.

However, the creation of the Note Detection Algorithm was the most complex part of the whole algorithm, due to the recurrence required to run through all the detected speech segments multiple times. This performance will be explained in more detail in the next section.

The note detection algorithm is an element that not only serves to discern between sung and spoken voice, but also for more applications in the processing of sung audio, more specifically in audio synthesis, due to its ability to discreetly transcribe exact musical notes [42].

3.4.1 Transformation of F0 curve to F0 cent curve

The first task in the implementation of the Note Detection Algorithm is to map the pitch curve into cent scale. The cent scale is a logarithmic unit of measure for musical intervals [43]. Equal temperament is the most used system today for tuning musical instruments.

The musical scale, composed of 12 notes that are repeated in octaves, is divided into 12 equal parts called semitones. Each semitone corresponds to 100 cents in the cent scale. It is a logarithmic scale because the relationship between the sounds of the musical scale (in Hz) is not linear.

The cent scale is mostly used for the comparison of different tuning systems. In this work it is used to provide a simpler and more accurate analysis, since the semitones are easily located every 100 cents, and this allows to position the musical notes.

For the transformation of the f0 curve to the cent scale, we simply apply the 3.2 formula with an offset so that all possible values we receive are positive. The reference frequency is 440Hz, which corresponds to the note A4 in the musical scale. It is widely used as a reference in the music field. The range of cents of the human voice goes, more or less, from 2950 cents to 4900. It corresponds, in Hz scale, to the range (85,255), which are the most common f0 in human voice [44].

$$f_{0c} = 1200 * \log_2(\frac{f_0}{f_r ef}) + 5800 \tag{3.2}$$

3.4.2 Obtention of musical notes

The scoring process itself is the implementation of the algorithm once the cent-scale curve is calculated.

It is a cyclic process that is performed on the segments catalogued by the VAD as voiced.

For the analysis, sub-sequence search techniques are used [45], that search for groups of substrings that meet two predetermined conditions to be considered as musical notes: minimum duration of 150ms and maximum variation of 100 cents. These limits have been proposed as the standard for Western music [7].

The algorithm is defined in the next steps:

1. First of all, the whole pitch curve in cent scale must be considered as a collection of K voiced segments, each one with its own length.

2. Later, a search of the longest sub-sequence that fills both conditions above mentioned must be done.

3. When the longest segment is found, it gets labeled as a musical note.

4. Later, the segment must be divided to the left and to the right of the sub-sequence catalogued as a note, and the process must be repeated for each of the parts.

5. This process is performed twice, i.e., on the left and right side of the longest note found in the segment, the process is performed again. In each new segment, the longest note is searched again and both segments split again and they reproduce again the process. So, the original process is performed twice, because performing it until no more notes were found was too difficult to implement.

6. Once the analysis is complete, it moves on to the next voiced segment until all segments are analyzed and the final result of the algorithm can be returned.

What can be found in the figure 3.9 is the graph that defines the Note Detection Algorithm.

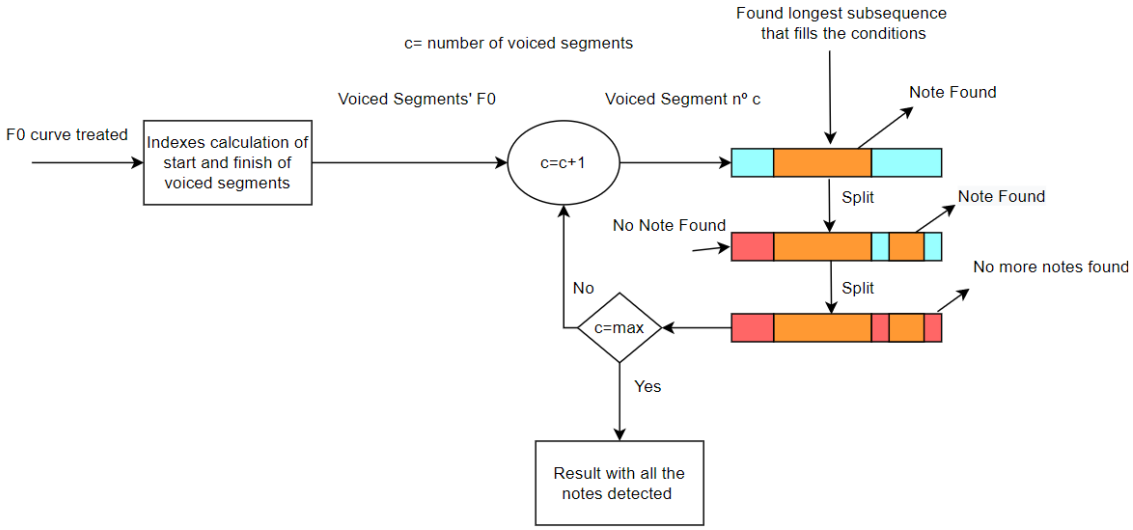

Figure 3.9: Note Detection graph

As a conclusion, it must be said that a concrete note detection analysis could be performed, because of the transcription of the notes detected from the cent scale to the Hz scale. Nevertheless, it has not been performed because the usability of this algorithm in the implementation of the whole project is not to know the concrete value of the musical notes, but the percentage of musical notes found in the 200ms window mentioned above.

What can be found in figure 3.10 is the performance of the Note Detection Algorithm over an entire sung audio. As it can be seen, when there is no audio, as in the first 350 samples, no notes are detected. After that, when the singer holds a musical note, different notes start being detected, and with this information, the parameters that come after this section are calculated.

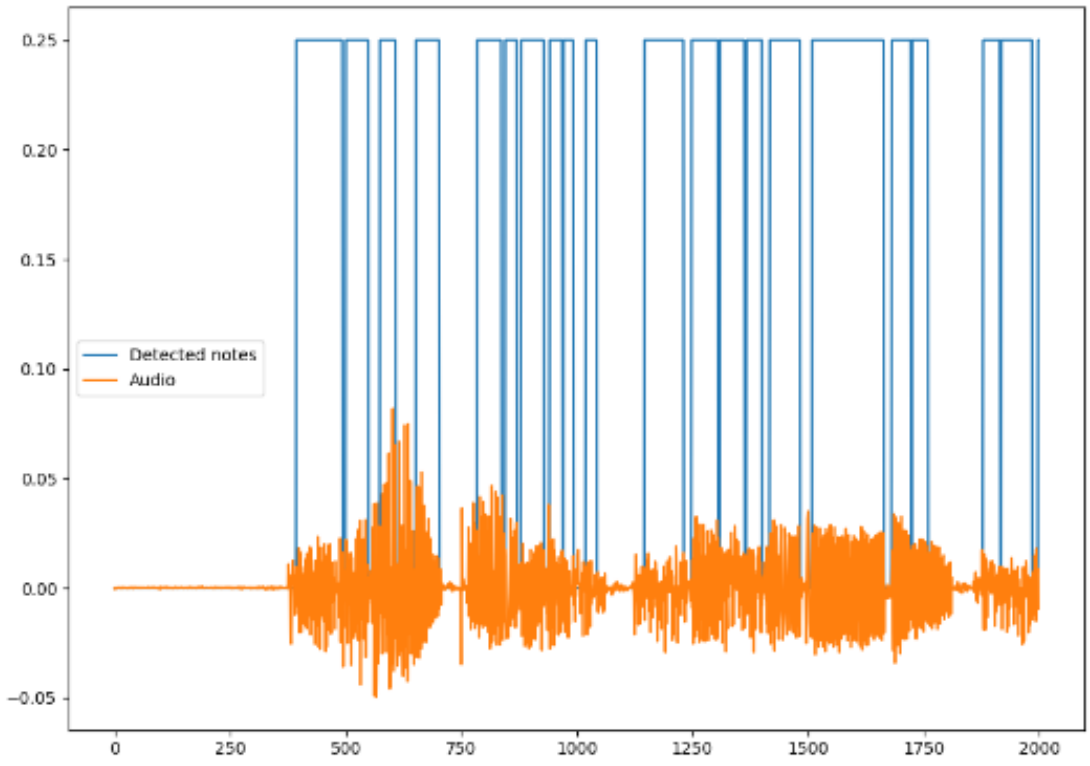

Figure 3.10: Note Detection

3.5 Calculation of PV and PN parameters

As it has been discussed throughout this paper, the classifier is based on 2 parameters derived from the classification of musical notes and the calculation of the fundamental frequency. Both parameters are derived from the pitch of the audio file: the f0 calculation and the note detection (which is based on the search of f0 stable values).

Proportion of voiced segments (PV) is the parameter that calculates the number of spoken segments compared to the total number of segments.

Percentage of frames labeled as a musical note (PN) is the parameter that calculates the number of frames labeled a musical note compared to the total number of segments.

The value of both parameters was calculated by scrolling a window around the entire audio file. For this reason, there is a value for each of the parameters for each windowed portion of the audio. In the case of 200ms window, each parameter has a value for each 20 frames.

When applying a windowing to the audio, the size of the window to be applied had to be chosen. As stated in the Objectives section of this paper, the ultimate goal of this work is the creation of a novel method of segmentation between sung and spoken speech in real time. That is why the window study is performed at low values, because the real time application requires the smallest possible window value. Both 200ms and 500ms window values have been studied and their metrics have been calculated.

It can be concluded that the 500ms value for the datasets used in this work has a slightly better performance, but nothing particularly remarkable in comparison to smaller values down to 200ms. That is the reason why, for both the offline model and the real time model, it has been used a 200ms window, since the loss of accuracy or f-score was not appreciable compared to the added delay that working with 500ms windows would entail in comparison with the 200ms window.

However, in the Python project there is a variable window parameter that can be changed to the user's preference. It is presupposed for the calculation of the parameters, as the Reference Paper concludes, that each value represented belongs to only one class: silence, speech or sung voice.

3.6 SVM based Decision Function

The last element of the implemented algorithm is the final decision function based on the parameters obtained in the previous part.

For the creation of this function, an Support Vector Machine has been implemented. SVM is a classical machine learning technique that helps in the work classification problems with big amounts of data. Although it was originally developed as a binary classification method, its application has been extended to multiple classification and regression problems. SVM has proved to be one of the best classifiers for a wide range of situations, and is therefore considered one of the benchmarks in the field of statistical learning and machine learning [17].

The in-depth understanding of the SVM technique is complex, due to the requirement of a solid knowledge of linear algebra. However, in this section a brief explanation is presented in order to understand the nature of this algorithm and how it has been applied in this work.

3.6.1 Theoretical explanation of SVM

The Support Vector Machine algorithm is based on two concepts: Hyperplane and Maximal Margin Classifier. All the information regarding this section can be found in this report[46].

In a p-dimensional space, a **hyperplane** is defined as a flat affine subspace of p-1 dimensions.The affine term means that the subspace does not have to pass through the origin. For example, in a two-dimensional space, the hyperplane is a 1-dimensional subspace, i.e., a line.

$$\beta + \beta_1 * x_1 + \beta_2 * x_2 = 0 \tag{3.3}$$

When n observations are available, each with p predictors and whose response variable has two levels, hyperplanes can be used to construct a classifier that allows predicting to which group an observation belongs based on its predictors. This is what happens in the

decision function, the predictors being the parameters calculated for the training dataset.

The definition of hyperplane for perfectly linearly separable cases results in an infinite number of possible hyperplanes, which makes necessary a method to select one of them as the optimal classifier. In the figure 3.11, different possible hyperplanes are plotted.

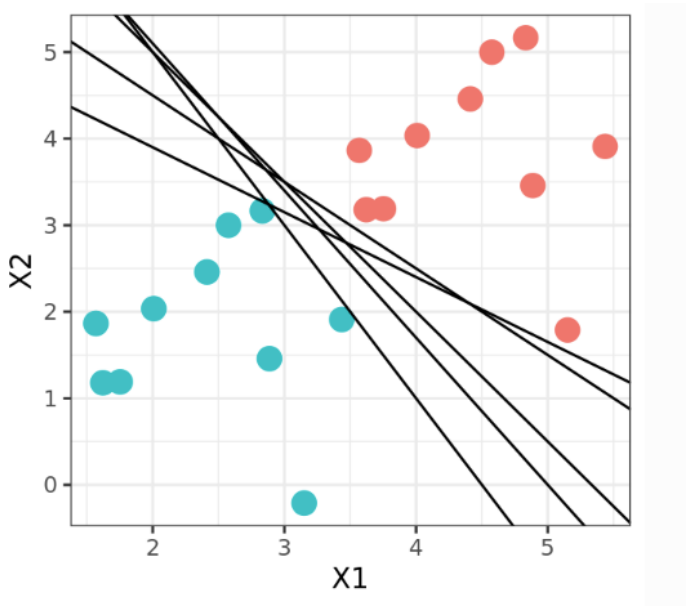

Figure 3.11: Different hyperplane possibilities (extracted from [46])

The optimal classifier is known as **Maximal Margin Hyperplane**, which corresponds to the hyperplane that is farthest away from all training observations. To obtain it, the perpendicular distance of each observation to a given hyperplane must be calculated. The maximal margin hyperplane is defined as the hyperplane that achieves the largest margin, i.e. the minimum distance between the hyperplane and the observations is as large as possible.

Moreover, the Maximal Margin Classifier described above does not obtain the best performance in practice, since cases where classes are perfect and linearly separable are rarely encountered.

For this reason, it is preferable to create a classifier based on a hyperplane that, although it does not perfectly separate the two classes, is more robust and has better predictive capability when applying new observations.

This is what Support Vector Classifiers achieve. To achieve this, instead of looking for the widest possible classification margin that gets the observations on the correct side of the margin; certain observations are allowed to be on the wrong side of the margin or even the hyperplane.

For the resolution of this problem, which is substantially more complex than the previous one presented, convex optimization is used, which will not be explained in this paper due to its difficulty.

When the cases are not perfectly linearly separated, another strategy has to be found in order to achieve an optimal classification. One strategy for dealing with scenarios in which the separation of groups is nonlinear is to expand the dimensions of the original space.

The fact that the groups are not linearly separable in the original space does not mean that they are not linearly separable in a higher dimensional space. In figure 3.12 an example of expansion of dimension is plotted.

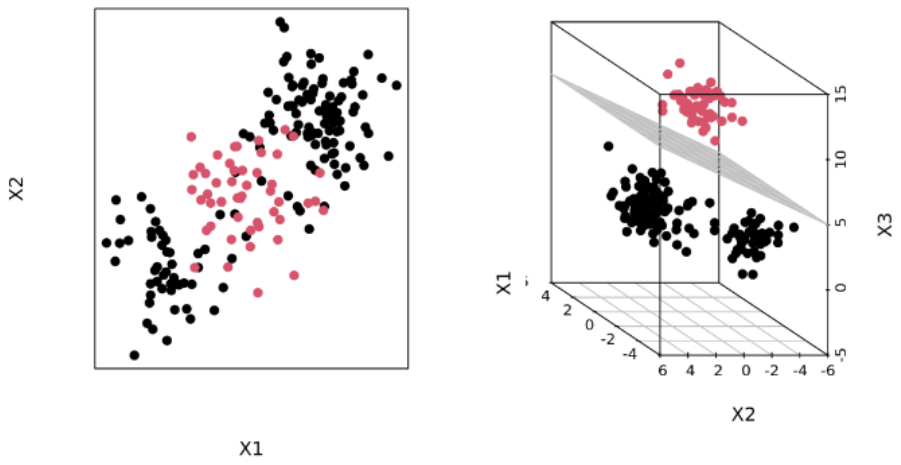

Figure 3.12: Expansion of dimension of the original space (extracted from [46])

This is where kernels come into play to know how the dimension is increased and which dimension is correct. This is often referred to as the ”kernel trick”, because, with only a slight modification of the original problem, thanks to kernels, the result can be obtained for any dimension. There are many different kernels, some of the most used are the linear kernel, polynomial kernel or gaussian kernel.

In figure 3.13, the three types of kernels are plotted and they show different ways of doing classification, depending on the distribution of data.

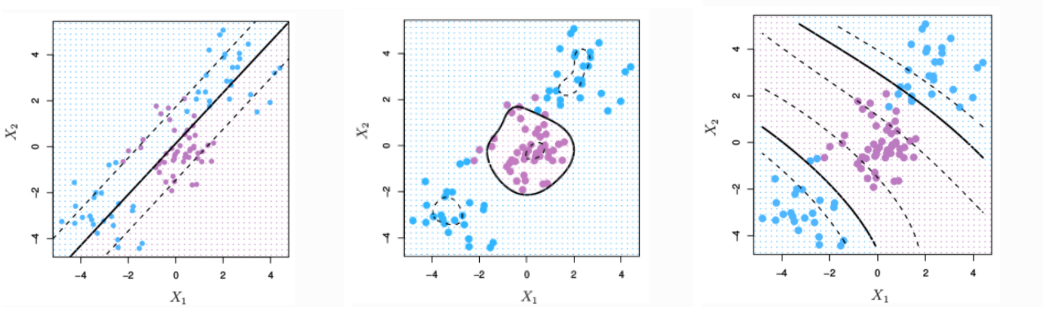

Figure 3.13: Comparison between linear, gaussian and polynomial kernels (extracted from [46])

3.6.2 Implementation of SVM with SCIKIT-LEARN

In this work, the SVM has been implemented thanks to the SKLEARN python library [47].

This library implements the C-Support Vector Classification, which is the Support Vector Machine technique explained in the last part of the previous section.

The syntax and implementation of SVM with this library are very simple. In this case, a method was developed that trained the SVM with the training dataset that will be explained later. The input parameters to the SVM were the two parameters of PN and PV of the training dataset. Only parameters other than 0 were used for training, since only the segments classified as speech pass through the SVM, as those classified as silence are directly classified without having to go through this classifier.

The different types of kernel were tested, and finally the one that gave the best performance, based on the training data, was the linear kernel.

The training configuration was saved in a file named ”finalize model.sav”, where the model trained with the fit method (method used to embed the training data to the network and train it) was saved.

Once the model has been trained, all that remains is to test it with the testing dataset that will be presented later. With this test, the metrics that were used to select the best possible kernel were extracted. In order to predict whether a frame is spoken or sung, all that remains is to pass to the SVM its two previously calculated parameters and the classifier will give an answer.

The parameters used for the prediction are calculated at the window level, so the classification is performed at that level as well. That is why, once the decision has been made, all the frames of that window are available and labeled according to the VAD decision and the SVM decision. This idea will be extensively explained in the main function implementation section.

3.7 Implementation in the main function of the Algorithm

The whole algorithm is collected in the main function, main.py, from reading the audio to the final decision making.

The function itself is simple, as the philosophy behind building the code for this algorithm has been to create methods in the scripts for each element of the algorithm.

Therefore, the dynamics of the main function is:

1. Reading the audio input to the algorithm using librosa [15]. The default sample rate of 48kHz is selected.

2. Import of both files containing the parameters of the previously trained VAD. These files will be used to pass the audio through the voice detection algorithm.

3. Calculation of the MFCC and with them, calculation of the binary vector that classifies the frames in silence or voice.

4. Calculation of the fundamental frequency of all the frames, independently of its value in the VAD binary vector.

5. Treatment of the f0 function, disregarding the values that are classified as silence by the VAD

6. Calculation of the Note Detection Algorithm with the f0 curve already treated.
7. Selection of the window size and calculation of both parameters: PV and PN.
8. Selection of the f0 that are cataloged as voice and insertion of them in the SVM for decision making at window level.
9. Once the decision vector is received, the decision at window level is transformed to frame level. The dynamics is as follows: if the VAD considers that the entire window is silent, the decision vector is filled with all silences. If the VAD considers speech and the SVM considers speech, then the frames cataloged as silence within that window are set to zero and the frames cataloged as speech are set to one. If the VAD considers speech and the SVM considers sung voice, then the frames cataloged as silence within that window are set to zero and those cataloged as speech are set to two. In this way it is possible to transform a decision at the window level to the frame level, taking into account the action of the Voice Activity Detector.

The final decision is housed in a vector which is the return of the main function, where 0 represents silence, 1 represents speech and 2 represents sung voice.

3.8 Graphic User Interface

The main function of the algorithm returns a vector representing the different classes with values 0 (silence), 1 (spoken speech) and 2 (sung voice). A graphical interface has been created to make the algorithm more manageable for the user.

The interface has been implemented using the python package Tkinter [48]. This package is the default Python interface to the tk GUI toolkit.

The graphical interface of the offline model has a logical path, forcing the user to it. Otherwise, errors will show and the algorithm will not allow the user to progress.

First, the audio must be selected from any local folder on the machine from which the algorithm is run. This step is achieved by pressing the ”Select an audio file” button. If any other button is pressed beforehand, an error window will pop up telling the user that the first thing to do is to select the audio.

When the audio is selected, the path is saved. On the other hand, there are 2 more buttons: ”Start Analyzing” and ”See chart”. As their names indicate, pressing the first one will call the main function of the algorithm with the path as parameter.

Once the algorithm has obtained the solution, pressing the second button produces a graph where the classification result and the audio itself are plotted, so that a comparison of the audio signal and the final decision can be made.

This interface allows the user to use the algorithm in a much simpler and more intuitive way, and to display results in a clear and concise manner. In addition, the decision vector itself is also produced with its respective values, in case the user would like to use it for any other application.

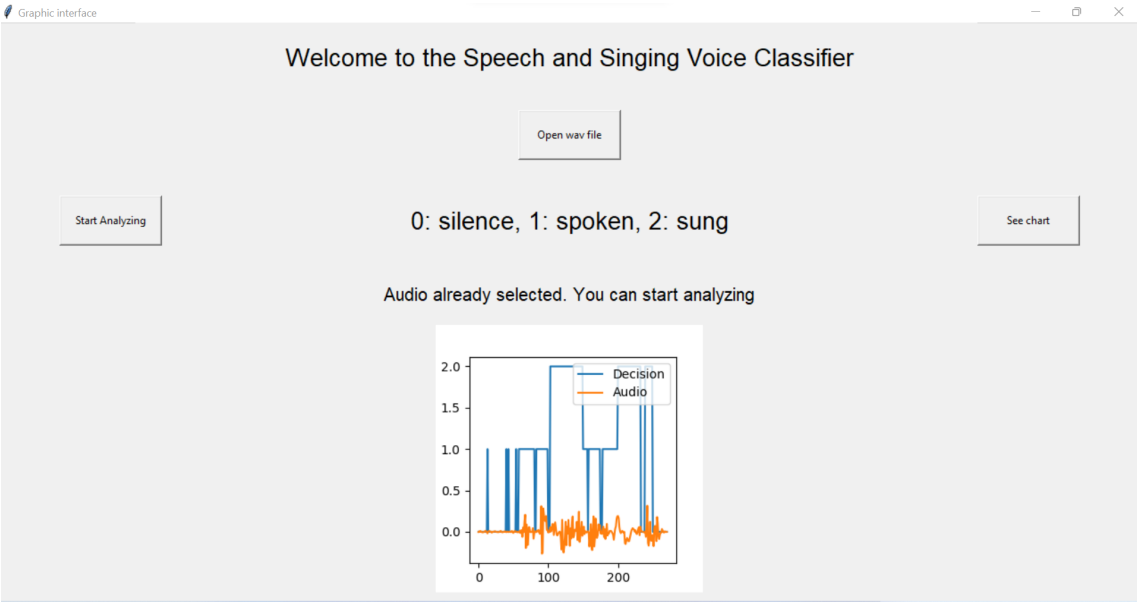

Figure 3.14: Graphic interface of the offline model

3.9 Real Time Implementation

After all, the main reason why this work was carried out is to achieve a novel method of classification between sung voice and spoken voice in real time. This algorithm, framed in the context of internship in VOICEMOD, is useful in its application in real time, since the work performed implements filters on the voice in real time.

In order to achieve a fast and less complex implementation of the model in real time, the approach of the algorithm in offline mode was from the beginning at frame level, so that, when it would be possible to implement it in real time, it would be much simpler and faster. The work of all the sub-algorithms of the offline model is done at the frame level.

For the implementation, the first step was the creation of an audio stream. For this, the Python sounddevice library was used [49]. The audio stream can be interpreted as a cable between the microphone used to capture the audio and the Python code that executes the algorithm. The creation of this module is vital for the algorithm to be able to save the audio for analysis.

As discussed in previous sections, in the offline model, a study of the best windows was carried out to go through all the audio and analyze it. It was concluded that the metrics between 0.2s and 0.5s windows were practically identical, which was good news because in this way it is likely that a real-time model with a 200ms delay can be implemented.

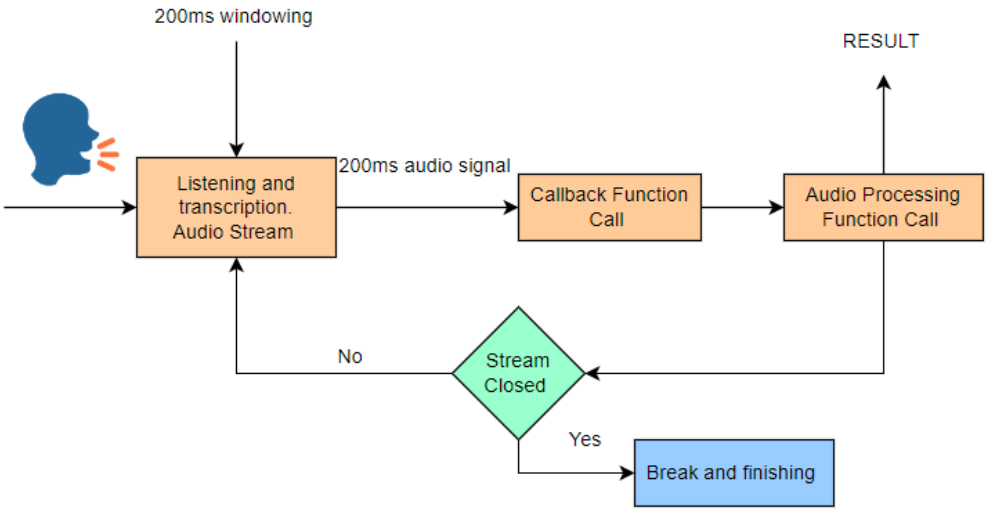

Figure 3.15: Real time implementation

Therefore, by choosing the 200ms window, the audio stream picks up the audio signal constantly, and 200ms fragments (20 frames) are passed through the audio processing function which returns a window-level solution of 200ms, which is an appreciable and reasonable amount for the user to distinguish the changes without difficulty.

The delay is 200ms, because the audio processing algorithm takes less than 200ms, so there is no need to implement a secondary buffer to store audio fragments. This is because when the algorithm is going to analyze a new 200ms fragment, the previous one will have already finished and the decision will have already been issued, so it will not need to wait.

For a quick and concise implementation of the algorithm in real time, most of the original algorithm files have been modified. Among the most modified is the note detection algorithm, which has been especially simplified.

This is because one of the conditions that the note detection algorithm must meet is that the note must have a duration equal to or greater than 150ms. For this reason, in each 200ms window only one musical note can be labeled. Therefore, the recurrence previously required in the offline model by the algorithm is no longer necessary, and therefore the complexity of note detection decreases drastically.

Other files such as the VAD, the SVM, the parameter calculation files or the main itself have been slightly modified, but nothing very remarkable to be commented on.

In the official repository of the project, the distinction between the offline algorithm and the real-time algorithm is clear. A folder called ”real time implementation” houses the entire implementation and can be executed in offline mode.

In the figure 3.15, the flow chart of this algorihtm can be found, in where the 200ms window real time is implemented. The result is always given unless the user stops manually the process.

3.9.1 Real time graphic interface

As in the offline version of the algorithm, for the real time version a graphical interface has been built to allow the user a clear and easy to distinguish user interaction.

In this case, it is even a slightly simpler version than the offline version. Since you do not have to load any audio from the workspace where the algorithm is running, just click on the ”Start Analyzing” button and the algorithm will start running.

The algorithm discussed in the previous section will start running, displaying a label in the center of the screen that will distinguish between silence, speech or sung voice. In this way, the user can easily see the algorithm in action and its result in real time. In the figures 3.16 and 3.17 is shown a comparison between the three possibilities of the algorithm: silence, talking and singing.

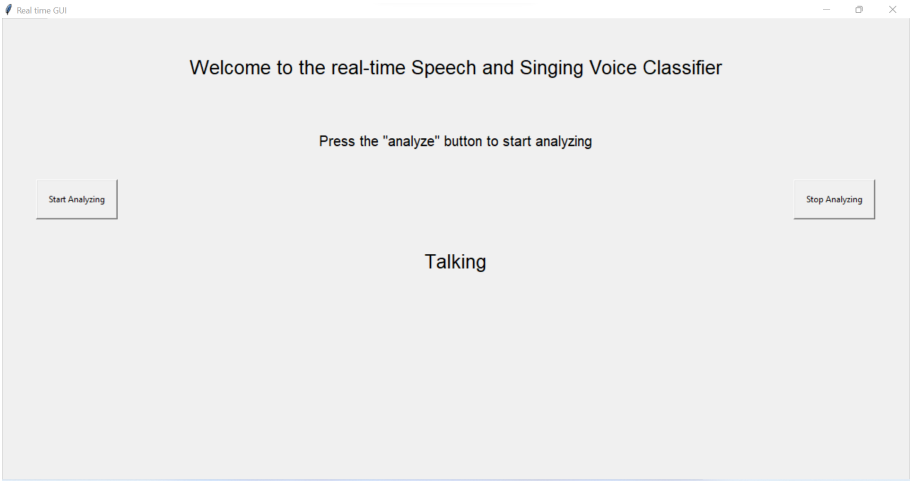

Figure 3.16: Real time Graphic Interface when talking

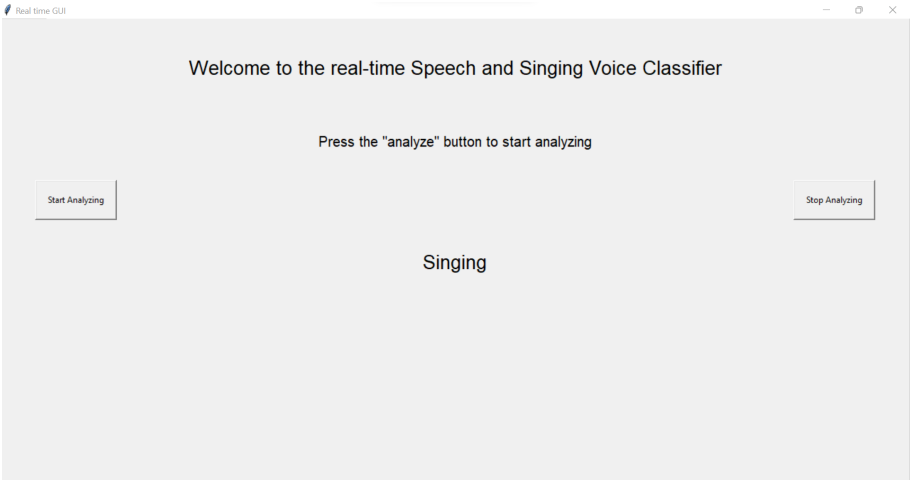

Figure 3.17: Real time Graphic Interface when singing

3.9.2 Computational Cost of the algorithm

The computational cost is going to be commented in terms of delay, how long does it take to the algorithm to complete its task.

An empirical study has been performed. The offline model, in terms of delay, lasts in perform its work a bit less than the duration of the audio file being analyzed. This means that, for example, for a 10 seconds audio file, the algorithm takes more or less 9 seconds in being run.

This is good news for the real time implementation, because it lasts less than the window size. This means that, when the audio starts being analyzed, the audio before has already finished and there is no delay. If this weren't like that, and the performance lasted more than the audio duration, a ring buffer must have been implemented.

This computational cost, is almost the best possible with these parameters, because the Note Detection Algorithm does not allow smaller delay. There are not references related to this topic, because the whole study was done empirically by analyzing the duration and how long it took to perform the algorithm in the testing dataset.

Chapter 4

Experiments and results

In this section of the paper we will present the datasets used for training and testing as well as the results obtained, by means of deterministic plots and presentation of metrics of the most important parts of the algorithm.

To begin with, both the two datasets and their labeling process will be discussed in detail. Later, the metrics and graphs of the behavior of the Voice Activity Detector and the SVM decision function will be presented.

4.1 Datasets

Two databases have been used for the training and testing of the algorithm in this paper. These datasets have been chosen because they are the ones used in the Reference Paper.

Together with the real time implementation, the main objective of this algorithm was to replicate as closely as possible the Reference Paper. In this way, the comparison of metrics and graphs would be as fair as possible. That is why the same dataset that was used in the Reference Paper implementation has been used.

The **labeling** of the dataset is the creation of the labels that identify each of the frames with a certain class to be classified. There are several possible labelings of a dataset, depending on the characteristic of the dataset to be analyzed.

In many cases, the labeling is done by the author of the dataset. This is not the case in both datasets to be used in this document.

That is why, due to the imperative need of labeling for algorithm training and metrics calculation, the labeling of both datasets was elaborated by hand by the author of this project.

It was a hard work, because each 10ms frame of the entire dataset had to be labeled by hand. To do this, the audio was observed frame by frame using Audacity [50] while filling in the label according to the author's own criteria.The intention of this labeling is its official publication as labeling generated by this project.

4.1.1 Training Dataset

For the training of the algorithm we have used the Bertsolaritza Dataset. Bertsolaritza is a live improvisation poetry art from Basque Country. In these live sessions a host introduces the singer and proposes the topic for the improvised verses that will be sung a capella. The singers are professional verse creators, but not professional singers. Many live sessions of Bertsolaritza are recorded and saved together with the corresponding transcriptions by Bertso associations.

The Bertsolaritza dataset found in the website `https://bdb.bertsozale.eus/en/web/bertsoa/bilaketa` is particularly extensive. Therefore, as is also done in the Reference Paper, a selection of audios of different characteristics was produced. These audios come from both women and men, from different ages and different lengths. It is composed of 27.48 minutes of audio. The 86.2% of the audios are sung voiced and the 13.8% are spoken voiced. It is an unbalanced dataset, but it has been treated like that and the training has been done taking that into account.

4.1.2 Testing Dataset

The test dataset is called "Nus Sung and Spoken Lyrics Corpus". It is explained and detailed in the following paper [51].

It consists of a 169-min collection of audio recordings of the sung and spoken lyrics of 48 (20 unique) English songs by 12 subjects and a complete set of transcriptions and duration annotations at the phone level for all recordings of sung lyrics, comprising 25,474 phone instances.

This dataset has a previous labeling with respect to the phonetics of the words mentioned by the selected members. However, as it was commented above, a labeling with respect to a clear differentiation between sung, spoken and silent at frame level was not done, so it had to be done manually.

As was also the case with the training dataset, a selection of the test dataset has been made for this work. This is mainly due to two reasons. On the one hand, the lack of time to label 169 minutes of dataset frame by frame. On the other hand, the Reference Paper has also used an extract of this dataset, so the exact reproduction of this paper requires a selection of a part of it.

The duration of the used dataset is 37.19 minutes. The 72.8% of the audios are sung voiced audios and the 27.2% of the audios are spoken voiced audios. As the training dataset, it is unbalanced but it has been treated like that and the testing has been done taking that into account.

4.2 Voice Activity Detection Metrics

The voice detection task is a very important section in the implementation of this algorithm. That is why it was necessary to create a good VAD, with metrics up to the level of the Reference Paper and capable of discerning between speech and silence in both the offline and real-time model.

In the results section of the Reference Paper, special emphasis has been placed on the metrics and tables shown by the VAD and by the general algorithm, and that is what will be shown in this and the following section. The metrics obtained by the VAD are very good. For the study of its performance, four different metrics have been calculated that give a broad view of how the algorithm behaves.

- Accuracy. It is the fraction of predictions that the model made correctly. It is represented as a percentage or a value between 0 and 1. It is a good metric when we have a balanced data set, that is, when the number of labels of each class is similar. In this case, our datasets are not balanced but in the accuracy calculation, a parameter was used in order to take into account that imbalance.

- Recall. It indicates the proportion of positive examples that are correctly identified by the model among all real positives. That is, TP / (TP + FN). In our example, the sensitivity value would be 1 / (1 + 0) = 1. TP (True Positives) is when the class of analysis is 1 (in this case, voice) and it predicts it correctly. In the other case, False Negatives (FN) is when a 0 is given as the prediction but the answer should be 1.

- Precision. This metric is determined by the fraction of items correctly classified as positive among all those that the model has classified as positive. The formula is TP / (VP + FP). The example model would have an accuracy of 1 / (1 + 1) = 0.5. FP (False Positives) is when a 1 is given as the prediction but the answer should be 0.

- F-score. It combines the Precision and Recall metrics to give a single result. This metric is most appropriate when we have unbalanced data sets. It is calculated as the harmonic mean of Precision and Recall. The formula is F1 = (2 * precision * recall) / (precision + recall). This metric is very important in the analysis not only of the VAD but of the whole algorithm, because it itself takes into account that the dataset is unbalanced, as the datasets used in this document are.

This model and algorithm analysis metrics are widely used in the field of Artificial Intelligence models [52]. In the case of this work, they have all been implemented using the Python sklearn library [53]. In the figure 4.1, TP, FP, FN and TN are defined in relation to the real values and the predicted values.

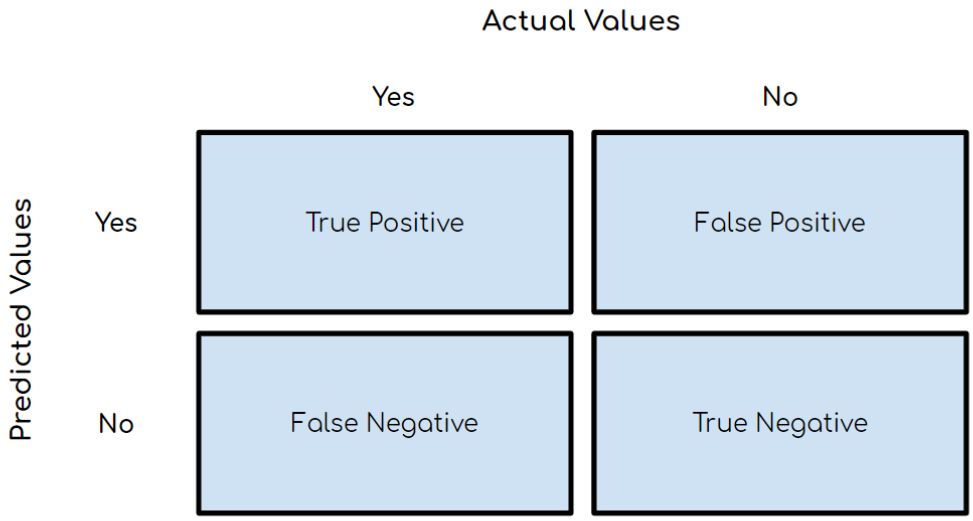

Figure 4.1: Confusion Matrix of a model (extracted from [54])

Table 4.1 shows the metrics obtained for the case of the VAD trained and tested with the aforementioned datasets. Values above 0.9 can be considered a very good metric for every model, and here at this GMM based VAD, those metrics are achieved.

Metrics	Values
Accuracy	0.899
Recall	0.909
Precision	0.956
F-score	0.932

Tabla 4.1: VAD Metrics table

4.3 Results of speech/singing classification

Once the VAD metrics, which stand out for their good values, have been presented, we now present the tables and metrics obtained from the algorithm trained and tested with the two datasets discussed in the previous section.

Classification in VAD is binary, i.e. the algorithm must predict a sample between two possible values. This is not the case in the final algorithm, where one must choose between three possible classes: silence, speech and sung voice. This is why the metrics are calculated for these three classes, taking into account that the TP, FP, TN and FN values are now calculated taking into account the hits and misses in the three classes.

For the case of the implementation of these metrics, and according to the documentation of sklearn.metrics proposes [53], there is a parameter called "average" that allows calculating the metrics for all classes and then makes a weighted average depending on the weight of each class in the dataset that has been used.

The Reference Paper, for the analysis of the training and testing of the SVM, and thus the output of the algorithm, draws two graphs showing the distribution of both parameters on which the decision is based (PN and PV) on the x/y axis and colors the labeling.

In this way, it shows how the end of the algorithm that makes the final decision of the voiced segments has been trained and tested. Figure 4.2 shows the two plots obtained from this algorithm, showing a clear distribution that allows to discern between the two classes.

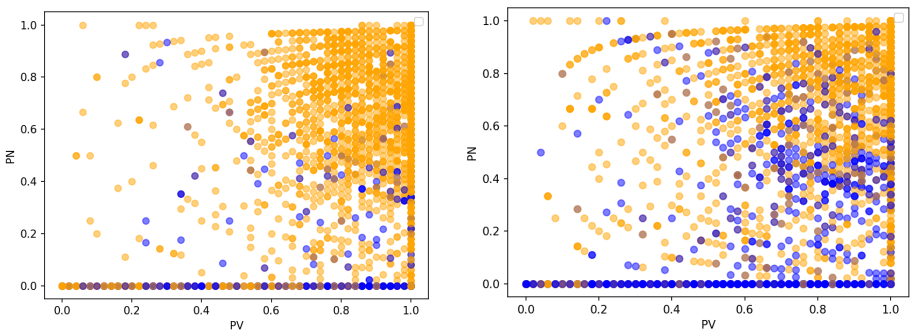

Figure 4.2: Distribution of both parameters in both datasets

The metrics obtained in the algorithm implemented in this paper are shown below. The objective, from the beginning, was the comparison of results and metrics with the Reference Paper. For this purpose, the algorithm created by the authors of the Paper framed in the aforementioned Doctoral Thesis was replicated as best as possible, without having access to the original code.

This is why, after implementation, the results obtained are shown in table 4.2. The metrics are almost near 80%, and the possible improvements to the algorithm to improve these metrics will be shown in the next chapter.

Metrics	Values
Accuracy	0.772
Recall	0.777
Precision	0.781
F-score	0.771

Tabla 4.2: Whole algorithm Metrics table

Chapter 5

Conclusions and further works

As the last section of this paper, a conclusion of the implemented project and possible future work that could be done on the algorithm will be discussed.

The objective of this project, and which has certainly been achieved, has been the replication and implementation of the Speaker Style Classifier Reference Paper algorithm between sung and spoken voice, both offline and in real time. The successful implementation of the algorithm in real time means that this project has been able to achieve the creation of a novel method of classification of sung and spoken voice in real time.

As mentioned above, this project has been framed in the internship, by the author of this paper, in the company VOICEMOD S.L. The possible applications that this algorithm could have industrially were, from the beginning, the objective of both the author and the company. In this case, the objective has been largely fulfilled, since the algorithm is ready to be implemented in C++ in real time, and later to be introduced in the core of the VOICEMOD application.

The implemented algorithm is a consequence, a chain of sub-algorithms capable of performing tasks on their own. In this section of conclusions, it is of object to highlight the creation of the Note Detection Algorithm, for what it entails to have created it for future applications, as mentioned in the corresponding section.

The implementation of all these sub-algorithms has undoubtedly generated many problems during its creation. These complications, which in some cases have even been set aside due to their complexity, are part of the possible future work that can be done on this project.

To be able to generate superior metrics throughout the implemented algorithm, the separation into classes exposed in Figure 4.2 should be more evident. In this way, the decision function would be better trained and tested, and therefore the results would be better.

These improvements mainly involve improving the Note Detection Algorithm. It detects notes by traversing all the segments that are considered to be spoken, and performing a recurrent process around them.

This recurrent and ”infinite” process until no new musical notes appear is very difficult to implement. In this algorithm, a repetition process has been implemented twice, not being ”infinite”. Therefore, one of the further works that could be done would be the improvement of this feature of the Note Detection Algorithm. This improvement would help both offline and real time models to improve their metrics. Another possible future work is the improvement of the delay in the real time algorithm. This is not an easy task, because if we wanted to go below 150ms of delay, we would have to rethink the note detection algorithm, since it considers that a musical note can be classified as such if it lasts more than 150ms. This aspect deserves a separate study, since it would be necessary to be able to demonstrate a maintenance or improvement of the metrics by changing the conditions to be met in the Note Detection Algorithm.

This algorithm, as discussed throughout the document, is based on this Reference Paper [7]. A possible future work would be the analysis of the parameters on which the algorithm is based. There are many parameters extractable from pitch and other discourse analysis techniques that could be taken into account to make the final decision. Vibrato analysis, phonetic analysis, duration of segments considered as speech, among others. An in-depth study of these aspects would be worthwhile in order to improve the metrics obtained in this paper.

Therefore, it can be considered, as a final conclusion of this paper, that the ”Speech and singing voice classifier based on musical note classification and fundamental frequency calculation” has been successfully implemented.

Bibliography

[1] Sarah McRoberts et al. “Exploring Interactions with Voice-Controlled TV”. In: *CoRR* abs/1905.05851 (2019). arXiv: 1905.05851. URL: http://arxiv.org/abs/1905.05851.

[2] Google. *https://developers.google.com/learn/topics/chatbots*. Last access: 1st July 2022.

[3] Google. *https://blog.google/products/translate/transcribe-speech/*. Last access: 1st July 2022.

[4] OpenAI. *https://openai.com/blog/openai-api/*. Last access: 1st July 2022.

[5] Antonio José Aragón Molina. https://idus.us.es/bitstream/handle/11441/125219/TFG-3474-ARAGON%20MOLINA.pdf?sequence=1&isAllowed=y. Seville, 2021.

[6] Ascensión Gallardo y Juan Manuel Montero. http://www-gth.die.upm.es/research/documentation/AN-099Tec-19.pdf. Last access: 1st July 2022.

[7] Xabier Sarasola et al. “Speech and monophonic singing segmentation using pitch parameters”. In: *Proc. IberSPEECH 2018*. 2018, pp. 147–151. DOI: 10.21437/IberSPEECH.2018-31. URL: http://dx.doi.org/10.21437/IberSPEECH.2018-31.

[8] Xabier Sarasola et al. “Application of Pitch Derived Parameters to Speech and Monophonic Singing Classification”. In: *Applied Sciences* 9.15 (2019). ISSN: 2076-3417. DOI: 10.3390/app9153140. URL: https://www.mdpi.com/2076-3417/9/15/3140.

[9] Xabier Sarasola Aramendia. “http://hdl.handle.net/10810/50503”. In: *UPV/EHU* (Last access: 1st July 2022).

[10] Voicemod. july 2022. URL: https://www.voicemod.net/.

[11] Python. Last access: 1st July 2022. URL: https://techvidvan.com/tutorials/python-advantages-and-disadvantages/.

[12] Python. Last access: 1st July 2022. URL: https://numpy.org/.

[13] Python. Last access: 1st July 2022. URL: https://scikit-learn.org/stable/.

[14] Python. Last access: 1st July 2022. URL: https://scipy.org/.

[15] Python. Last access: 1st July 2022. URL: https://librosa.org/doc/main/generated/librosa.feature.mfcc.html.

[16] Guorong Xuan, Wei Zhang, and Peiqi Chai. "EM algorithms of Gaussian mixture model and hidden Markov model". In: *Proceedings 2001 International Conference on Image Processing (Cat. No.01CH37205)*. Vol. 1. 2001, 145–148 vol.1. DOI: 10.1109/ICIP.2001.958974.

[17] William S.Noble. "https://www.nature.com/articles/nbt1206-1565". In: *nature biotechnology* (1December 2006).

[18] Python. Last access: 1st July 2022. URL: https://pypi.org/project/pyworld/.

[19] Toan Pham Van, Ngoc N. Tran, and Ta Minh Thanh. "Deep Learning Approach for Singer Voice Classification of Vietnamese Popular Music". In: *CoRR* abs/2102.12111 (2021). arXiv: 2102.12111. URL: https://arxiv.org/abs/2102.12111.

[20] Ramy Monir, Daniel Kostrzewa, and Dariusz Mrozek. "Singing Voice Detection: A Survey". In: *Entropy* 24.1 (2022). ISSN: 1099-4300. DOI: 10.3390/e24010114. URL: https://www.mdpi.com/1099-4300/24/1/114.

[21] Beigi Homayoon. "Fundamentals of Speaker Recognition". In: (2011). DOI: https://doi.org/10.1007/978-0-387-77592-0.

[22] Tom Bäkcström. Last access: 1st July 2022. URL: https://wiki.aalto.fi/pages/viewpage.action?pageId=149890776.

[23] A.L. Berenzweig and D.P.W. Ellis. "Locating singing voice segments within music signals". In: *Proceedings of the 2001 IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics (Cat. No.01TH8575)*. 2001, pp. 119–122. DOI: 10.1109/ASPAA.2001.969557.

[24] Md. Afzal Hossan, Sheeraz Memon, and Mark A Gregory. "A novel approach for MFCC feature extraction". In: *2010 4th International Conference on Signal Processing and Communication Systems*. 2010, pp. 1–5. DOI: 10.1109/ICSPCS.2010.5709752.

[25] IBM Cloud Education. https://www.ibm.com/cloud/learn/neural-networks. Last visit: july 2022.

[26] Theodoros Giannakopoulos. In: *Science Direct* (Last access: 8th July 2022). URL: https://www.sciencedirect.com/topics/engineering/zero-crossing-rate.

[27] Yann Teytaut and Axel Roebel. "Phoneme-to-Audio Alignment with Recurrent Neural Networks for Speaking and Singing Voice". In: *Proc. Interspeech 2021*. 2021, pp. 61–65. DOI: 10.21437/Interspeech.2021-1676.

[28] Patricia Alonso de Apellániz. *https://repositorio.uam.es/handle/10486/688469*. july 2018.

[29] Kirill Sakhnov, Ekaterina Verteletskaya, and Boris Simak. "Approach for Energy-Based Voice Detector with Adaptive Scaling Factor." In: *IAENG International Journal of Computer Science* 36.4 (2009).

[30] Tomi Kinnunen et al. "Voice Activity Detection Using MFCC Features and Support Vector Machine". In: 2 (Mar. 2012).

[31] Matthias Mauch et al. "Computer-aided Melody Note Transcription Using the Tony Software: Accuracy and Efficiency". In: May 2015.

[32] kdnuggets. https://www.kdnuggets.com/2017/04/must-know-evaluate-binary-classifier.html. Last visit: july 2022.

[33] Unknown. Last access: 1st July 2022. URL: http://hyperphysics.phy-astr.gsu.edu/hbase/Music/cents.html.

[34] Aristidis Likas. Last access: 1st July 2022. URL: https://www.sciencedirect.com/science/article/abs/pii/S0031320302000602.

[35] Oscar Contreras Carrasco. https://towardsdatascience.com/gaussian-mixture-models-explained-6986aaf5a95. Last access: 1st July 2022.

[36] Vibha Tiwari Tiwari. “MFCC and its applications in speaker recognition”. In: *Int. J. Emerg. Technol.* 1 (Jan. 2010).

[37] Leland Roberts. https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53. Last visit: july 2022.

[38] Python. Last access: 1st July 2022. URL: https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html#sklearn.mixture.GaussianMixture.fit.

[39] Masanori Morise. “Harvest: A High-Performance Fundamental Frequency Estimator from Speech Signals”. In: *Proc. Interspeech 2017*. 2017, pp. 2321–2325. DOI: 10.21437/Interspeech.2017-68.

[40] Praat. https://www.fon.hum.uva.nl/praat/. Last visit: july 2022.

[41] Python. Last visit: july 2022. URL: https://intermediate-and-advanced-software-carpentry.readthedocs.io/en/latest/c++-wrapping.html.

[42] Alexandre Lacoste and Eck Douglas. “A Supervised Classification Algorithm for Note Onset Detection”. In: *EURASIP Journal on Advances in Signal Processing* 2007 (Jan. 2007). DOI: 10.1155/2007/43745.

[43] Britannica. *https://www.britannica.com/art/equal-temperament*. Last access: 1st July 2022.

[44] Wikipedia. https://es.wikipedia.org/wiki/Frecuencia_de_voz#:~:text=El%20discurso%20sonoro%20de%20un,de%20165%20a%20255%20Hz. Last visit: july 2022.

[45] O.K.Ostakis. *Subsequence Search in Event-Interval Sequences, pp. 851-854*. 2015.

[46] Joaquín Amat Rodrigo. April, 2017. URL: https://www.cienciadedatos.net/documentos/34_maquinas_de_vector_soporte_support_vector_machines.

[47] Python. Last access: july 2022. URL: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html.

[48] Python. *https://docs.python.org/3/library/tk.html*. Last access: july 2022.

[49] Python. *https://python-sounddevice.readthedocs.io/en/0.3.15/api/streams.html*. Last access: july 2022.

[50] Python. *https://www.audacityteam.org/about/*. Last visit: july 2022.

[51] Zhiyan Duan et al. "The NUS sung and spoken lyrics corpus: A quantitative comparison of singing and speech". In: *2013 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*. 2013, pp. 1–9. DOI: 10.1109/APSIPA.2013.6694316.

[52] Nagesh Singh Chauhan. *https://www.datasource.ai/es/data-science-articles/metricas-de-evaluacion-de-modelos-en-el-aprendizaje-automatico*. Last visit: july 2022.

[53] Python. *https://scikit-learn.org/stable/modules/model$_e$valuation.html*. Last visit: july 2022.

[54] Python. Last access: july 2022. URL: https://www.datasource.ai/es/data-science-articles/comprension-de-la-matriz-de-confusion-y-como-implementarla-en-python.