Technical University "Gheorghe Asachi" of Iași

Faculty of Electronics, Telecommunications and Information Technology

Specialization: Telecommunications Systems Technology

DEGREE THESIS

Scientific Coordinator:

Assoc. Prof. Nicolae Cleju, PhD

Graduate:

Partnoi Silviu

Iași
2024

Technical University "Gheorghe Asachi" of Iași
Faculty of Electronics, Telecommunications and Information Technology
Specialization: Telecommunications Systems Technology

Machine Learning Techniques in Audio Signal Processing

Scientific Coordinator:

Assoc. Prof. Nicolae Cleju, PhD

Graduate:

Partnoi Silviu

Iași
2024

Technical University "Gheorghe Asachi" of Iași
Faculty of Electronics, Telecommunications and Information Technology
Specialization: Telecommunications Systems Technology

Content

Chapter I. Justification Report

Voice recognition technology has seen significant development and widespread adoption over time. Applications based on voice recognition use advanced machine learning technologies to understand and interpret words, transforming the received information into useful actions. Whether we talk about the concept of "Smart Home," the security field, or the automotive industry, voice recognition is increasingly utilized, offering high quality regardless of the application. The desire to develop technologies that create a more pleasant, friendly, and better living environment has also led to the development of voice recognition systems.

When referring to the concept of "Smart Home," we can think of the "Alexa" application, developed by Amazon. Through it, using certain home systems such as lighting, temperature, and others becomes much easier, offering comfort without the need for physical devices like remotes or panels to operate these services. Thus, just by using a word or phrase, an action that would have required physical human activity can be performed if not for the voice recognition system.

Similarly, the "Siri" application, developed by Apple, offers automatic navigation on the phone to a certain service desired by the user without them having to physically access any application on the device.

Another application of audio recognition is in the automotive field, providing the driver the ability to perform other activities, such as adjusting the music volume or setting the car's temperature, without physically operating the buttons, thereby helping the driver maintain focus on driving.

The desire to undertake this thesis was based on several questions, such as: "On what principle do these applications work? What does the algorithm behind these functionalities look like?" Thus, following the study conducted, I will present in this work one of the many methods that underpin voice recognition applications and systems.

Chapter II. Introduction

2.1. Brief History of Audio Signal Use and Voice Recognition Technology

The use of audio signals has its roots at the very beginning of humanity, with people communicating through various means. Over time, and with the advent of technology, their modes of communication have changed. The invention of the telegraph, radio, and later the telephone, provided people with the facility to communicate over long distances. Today, audio signals are increasingly used in voice recognition, covering a wide range of applications in various fields such as the medical, military, and automotive sectors.

Speech recognition is the process by which human language is captured and processed by a computer. The vocal signal emitted by each of us when we speak is captured by a microphone, and the data obtained from these signals is analyzed by a computing unit to decipher the word or phrase spoken by the person.

This voice recognition technology began in the mid-20th century, in 1952, when a device capable of recognizing spoken digits was created. Later, IBM developed a voice recognition system that printed on a computer screen the words spoken by a person. Given that the data processing and analysis algorithms required a very high computational power, the application could not run on a regular computer. Thus, a special hardware comprising four "Albert" cards was created, which contained enough memory for the voice recognition model to be stored and then learned by the computer.

In addition to IBM, several companies and engineers from Italy contributed to the development of voice recognition algorithms, starting with the recognition of a single word. This work, carried out in 1960 by Angelo Raffaele Meo, led to the creation by the CSELT research center of a special microprocessor implemented in the telecommunications industry, aimed at continuous speech over the phone.

The 21st century begins promisingly for technology where human voice can be recognized. Companies such as Google, Amazon, Baidu, Microsoft, and IBM take this technology to the next level. From recognizing a single word and displaying it on a screen, as done by IBM half a century ago, we now have the capability to control other devices, such as home appliances, car functionalities, and especially mobile phones and computers, through operating system assistants.

Chapter III. Parameter Extraction from the Vocal Signal

Waveform audio files, known as WAV files (waveform audio file format), are standard audio files with the .wav extension. They represent an audio format that extends the bitstream format interface, usable on the Windows operating system. They are similar to other audio formats on the Mac operating system, specifically AIFF and 8SVX. WAV files can be either compressed or uncompressed. In terms of capacity, uncompressed WAV files can be larger compared to other audio formats, such as MP3.

3.1. Mel-Frequency Cepstral Coefficients (MFCC)

The beginning of any voice recognition algorithm is the extraction of features from the audio signal. This step is essential and beneficial because if the processing were done on the raw signal, the result would not be very accurate. This inaccuracy arises because the standard signal is a sum of two components: the useful signal and the noisy signal. Therefore, before processing the signal, it is necessary to decompose it into a series of features that contain the useful signal while eliminating the noisy signal.

Audio signals used for voice recognition are sounds generated by humans. These sounds are filtered by the shape of the vocal tract, which includes the tongue, teeth, etc., and this shape gives us the output sound. The exact representation of the produced sound depends on the shape of the vocal tract, which manifests in the short-term power spectrum envelope. Thus, to determine what the audio signal contains, we need to reproduce this spectral envelope, which can be achieved digitally using the Mel-Frequency Cepstral Coefficient (MFCC) algorithm.

The MFCC algorithm was created by the prestigious Massachusetts Institute of Technology (MIT) in the USA in 1960, initially aimed at studying seismic echoes. Later, it was adopted in the field of voice recognition. Despite the considerable time since this algorithm was created, the use of MFCCs remains current, being one of the most widely used algorithms for extracting audio signal features.

The steps of the MFCC algorithm that must be followed to obtain the cepstral coefficients are:

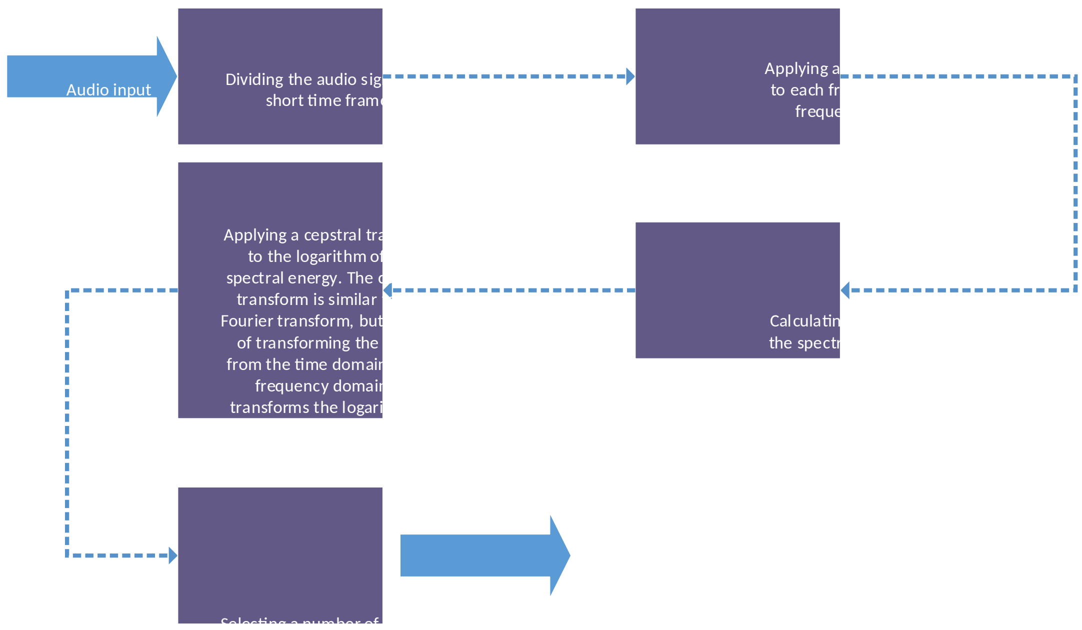

Audio input
Dividing the audio sig short time fram
Applying a to each fr freque
Calculatin the spectr
Applying a cepstral tra to the logarithm of spectral energy. The c transform is similar Fourier transform, but of transforming the from the time domain frequency domain transforms the logari
Selecting a number of

Fig 1. Structure of the MFCC Algorithm

The use of delta (differential) and delta-delta (acceleration) coefficients is based on the idea that to better recognize speech, it is essential to understand the dynamics of the power spectrum, i.e. the trajectories of MFCCs over time. Delta coefficients are calculated using the following formula.

$$d_t = \frac{\sum_{n=1}^{N} n(c_{t+n} - c_{t-n})}{2\sum_{n=1}^{N} n^2}$$

where is d_t a delta coefficient from frame t computed in terms of the static coefficients c_{t-n} to c_{t+n}. n is usually taken to be 2. The acceleration coefficients are computed similarly, but using the differential instead of the static coefficients.

<u>3.1.1 The Vocal Signal</u>

Any word or sound emitted by a human represents an audio signal, characterized by elements such as amplitude, period, intensity, etc. The vocal signal that we generate is represented as a waveform in the time domain, as shown in the figure below:

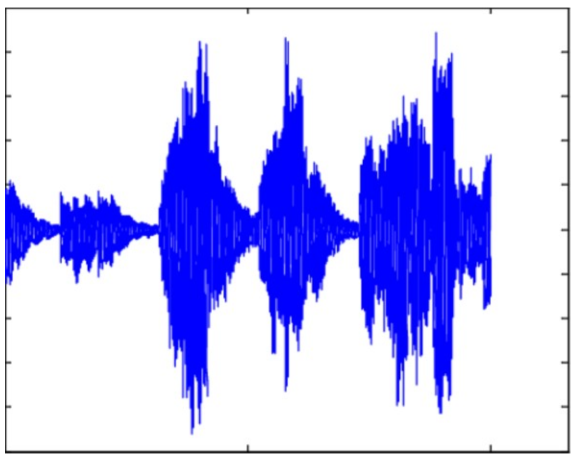


Fig 2. Waveform of the Audio Signa

Given that the audio signal changes constantly on a large scale, it is necessary to divide it into smaller frames, typically 20-40 ms, which will be processed separately. To eliminate distortions that may occur due to abrupt transitions and to improve the quality of the spectral analysis, the Hamming window can be used. This window modifies the beginning and end of the frame by attenuating the amplitude, while keeping the central part of the frame relatively intact.

The waveform of such a window is represented as shown in the image below:

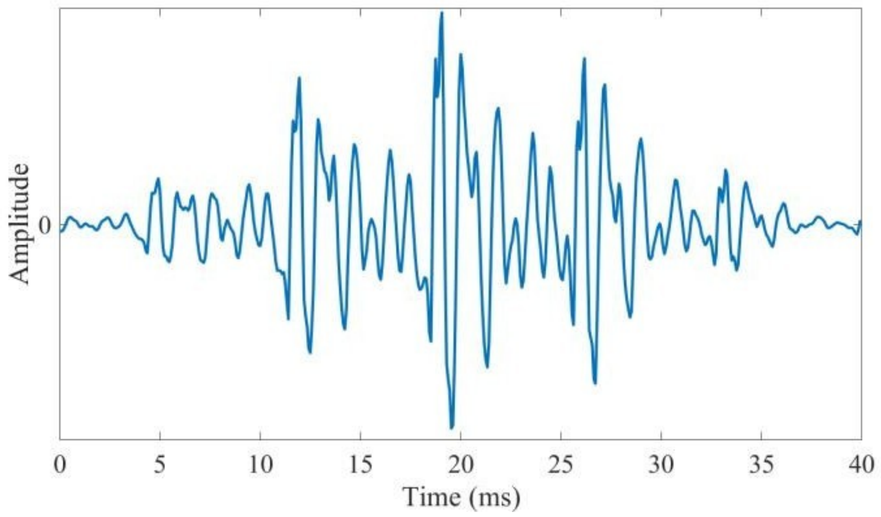


Fig 3. Waveform of a Hamming Window from the Audio Signal

3.1.2 Fourier Transformr

Once the initial audio signal is divided into multiple frames, each frame undergoes a Fourier transform.

The Fourier Transform, named after the mathematician and physicist Joseph Fourier, is an operation aimed at obtaining a complex function that contains the same information as the original function, but organized by component frequencies. For example, consider a signal in the time domain, such as a sinusoidal signal. After applying the Fourier transform, the signal will be decomposed by frequency, producing its spectrum.

The Fourier transform formula is as follows:

$$F(\omega) = \int f(t) e \, \infty \, -i\omega t \, -\infty \, dt, \ \omega \in R \quad (1)$$

where f(t) is the signal in the time domain.

The formula of the Inverse Fourier Transform is:

$$f(t) = F \, -1 \, \{F(\omega)\}(t) = \int F(\omega) e \, \infty \, i\omega t \, -\infty \, d\omega \quad (2)$$

A suggestive image for this transformation is the one below:

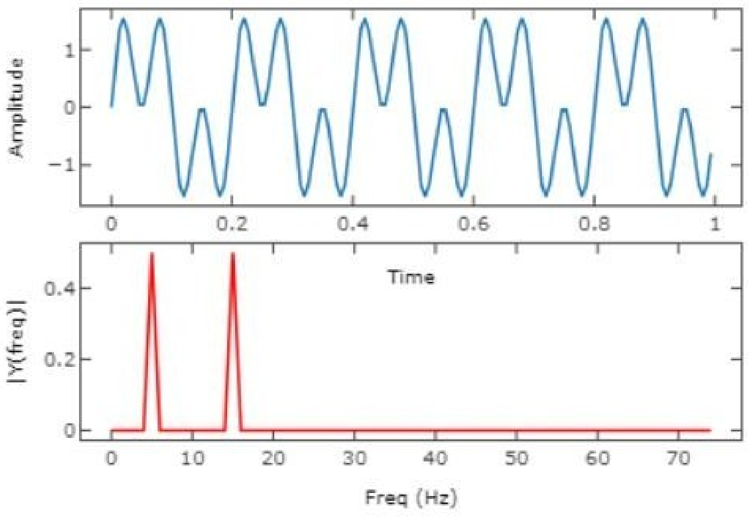


Fig 4. Applying the Fourier Transform to a Signal in the Time Domain

It will be called a Fourier pair: f(t) <=> F(ꞷ), where F(ꞷ) is called spectral density.

The power spectrum of a time series is a way of describing the distribution of power in the discrete frequency components that make up that signal. The statistical average of a signal, measured by its frequency content, is called a spectrum. The spectral density of a digital signal describes the frequency content of the signal.

3.1.3 Mel Scale and Filter Bank

The Mel Scale, named after the word "melody," represents a scale of tones that are perceived by listeners as having equal distance from each other, being a logarithmic scale. The human auditory system has better resolution at lower frequencies compared to higher frequencies. Thus, a sound with frequencies of 200Hz and 300Hz can be easily differentiated. However, the same cannot be said for a sound with frequencies of 1500Hz and 1600Hz. Although both sounds have the same difference of 100Hz between them, due to this Mel scale, the human ear perceives the difference better at lower frequencies.

Unlike the human ear, digital processing has the same differentiation efficiency regardless of frequencies, whether they are high or low, offering good system performance.

So, we will use the Mel scale to map the actual frequency to the frequency perceived by humans. The mapping formula is given below.

$$\mathrm{mel}(f) = 1127\ln(1+\frac{f}{700}) \quad (3)$$

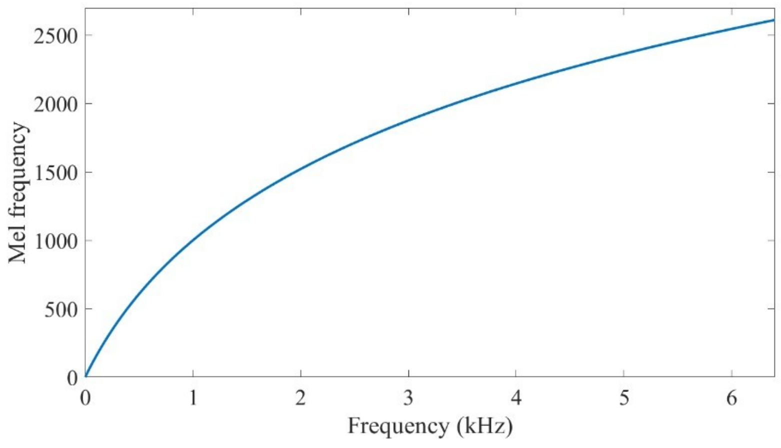


Fig 5. Mel Scale

The Filter Bank is a set of triangular filters, typically ranging from 20 to 40 filters, which we apply to determine the spectral power values.

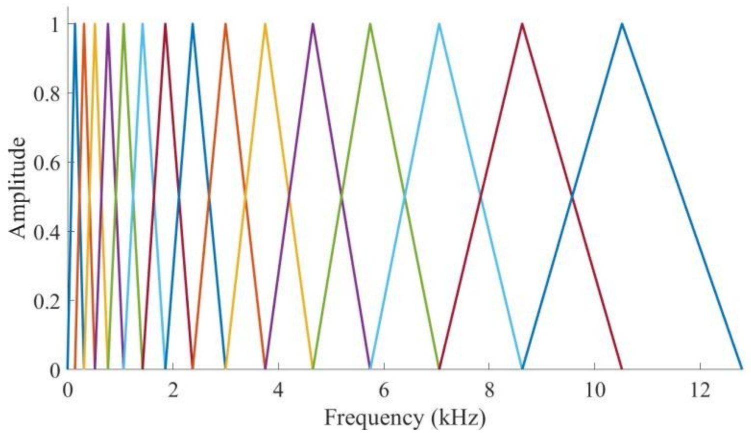


Fig 6. Mel Filter Bank

According to the above graph, for low frequencies, where, according to the Mel scale, differences between sounds are more easily perceived by the human ear, the filters are narrower. For high frequencies, the filters are wider, but this is less important for our analysis. The graph is just a representation of how the filters are applied.

After applying the Fourier transform, its magnitude is extracted, resulting in the magnitude spectrum. This magnitude spectrum is then warped according to the Mel scale to adapt the frequency resolution to the characteristics of the human ear.

After each spectrum is multiplied by the triangular filter, the Mel-weighted spectrum is obtained, represented by the spectrogram in Figure 7.

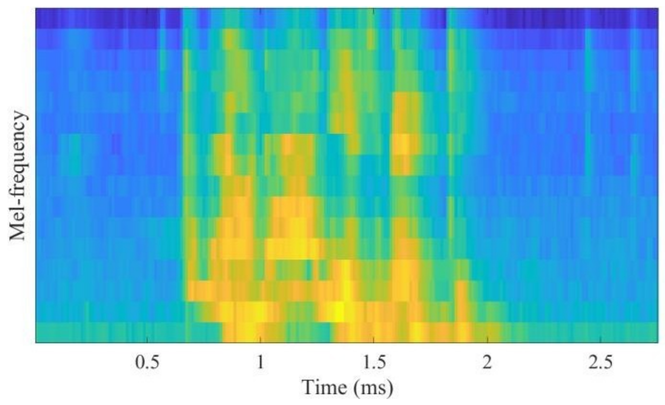


Fig 7. Mel Frequency Spectrogram

Scara Mel and the linear frequency scale are related according to the relationship:

$$\text{melf} = 2595 * \log_{10}(1 + \text{linf} / 700) \quad (4)$$

where melf is the Mel frequency, and linf is the linear frequency. What this formula does is to transform frequencies from the linear domain to the logarithmic domain.

The Filter Bank, composed of a sequence of overlapping triangular filters, divides the magnitude spectrum into a number of critical bands.

These filter banks are defined by several parameters, such as:

a. Number of Mel filters
b. Minimum frequency
c. Maximum frequency

The value of the minimum frequency is suggested to be higher than 50-60 Hz to cancel out the noise caused by alternating current. As for the maximum frequency value, it should be lower than the Nyquist frequency.

The formula delta_melf = (melf_max = melf_min) / (F + 1) (5), It helps calculate the fixed resolution, where melf_max și melf_min are the Mel scale frequencies corresponding to linf_max and linf_min linear frequencies.

To transition from the Mel scale to the frequency domain of the spectra, the inverse relationship between linear frequency and Mel frequency is used:

$$M^{-1}(m) = 700(e^{m/1125}-1) \quad (6)$$

3.1.4 Logarithm of the Spectrum Amplitude

The human ear doesn't perceive volume movement on a linear scale. To double the volume of an audio wave, the wave's energy must be amplified by a factor of amplification, the value of which is 8.

Once we have the spectra of the audio signal, they will be logarithmized.

The waveform of the logarithmized power spectra is represented in Figure 8.

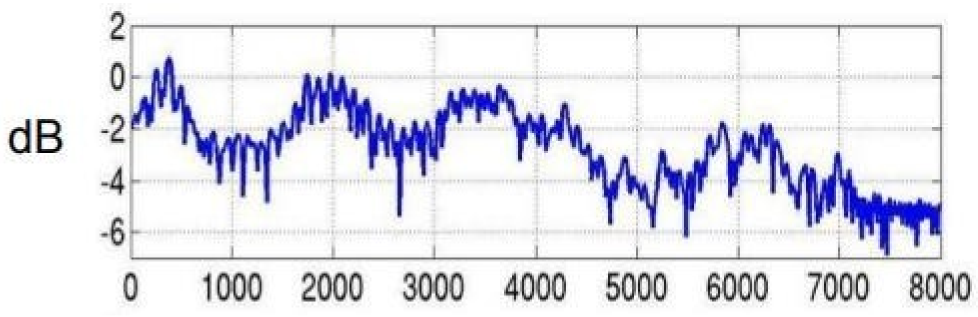


Fig 8. Applying the logarithm to the power spectrum

This waveform consists of two signals: the spectral envelope and the spectral noise.

The spectral envelope contains the characteristics of the sound, represented by phonetics, voice timbre, etc. These sound information is represented by the positive alternations of the waveform. The spectral envelope corresponds to the shape of the sound emitted by the vocal tract, a filtered sound, free of noise.

Figure 9 represents the spectral envelope, and the circled peaks represent the characteristics of the sound.

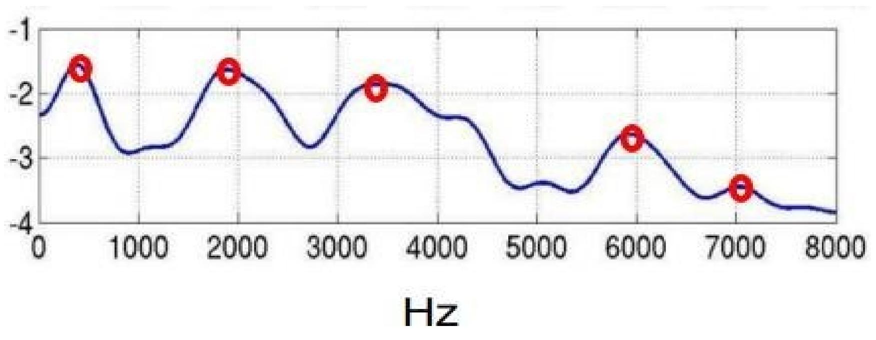


Fig 9. The spectral envelope and the feature vector containing the sound characteristics

High frequencies are associated with the vibrations caused by the vocal cords. As the exact characteristics of vocal sounds are desired, the high frequencies will be removed from the waveform based on a low-pass filter (LPF).

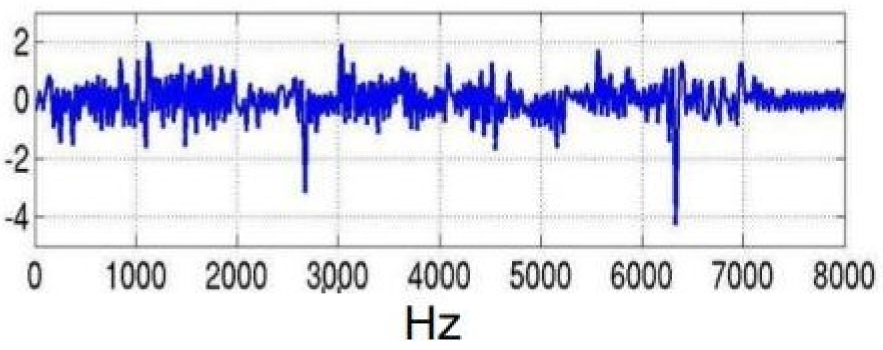


Fig 10. Spectral Noise

A mathematical calculation will help us better understand how the removal of high frequencies occurs, to the detriment of obtaining the envelope.

In the time domain, the formula for the audio signal is as follows:

$$x(t) = e(t) + h(t) \quad (7)$$

where x(t) is the vocal signal, e(t) is the spectral noise, and h(t) is the spectral envelope.

Following the application of the Fourier transform, the formula is rewritten as:

$$X(t) = E(t) + H(t) \quad (8)$$

where X(t) is the spectrum of the vocal signal, E(t) is the spectrum of the spectral noise, and H(t) is the spectral envelope. Applying the logarithm, the formula is as follows:

$$\log(X(t)) = \log(E(t) * H(t)) \quad (9)$$

$$\log(X(t)) = \log(E(t)) + \log(H(t)) \quad (10)$$

where log(E(t) contains the spectral noise, which is not essential in determining the characteristics of our signal, which is why it is canceled out.

Therefore, in the end, the spectral envelope is the only signal that remains, containing the necessary information for determining the cepstral coefficients..

3.2.5 Inverse Fourier Transform. Determining the cepstrum.

The application of the inverse Fourier transform results in the determination of Mel-frequency cepstral coefficients (MFCC). This result is based on the following formula:

$$C(x(t)) = \mathcal{F}^{-1}[\log(F[x(t)])] \quad (11)$$

In Figure 11, the MFCC spectrogram is illustrated. This spectrogram of Mel-frequency cepstral coefficients contains information about the spectral envelope of the speech signal, representing it in an abstract domain.

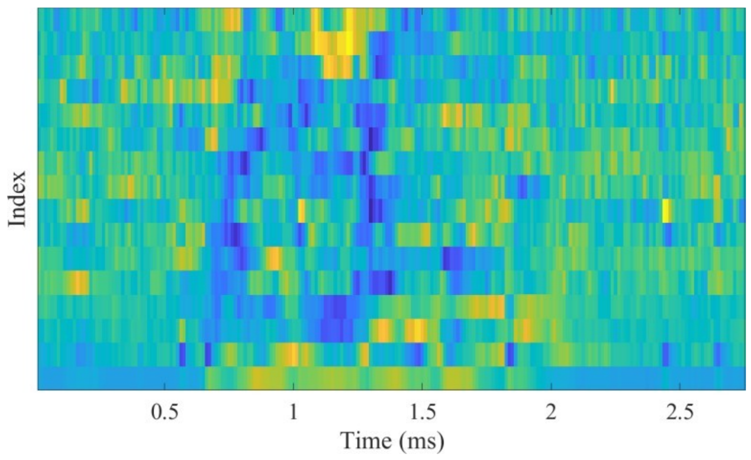


Fig 11. The MFCC spectrogram

3.3 Dynamic time warping (DTW)

A voice recognition system aims to identify commands, words, or phrases spoken by a person. The way a word is pronounced by the same person can vary from one utterance to another, even more so when words are spoken by different people. It is possible that the first time a person speaks, the vocal signal might be longer in duration, and the second time shorter. Therefore, the program should be able to identify the transmitted message in both situations. Suppose that the office where we work opens via a voice recognition algorithm, and each employee needs to state their first and last name. Even if I say "Partnoooi Silviuuuu" with more emphasis once, and then simply "Partnoi Silviu" another time, the system will still open the office door.

What makes it possible for the system to identify the command we speak is the Dynamic Time Warping (DTW) algorithm. This algorithm compares and aligns data sequences with equal or different temporal dimensions.

Let's assume we have two signals, each represented by a data vector that identifies the characteristics of that signal (signal_1 = [1;2;3;4] and signal_2 = [5;6;7;8]). If we want to identify similarities between these signals without the DTW algorithm, we would use the Euclidean method, checking each element of the first vector against its corresponding element in the second data vector. The example mentioned above assumes two signals identical in size, but how will this method work if the data size is different? If the Euclidean method is applied, it will compare a part of the existing data, but the vector containing more data will have some elements left unchecked. In such a situation, the DTW algorithm comes into play, differentiating itself from the Euclidean method by allowing one element from the first vector to be compared with multiple elements from the second vector and vice versa. Thus, regardless of the size difference between the two signals, all data contained within them will be compared.

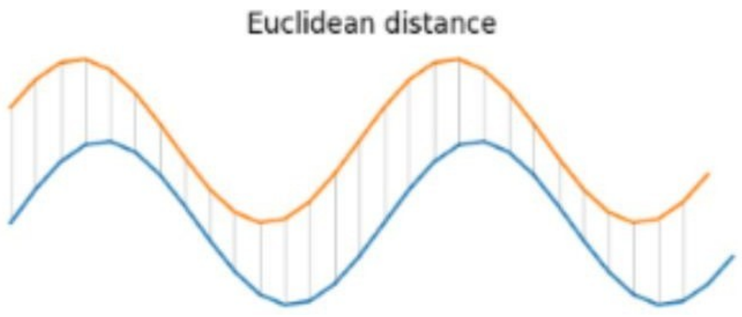


Fig 12. Data matching using the Euclidean method

In Figure 12, the Euclidean method is presented, where the data of two signals (the orange signal and the blue signal) are compared element by element.

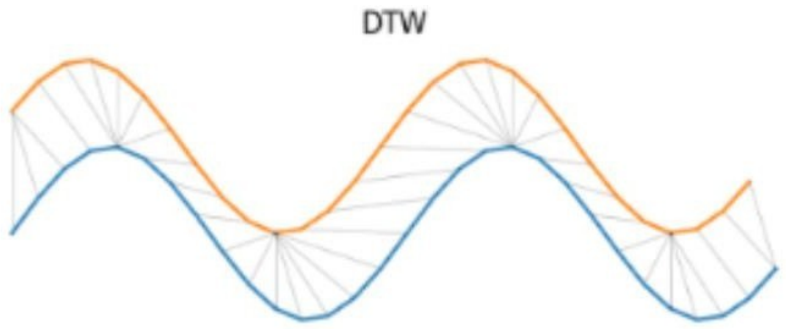


Fig 13. Data matching using the DTW algorithm

The way the DTW algorithm works is visible in Figure 13. Multiple elements from the data vector of the orange signal are matched with a single element from the blue signal, and vice versa.

The DTW algorithm is highly beneficial when dealing with a wide variety of people using a voice recognition system, providing it with high performance and good results.

Although we have discussed this algorithm, the practical part of this work does not include its implementation. It was mentioned solely for informational purposes.

Chapter IV. Data Classification

4.1 The Concept of Artificial Intelligence, Machine Learning, and Deep Learning

Artificial Intelligence is one of the most current technologies on the market, also known as "The Technology of the Future." With a history of over 50 years, this technology leaves its mark in every possible field of activity. Whether in the online environment, where it is used to recommend products and services based on user interactions, or in the medical field, where it is extensively exploited to diagnose health issues, AI creates a safer, more comfortable, and precise living environment, providing people with what they need when they need it. It efficiently manages resources such as time and information, benefiting both users and beneficiaries of this technology.

Artificial Intelligence represents the replication of human intelligence in electronic systems, machines, robots, with the goal of performing activities similar to those done by humans. This technology is implemented in various systems, aimed at training them to perform tasks currently done by humans. For example, consider AI applied in a warehouse setting. Currently, product storage is managed by humans, but in the future, robots equipped with AI capabilities will be able to visually identify products, learn how to handle them, and store them in designated locations, similar to human operators.

This is just one of the many scenarios where artificial intelligence can be applied.

4.1.2 Machine Learning

Machine Learning is a branch of Artificial Intelligence that aims to create systems capable of learning and improving performance based on processed datasets.

Often, the terms "Artificial Intelligence" and "Machine Learning" are considered synonymous, but there are significant differences between them. One key difference is that all Machine Learning systems fall under Artificial Intelligence, but not all Artificial Intelligence systems incorporate Machine Learning.

Machine Learning algorithms can be categorized into two types: supervised and unsupervised learning. The distinction lies in how the dataset is learned and predictions are made.

In supervised learning, the system makes decisions based on a labeled dataset. This means the dataset's elements are classified into classes. When the system needs to make a decision, such as recognizing a word in a speech recognition system (e.g., "Forward," "Backward," "Start," "Stop"), each input signal (e.g., an audio signal) is compared to each element in the dataset. The input is then assigned to the class that most closely matches its characteristics. For instance, if the system receives the signal "Start" and it corresponds to class 1 in the dataset, the system processes it and returns the label of the assigned class.

On the other hand, unsupervised learning does not involve predefined classes. It is more independent, identifying patterns and features from the data and subsequently creating clusters or groups based on these features.

The necessity for these algorithms arises from two essential factors: complexity and adaptability. If a system's goal is to detect various types of images, using traditional methods would involve coding numerous specifications and checks for each image, which is both laborious and challenging. In contrast, Machine Learning algorithms recognize images based on their "learned experiences" from the dataset, resulting in more accurate outcomes.

Additionally, these algorithms are valued for their adaptability to different situations. Once trained, they can process input data correctly, even if the data is constantly changing. This adaptability ensures that the algorithm maintains accuracy, unlike standard scenarios where the algorithm's precision would decrease if the input data were modified.

Overall, Machine Learning enhances systems' capabilities by enabling them to learn from data and adapt to new information, making them more efficient and effective in various applications.

4.1.3 Deep Learning

Deep Learning, like Machine Learning, is a subset of Artificial Intelligence. Essentially, deep learning refers to neural networks with at least three layers. Using just one layer typically doesn't yield optimal performance. However, as the number of layers increases—creating a more complex and "intelligent" neural network—the algorithm's performance improves significantly, becoming more efficient and providing more accurate predictions.

While deep learning is a part of machine learning, the differences lie in how they learn to identify input data. Machine learning relies on structured datasets, where input data is evaluated based on this dataset, and predictions are made accordingly. On the other hand, deep learning involves identifying patterns in input data based on distinctive features. With each input, the accuracy of predictions improves, making deep learning suitable for deep understanding and classification of data by hierarchically organizing extracted features. This capability allows deep learning to tackle highly complex problems such as image recognition, language processing, and more.

To delve briefly into the foundational concept of deep learning—neural networks—we can introduce a type of neural network commonly used in current technologies.

CNN(Convolutional Neural Network)

Convolutional neural network is one of the neural networks used by deep learning. This is characterized by multiple convolutional levels, levels to which mathematical functions are applied, such as the convolution function, hence the name of this network. Its applicability is especially in image recognition, making a model based on images, in order to determine their characteristics.

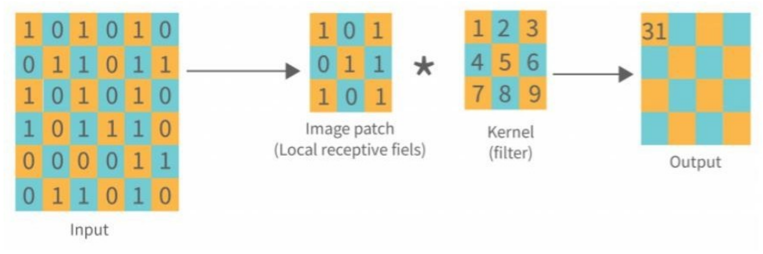


Fig 14. The layers of a convolutional neural network

The architecture of a CNN consists of 3 levels, which lead to achieving good predictions for the network..

1. Convolutional Layer - This is the cornerstone of CNNs, aiming to perform convolutional functions using a kernel. The kernel's main task is to adjust the matrix of values along rows and columns to simplify the input information. In addition to the convolutional functions, which are linear operations, there are also nonlinear activation functions such as Rectified Linear Unit (ReLU), which is used to obtain the output of the convolutional layer.

2. **Pooling layer –**The responsibility of this layer is to reduce the dimensionality of the matrix to process a smaller volume of data. This reduction in dimensionality can be achieved through two methods: max pooling and average pooling. The kernel is responsible for this task, grouping the data into average and maximum values.

3. **Fully Connected Layer –** These layers are found at the end of a convolutional architecture. The purpose of these layers is to aggregate all the features extracted from earlier layers and prepare them for the final classification.

Countless applications and state-of-the-art systems used in voice recognition projects employ the concept of deep learning, providing excellent quality to the system. However, in the practical work of this project, machine learning technology was used, where word recognition was based on a dataset.

4.2 Classification Algorithms

Classification algorithms are methods used in machine learning aimed at learning mathematical models to classify objects into different classes or categories. Classification involves assigning an object or example to a specific class or category based on its analyzed features.

The goal of these learning algorithms is to train on datasets to make accurate predictions about how input data should be classified into their respective classes.

These algorithms find wide applicability across various domains. In the medical field, for instance, classification algorithms are used to identify whether a specific disease is present or not, or to assess the optimal condition of organs in the body. In office settings, we encounter classification algorithms daily, particularly in managing emails. Given the high volume of emails received from diverse individuals and organizations, there's a need to organize these messages for easier retrieval. Filters can be applied to redirect messages from specific persons or organizations into designated folders, facilitated by classification algorithms.

For example, the input data for our algorithm consists of the names of individuals sending emails. The algorithm uses a dataset containing names of frequent senders and compares the input data's characteristics with each class. Based on its prediction, the email is assigned to the class containing the name of the person or organization. These are just two examples from a multitude of domains where classification algorithms play a crucial role.

There are several data classification algorithms, each differing in their characteristics and unique working methods. Some of these algorithms include:

a. Naive Bayes Algorithm - based on Bayes' theorem, assumes conditional independence of features;
b. Logistic Regression Algorithm - used to estimate the probability of object belonging to classes;
c. Support Vector Machines - involves separating data classes;Arborii de decizie – sunt reprezentate caracteristicile și valorile acestora;
d. Decision Trees - represent features and their values;
e. Neural Networks - used in understanding and classifying complex features.

<u>4.3 The K-Nearest Neighbors Algorithm</u>

In addition to the numerous classification algorithms listed above, another algorithm that falls into this category is called "K-Nearest Neighbors" or KNN. This algorithm forms the basis of several learning methods such as multiple classification and spectral clustering. Behind these algorithms lies the principle of discovering the nearest neighbors. What does this mean? Essentially, a point is selected for which its membership in one of the existing classes is desired. The way this point is assigned to a specific class involves identifying the majority of samples from a class that are located near this point of interest. Distance determination is based on a simple method such as the Euclidean distance, which calculates the distance between two points.

Although the way these algorithms work is not particularly complex, but rather straightforward, their results have proven to be very effective. Their use in classifying high-resolution images, such as those from satellites, has confirmed their well-deserved position at the top among classification algorithms.

Supervised learning based on neighbors has two variants:

a. **Classification** - for data with discrete labels
b. **Regression** - for data with continuous labels

4.3.1 Nearest Neighbors Classification

Classification of a point based on the nearest neighbor principle employs instance-based learning, which memorizes instances of training data and does not utilize a generalized internal model. This means that for each point that needs to be assigned to a specific class, the distance from it to the other points in its neighborhood . The class containing the most points in the vicinity of the query point will be the class to which this point will be assigned.

The number of points selected in the neighborhood of the query point is an integer k, whose value can vary. However, as this number increases, the boundaries for differentiating between classes become less distinct. By applying the distance method to classify voice commands, a certain weight is assigned to the points in the vicinity of the object being classified. The value of this weight is equal to the inverse of the distance between the neighbor and the point. Therefore, if the distance between the neighbor and the point is smaller, the neighbor will have a higher weight. In situations where the distance is greater, the assigned weight will be lower.

In addition to this method, which involves calculating the distance to k neighbors to determine the class membership of a point, another method is used that involves creating a radius that defines the neighborhood. This method is applied when the data is not uniformly distributed and neighbors are relatively sparse.

As mentioned earlier, at the core of classification lies the calculation of the distance between the query point and the points in its vicinity. However, in addition to distance, a uniform weight is also applied. This uniform weight involves assigning an equal value to all neighbors closer to the query point.

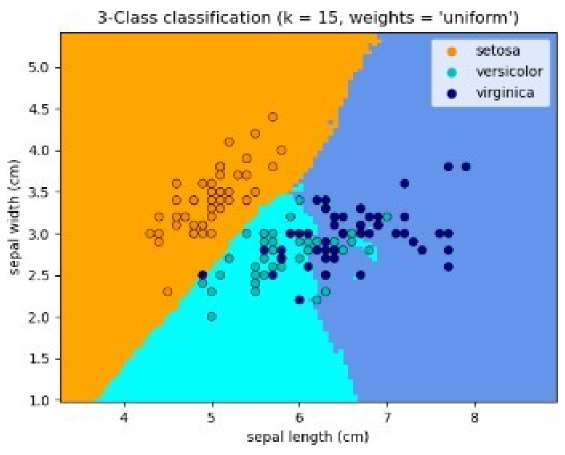


Fig 15. Classification based on weights

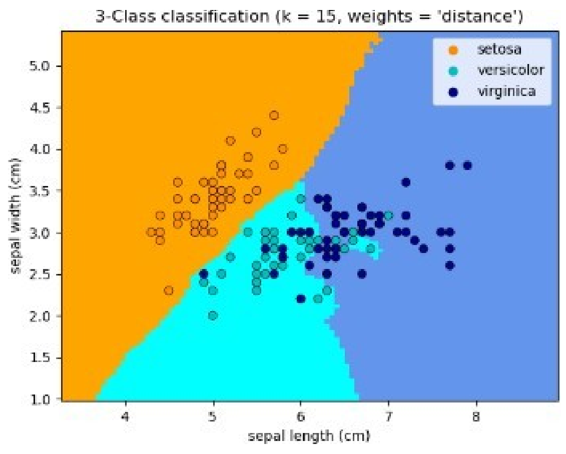


Fig 16. Classification based on distance

4.3.2 Regression

For the classification of points into classes represented by continuous variables, K-Nearest Neighbors regression is used. The class membership of the point is obtained by calculating the mean of the classes of the nearest neighbors.

Similar to classification, two methods are used to calculate the distance to points in the vicinity of the query point: the method that involves calculating the distance to k neighbors and the method that calculates the distance to a small number of neighbors within a specific radius. Parallel to classification, the regression method also determines the membership of an object in a certain class by applying Euclidean distance and weighting towards the neighbors of the object, as illustrated in Figure 17.

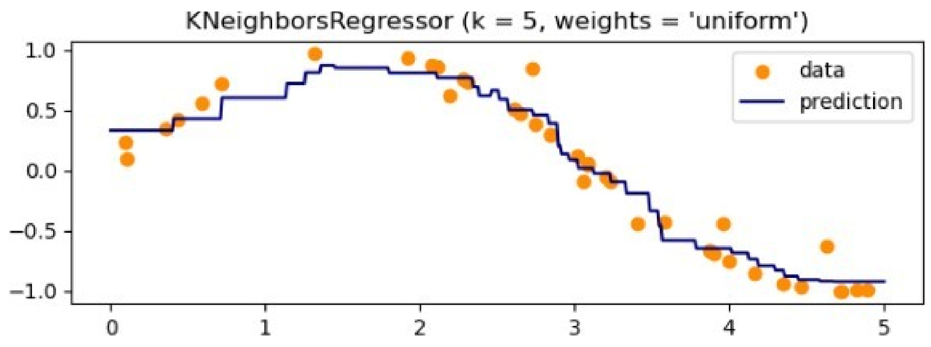


Figure 17. Regression based on weights

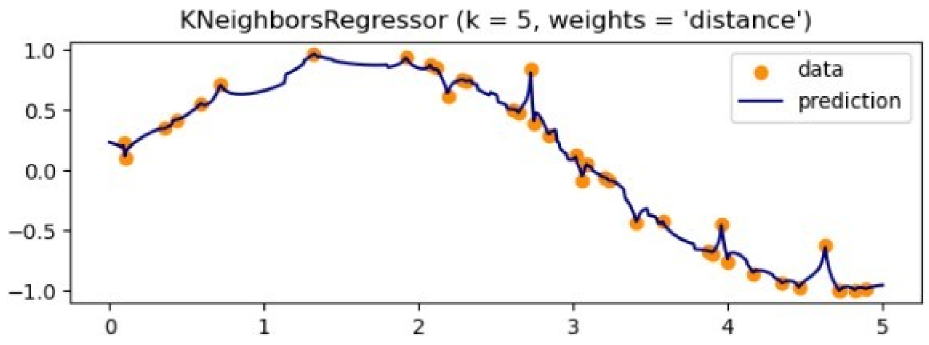


Fig 18. Distance-based regression

4.3.2 Neighborhood component analysis

Standard classification of an object into one of the existing classes is based on calculating the Euclidean distance between the object and its neighboring points. An improvement has been made to this method, specifically the analysis of the components within the neighborhood. This enhancement aims to maximize the value obtained through the Euclidean distance calculation during the training phase

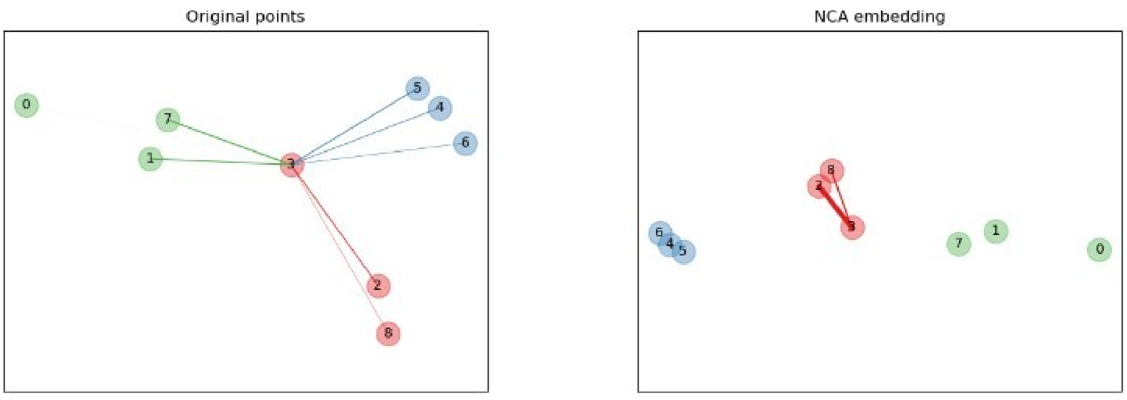


Fig 19. Neighborhood component analysis

In the above figure, an example is depicted that contrasts the classic method of classification (where Euclidean distance is calculated) with a method that maximizes the original classification. The element intended to be classified is element number 3. The emphasis of the lines connecting this element to the others can be interpreted as the weight of each neighbor individually (strong emphasis of a line representing a higher weight). In the second image, the improvement brought by analyzing the components in the neighborhood is evident, as object number 3 is very easily classified into the "Red" class compared to the first image, where classifying the object is quite difficult.

Chapter V. Practical Development of the Work

The voice-commanded robotic system developed in this work aims to highlight voice recognition algorithms, which we now encounter on all smart devices we have. Even though, in this paper, the classification results are used to control some motors, attention should be focused on the processes occurring at the level of the MFCC and KNN algorithms, considering that what can be commanded or executed following the classification is very broad.

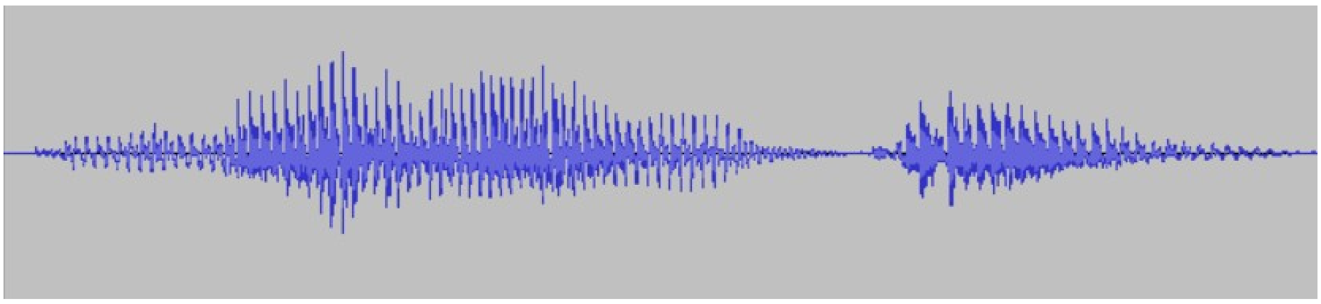
5.1 Software Component

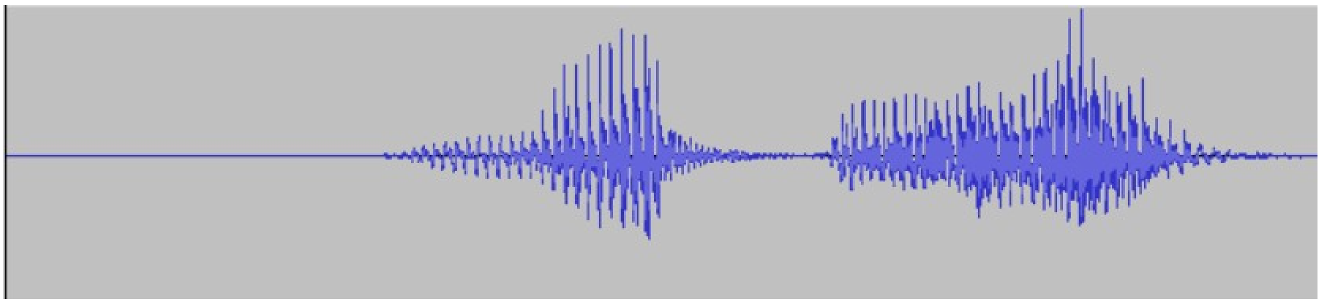
5.1.1 The Dataset

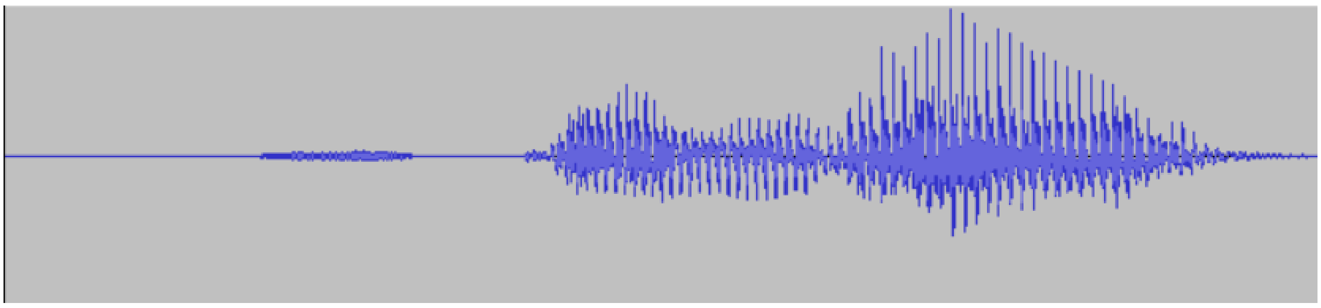
The first step in the process of this system is the creation of the dataset. The dataset recording was performed using a script written in the Python programming language. Upon starting the program, the operating system's recording functions are accessed via the programming language's API, and commands are recorded for two seconds. After recording, a file with the .wav (waveform audio format) extension is created and saved to a predefined path in a directory on the computer. Subsequently, this file will be processed, representing the input data for the voice recognition algorithm.

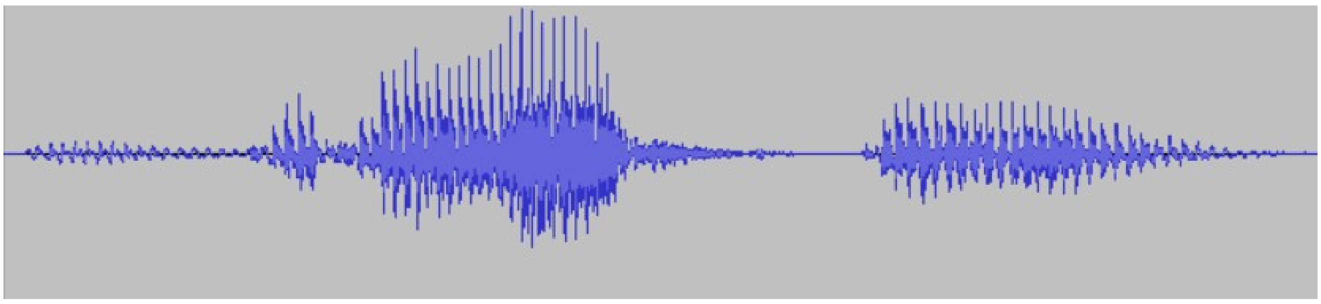
Considering that the system aims to recognize specific words, the dataset will consist of a series of recordings, including both the words the system needs to recognize and a series of "incorrect" words. Each word to be recognized is recorded 100 times, each constituting a class in the classification process. The "Other" class contains 40 distinct words that do not need to be recognized.

Next, I will present the waveform of a word from each class.


"Forward" waveform

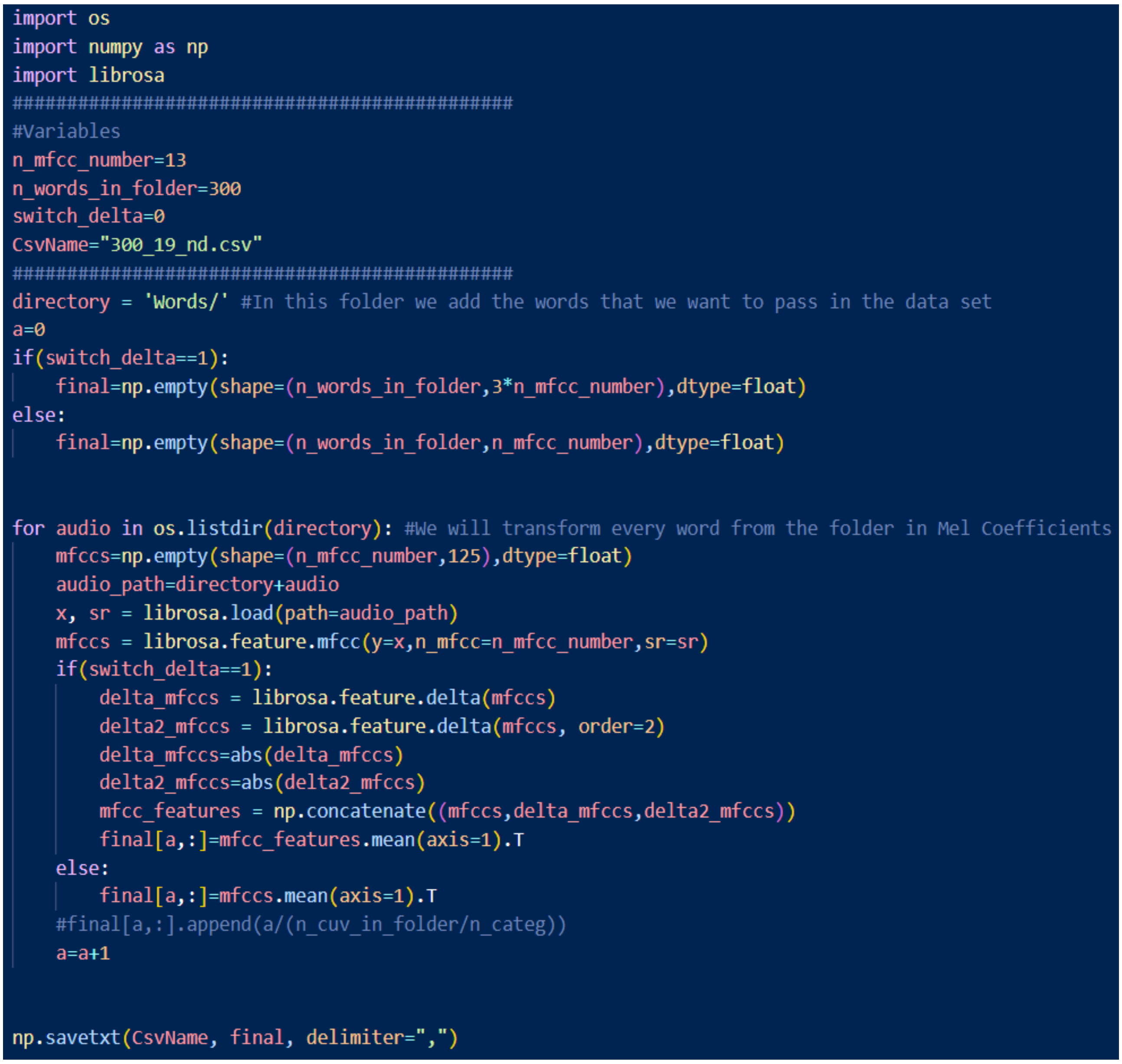

"Back" waveform


"Left" waveform


"Right" waveform

Once created, the audio signals represented by the wav file will be picked up and processed by the MFCC algorithm. To retrieve and process the audio signals, a program was created in Python, which takes the wav files from a predefined directory and applies the MFCC algorithm to each file separately.

The program in Python is as follows:



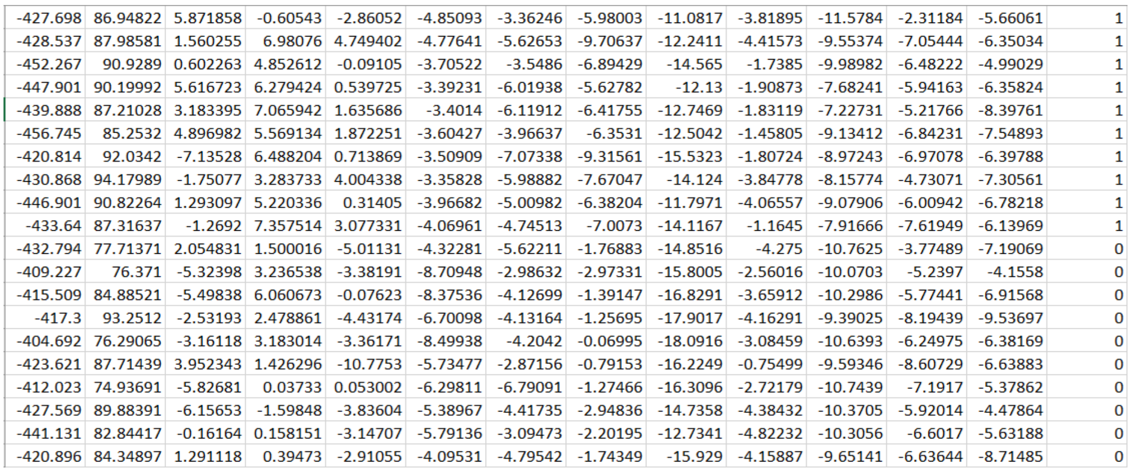
```python
import os
import numpy as np
import librosa
###############################################
#Variables
n_mfcc_number=13
n_words_in_folder=300
switch_delta=0
CsvName="300_19_nd.csv"
###############################################
directory = 'Words/' #In this folder we add the words that we want to pass in the data set
a=0
if(switch_delta==1):
    final=np.empty(shape=(n_words_in_folder,3*n_mfcc_number),dtype=float)
else:
    final=np.empty(shape=(n_words_in_folder,n_mfcc_number),dtype=float)


for audio in os.listdir(directory): #We will transform every word from the folder in Mel Coefficients
    mfccs=np.empty(shape=(n_mfcc_number,125),dtype=float)
    audio_path=directory+audio
    x, sr = librosa.load(path=audio_path)
    mfccs = librosa.feature.mfcc(y=x,n_mfcc=n_mfcc_number,sr=sr)
    if(switch_delta==1):
        delta_mfccs = librosa.feature.delta(mfccs)
        delta2_mfccs = librosa.feature.delta(mfccs, order=2)
        delta_mfccs=abs(delta_mfccs)
        delta2_mfccs=abs(delta2_mfccs)
        mfcc_features = np.concatenate((mfccs,delta_mfccs,delta2_mfccs))
        final[a,:]=mfcc_features.mean(axis=1).T
    else:
        final[a,:]=mfccs.mean(axis=1).T
    #final[a,:].append(a/(n_cuv_in_folder/n_categ))
    a=a+1


np.savetxt(CsvName, final, delimiter=",")
```

The program takes the records from the file "Data_set_recognize_words", and then each record is loaded into the program by means of the code line signal, sample_rate = librosa.load(audio_path). This function will return a "signal" vector that will contain the samples of the audio signal and its sampling rate.

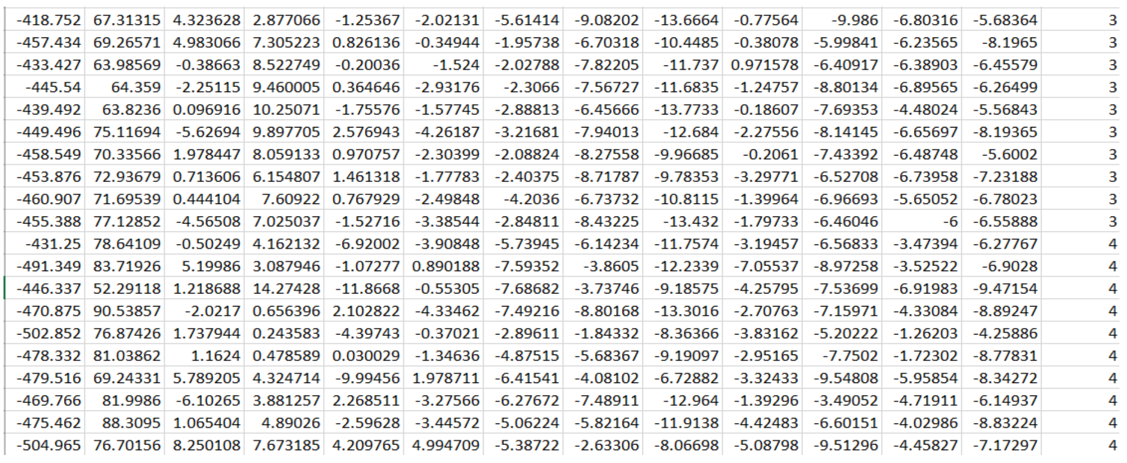
The data set with the characteristics of each individual signal is created in the form of a csv file. The mel frequency domain cepstral coefficients (MFCC) will be calculated for each signal separately, by means of the function mfccs = librosa.feature.mfcc(y= signal,n_mfcc=13, sr = sr), which will return the MFCC matrix. The number of cepstral coefficients for each signal will be equal to 13, and each of these coefficients consists of a number of windows containing information about the audio signal. For the processing of these windows, the average value calculation method is applied, so that each cepstral coefficient has a single value and not a vector of values. Calculating the average value has a multiple role: it reduces the size of the cepstral coefficient; perform a summary of the signal characteristics, without taking into account each individual characteristic; it reduces the noise or variations that the signal may have.

Later, following signal processing, the classes are also defined, depending on the name of the wav file.

The dataset in the csv file looks like this:



-427.698	86.94822	5.871858	-0.60543	-2.86052	-4.85093	-3.36246	-5.98003	-11.0817	-3.81895	-11.5784	-2.31184	-5.66061	1
-428.537	87.98581	1.560255	6.98076	4.749402	-4.77641	-5.62653	-9.70637	-12.2411	-4.41573	-9.55374	-7.05444	-6.35034	1
-452.267	90.9289	0.602263	4.852612	-0.09105	-3.70522	-3.5486	-6.89429	-14.565	-1.7385	-9.98982	-6.48222	-4.99029	1
-447.901	90.19992	5.616723	6.279424	0.539725	-3.39231	-6.01938	-5.62782	-12.13	-1.90873	-7.68241	-5.94163	-6.35824	1
-439.888	87.21028	3.183395	7.065942	1.635686	-3.4014	-6.11912	-6.41755	-12.7469	-1.83119	-7.22731	-5.21766	-8.39761	1
-456.745	85.2532	4.896982	5.569134	1.872251	-3.60427	-3.96637	-6.3531	-12.5042	-1.45805	-9.13412	-6.84231	-7.54893	1
-420.814	92.0342	-7.13528	6.488204	0.713869	-3.50909	-7.07338	-9.31561	-15.5323	-1.80724	-8.97243	-6.97078	-6.39788	1
-430.868	94.17989	-1.75077	3.283733	4.004338	-3.35828	-5.98882	-7.67047	-14.124	-3.84778	-8.15774	-4.73071	-7.30561	1
-446.901	90.82264	1.293097	5.220336	0.31405	-3.96682	-5.00982	-6.38204	-11.7971	-4.06557	-9.07906	-6.00942	-6.78218	1
-433.64	87.31637	-1.2692	7.357514	3.077331	-4.06961	-4.74513	-7.0073	-14.1167	-1.1645	-7.91666	-7.61949	-6.13969	1
-432.794	77.71371	2.054831	1.500016	-5.01131	-4.32281	-5.62211	-1.76883	-14.8516	-4.275	-10.7625	-3.77489	-7.19069	0
-409.227	76.371	-5.32398	3.236538	-3.38191	-8.70948	-2.98632	-2.97331	-15.8005	-2.56016	-10.0703	-5.2397	-4.1558	0
-415.509	84.88521	-5.49838	6.060673	-0.07623	-8.37536	-4.12699	-1.39147	-16.8291	-3.65912	-10.2986	-5.77441	-6.91568	0
-417.3	93.2512	-2.53193	2.478861	-4.43174	-6.70098	-4.13164	-1.25695	-17.9017	-4.16291	-9.39025	-8.19439	-9.53697	0
-404.692	76.29065	-3.16118	3.183014	-3.36171	-8.49938	-4.2042	-0.06995	-18.0916	-3.08459	-10.6393	-6.24975	-6.38169	0
-423.621	87.71439	3.952343	1.426296	-10.7753	-5.73477	-2.87156	-0.79153	-16.2249	-0.75499	-9.59346	-8.60729	-6.63883	0
-412.023	74.93691	-5.82681	0.03733	0.053002	-6.29811	-6.79091	-1.27466	-16.3096	-2.72179	-10.7439	-7.1917	-5.37862	0
-427.569	89.88391	-6.15653	-1.59848	-3.83604	-5.38967	-4.41735	-2.94836	-14.7358	-4.38432	-10.3705	-5.92014	-4.47864	0
-441.131	82.84417	-0.16164	0.158151	-3.14707	-5.79136	-3.09473	-2.20195	-12.7341	-4.82232	-10.3056	-6.6017	-5.63188	0
-420.896	84.34897	1.291118	0.39473	-2.91055	-4.09531	-4.79542	-1.74349	-15.929	-4.15887	-9.65141	-6.63644	-8.71485	0



-418.752	67.31315	4.323628	2.877066	-1.25367	-2.02131	-5.61414	-9.08202	-13.6664	-0.77564	-9.986	-6.80316	-5.68364	3
-457.434	69.26571	4.983066	7.305223	0.826136	-0.34944	-1.95738	-6.70318	-10.4485	-0.38078	-5.99841	-6.23565	-8.1965	3
-433.427	63.98569	-0.38663	8.522749	-0.20036	-1.524	-2.02788	-7.82205	-11.737	0.971578	-6.40917	-6.38903	-6.45579	3
-445.54	64.359	-2.25115	9.460005	0.364646	-2.93176	-2.3066	-7.56727	-11.6835	-1.24757	-8.80134	-6.89565	-6.26499	3
-439.492	63.8236	0.096916	10.25071	-1.75576	-1.57745	-2.88813	-6.45666	-13.7733	-0.18607	-7.69353	-4.48024	-5.56843	3
-449.496	75.11694	-5.62694	9.897705	2.576943	-4.26187	-3.21681	-7.94013	-12.684	-2.27556	-8.14145	-6.65697	-8.19365	3
-458.549	70.33566	1.978447	8.059133	0.970757	-2.30399	-2.08824	-8.27558	-9.96685	-0.2061	-7.43392	-6.48748	-5.6002	3
-453.876	72.93679	0.713606	6.154807	1.461318	-1.77783	-2.40375	-8.71787	-9.78353	-3.29771	-6.52708	-6.73958	-7.23188	3
-460.907	71.69539	0.444104	7.60922	0.767929	-2.49848	-4.2036	-6.73732	-10.8115	-1.39964	-6.96693	-5.65052	-6.78023	3
-455.388	77.12852	-4.56508	7.025037	-1.52716	-3.38544	-2.84811	-8.43225	-13.432	-1.79733	-6.46046	-6	-6.55888	3
-431.25	78.64109	-0.50249	4.162132	-6.92002	-3.90848	-5.73945	-6.14234	-11.7574	-3.19457	-6.56833	-3.47394	-6.27767	4
-491.349	83.71926	5.19986	3.087946	-1.07277	0.890188	-7.59352	-3.8605	-12.2339	-7.05537	-8.97258	-3.52522	-6.9028	4
-446.337	52.29118	1.218688	14.27428	-11.8668	-0.55305	-7.68682	-3.73746	-9.18575	-4.25795	-7.53699	-6.91983	-9.47154	4
-470.875	90.53857	-2.0217	0.656396	2.102822	-4.33462	-7.49216	-8.80168	-13.3016	-2.70763	-7.15971	-4.33084	-8.89247	4
-502.852	76.87426	1.737944	0.243583	-4.39743	-0.37021	-2.89611	-1.84332	-8.36366	-3.83162	-5.20222	-1.26203	-4.25886	4
-478.332	81.03862	1.1624	0.478589	0.030029	-1.34636	-4.87515	-5.68367	-9.19097	-2.95165	-7.7502	-1.72302	-8.77831	4
-479.516	69.24331	5.789205	4.324714	-9.99456	1.978711	-6.41541	-4.08102	-6.72882	-3.32433	-9.54808	-5.95854	-8.34272	4
-469.766	81.9986	-6.10265	3.881257	2.268511	-3.27566	-6.27672	-7.48911	-12.964	-1.39296	-3.49052	-4.71911	-6.14937	4
-475.462	88.3095	1.065404	4.89026	-2.59628	-3.44572	-5.06224	-5.82164	-11.9138	-4.42483	-6.60151	-4.02986	-8.83224	4
-504.965	76.70156	8.250108	7.673185	4.209765	4.994709	-5.38722	-2.63306	-8.06698	-5.08798	-9.51296	-4.45827	-7.17297	4

Fig 26. The dataset

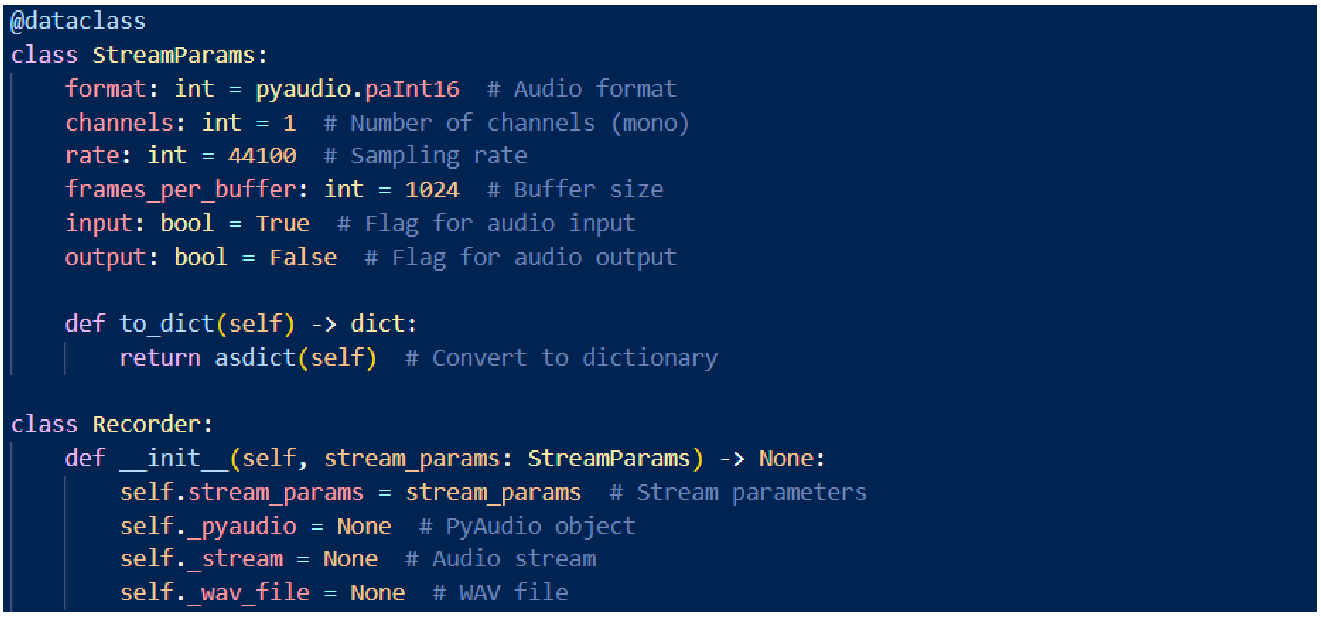
During the documentation for the realization of this data set, I also tried the option in which, after taking over the data set, it is normalized.

Normalization is a specific form of feature scaling that transforms the range of features to a standard scale. Normalization and, for that matter, any data scaling technique is required only when your dataset has features of varying ranges. Normalization encompasses diverse techniques tailored to different data distributions and model requirements.

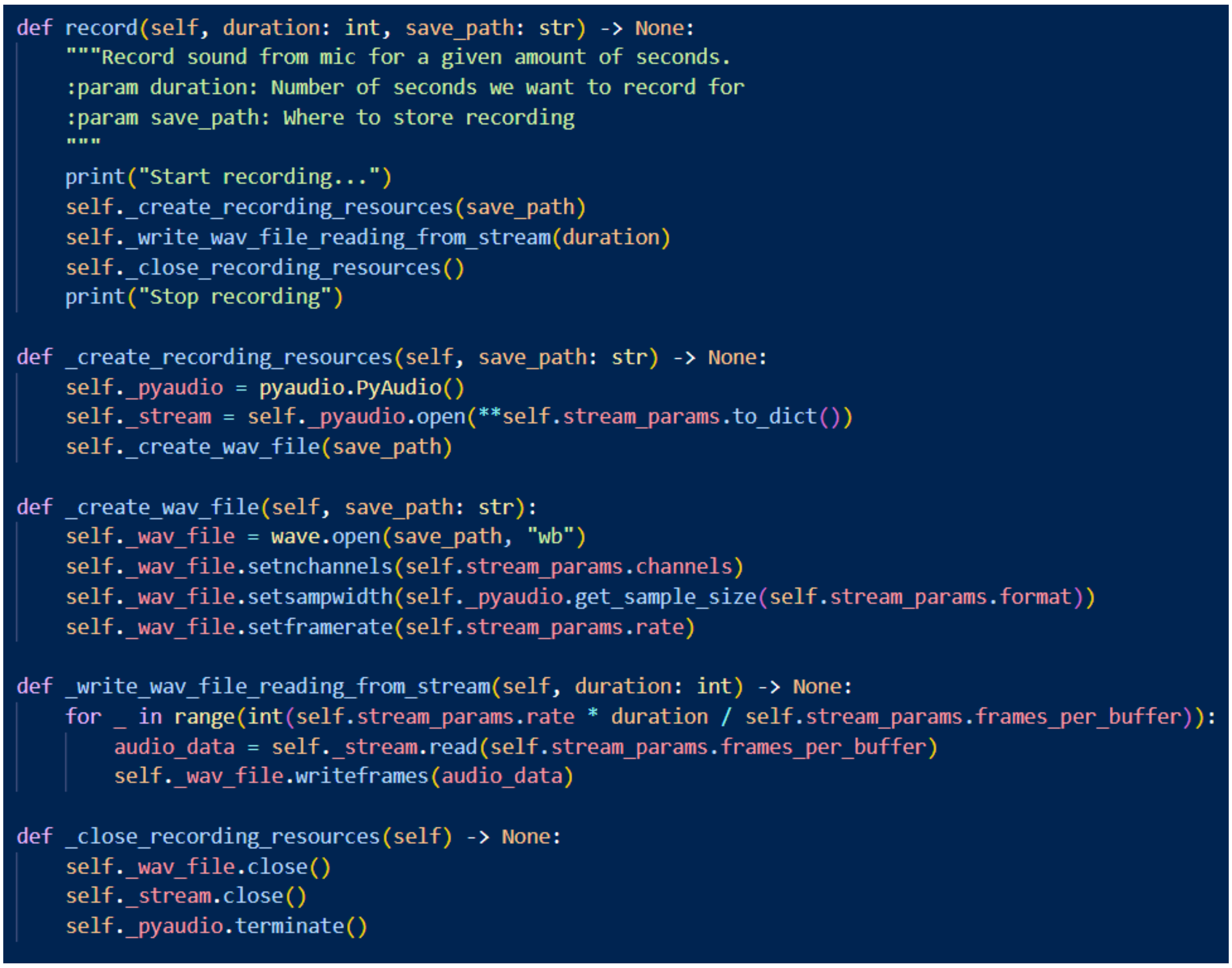
Normalized data enhances model performance and improves the accuracy of a model. It aids algorithms that rely on distance metrics, such as k-nearest neighbors or support vector machines, by preventing features with larger scales from dominating the learning process.

5.1.2 Taking over the audio signal

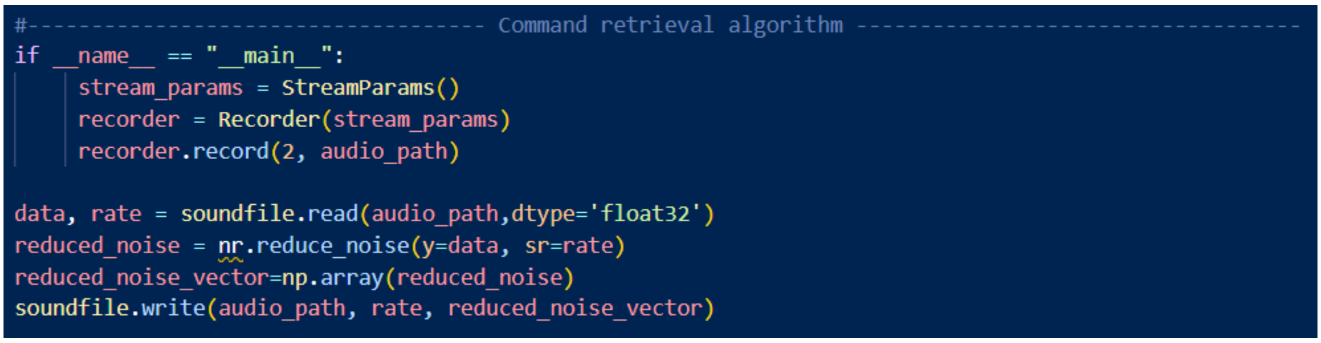
To take the command given by a certain person, the laptop microphone was used. The command given by the speaker will be saved as a wav file that will be processed later. From a software point of view, the process of recording the command over a certain amount of time and saving it in a waveform audio file format is based on a series of functions made in Python, as follow:



```python
@dataclass
class StreamParams:
    format: int = pyaudio.paInt16  # Audio format
    channels: int = 1  # Number of channels (mono)
    rate: int = 44100  # Sampling rate
    frames_per_buffer: int = 1024  # Buffer size
    input: bool = True  # Flag for audio input
    output: bool = False  # Flag for audio output

    def to_dict(self) -> dict:
        return asdict(self)  # Convert to dictionary

class Recorder:
    def __init__(self, stream_params: StreamParams) -> None:
        self.stream_params = stream_params  # Stream parameters
        self._pyaudio = None  # PyAudio object
        self._stream = None  # Audio stream
        self._wav_file = None  # WAV file
```



```python
def record(self, duration: int, save_path: str) -> None:
    """Record sound from mic for a given amount of seconds.
    :param duration: Number of seconds we want to record for
    :param save_path: Where to store recording
    """
    print("Start recording...")
    self._create_recording_resources(save_path)
    self._write_wav_file_reading_from_stream(duration)
    self._close_recording_resources()
    print("Stop recording")

def _create_recording_resources(self, save_path: str) -> None:
    self._pyaudio = pyaudio.PyAudio()
    self._stream = self._pyaudio.open(**self.stream_params.to_dict())
    self._create_wav_file(save_path)

def _create_wav_file(self, save_path: str):
    self._wav_file = wave.open(save_path, "wb")
    self._wav_file.setnchannels(self.stream_params.channels)
    self._wav_file.setsampwidth(self._pyaudio.get_sample_size(self.stream_params.format))
    self._wav_file.setframerate(self.stream_params.rate)

def _write_wav_file_reading_from_stream(self, duration: int) -> None:
    for _ in range(int(self.stream_params.rate * duration / self.stream_params.frames_per_buffer)):
        audio_data = self._stream.read(self.stream_params.frames_per_buffer)
        self._wav_file.writeframes(audio_data)

def _close_recording_resources(self) -> None:
    self._wav_file.close()
    self._stream.close()
    self._pyaudio.terminate()
```



```python
#----------------------------------- Command retrieval algorithm -----------------------------------
if __name__ == "__main__":
    stream_params = StreamParams()
    recorder = Recorder(stream_params)
    recorder.record(2, audio_path)

data, rate = soundfile.read(audio_path,dtype='float32')
reduced_noise = nr.reduce_noise(y=data, sr=rate)
reduced_noise_vector=np.array(reduced_noise)
soundfile.write(audio_path, rate, reduced_noise_vector)
```

The "Record" class belongs to the Python programming language and contains a number of useful methods for recording a word and saving it to a specific path on the laptop. This class uses the properties of the pyAudio module, which is a library of the Python programming language.

This library interacts with the drivers of operating systems, such as Windows, macOS and Linux, through APIs specific to these systems. These APIs offer the possibility to manage the audio devices of the device being worked on, to configure its sampling parameters (sampling frequency and sample size), but also to record and play sound.

A first step in receiving the signal is initializing the pyAudio module, which will facilitate access to the device's recording functions. This initialization is done via the _create_recording_resources method. Also within this method the aperture is set a stream, through the same pyAudio module, based on which the audio signal is recorded.

Once the system's audio functions are initialized and the stream is active, the voice signal is recorded and saved to a wav file at a predefined path. The creation of this file and the folder in which it is saved is done with the method _create_wav_file, where a configuration of the wav file also takes place, regarding the sampling rate, the size of each sample and the number of channels of this file.

The recording of the voice signal is done by means of the method _write_wav_file_reading_from_stream. This method is responsible for reading the data from the audio signal and writing it to the wav file.

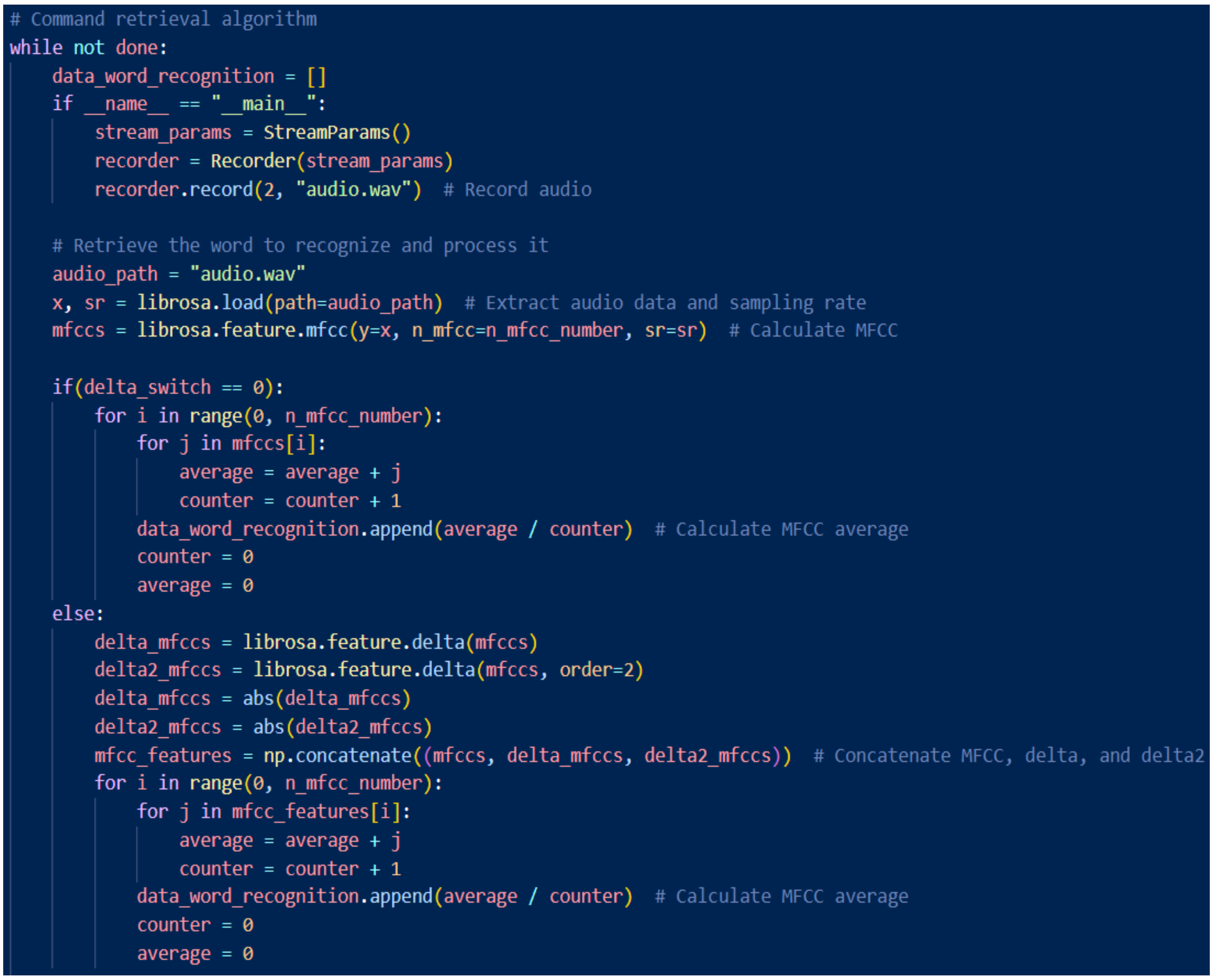
The _close_recording_resources method is used after recording the voice command and saving it, in order to save and close the wav file, close the previously started stream and release all the laptop's audio resources used by pyAudio.

All the methods mentioned above are subsequently called in another method, namely the record method. By calling this method, all other functions will be executed, thus recording the audio signal and saving it to a wav file.

5.1.3 Processing the recorded audio signal

We have reached the stage where the data set is created and the signal is recorded. What we're going to do next is process the voice command, just like we did when we created the dataset. So, in the program I made a part of the code where the wav file is taken from the path where it was saved and the function is applied to obtain the cepstral coefficients of the mel frequency. As with the other processed audio signals, 13 cepstral coefficients were created, and for each individual coefficient the mean value was calculated, so that the coefficient was represented by a single value, not a vector of values.

The code that performs all of the above functionality is as follows:



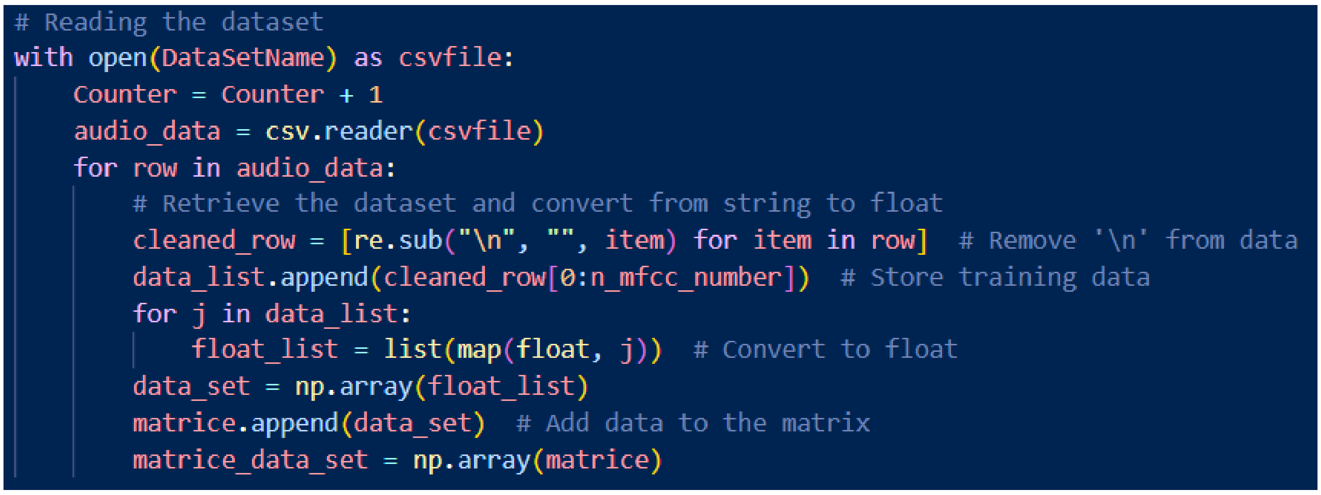
```python
# Command retrieval algorithm
while not done:
    data_word_recognition = []
    if __name__ == "__main__":
        stream_params = StreamParams()
        recorder = Recorder(stream_params)
        recorder.record(2, "audio.wav")  # Record audio

    # Retrieve the word to recognize and process it
    audio_path = "audio.wav"
    x, sr = librosa.load(path=audio_path)  # Extract audio data and sampling rate
    mfccs = librosa.feature.mfcc(y=x, n_mfcc=n_mfcc_number, sr=sr)  # Calculate MFCC

    if(delta_switch == 0):
        for i in range(0, n_mfcc_number):
            for j in mfccs[i]:
                average = average + j
                counter = counter + 1
            data_word_recognition.append(average / counter)  # Calculate MFCC average
            counter = 0
            average = 0
    else:
        delta_mfccs = librosa.feature.delta(mfccs)
        delta2_mfccs = librosa.feature.delta(mfccs, order=2)
        delta_mfccs = abs(delta_mfccs)
        delta2_mfccs = abs(delta2_mfccs)
        mfcc_features = np.concatenate((mfccs, delta_mfccs, delta2_mfccs))  # Concatenate MFCC, delta, and delta2
        for i in range(0, n_mfcc_number):
            for j in mfcc_features[i]:
                average = average + j
                counter = counter + 1
            data_word_recognition.append(average / counter)  # Calculate MFCC average
            counter = 0
            average = 0
```

5.1.4 Retrieving values from the data set

Before comparing the characteristics of the recorded signal with the data set values, it is necessary to fetch these values and store them in an array. So, the csv file in which the data set is written is accessed and the values are extracted, being converted from string to float. After the conversion takes place, the list of values is transformed into a vector of data, which will be useful when comparing the features of the voice command with the values in the data set.

The representative classes for the values in the csv file will also be converted from the string data type to the int data type, and then stored in a value vector.

The code used to retrieve the values from the csv file is as follows:



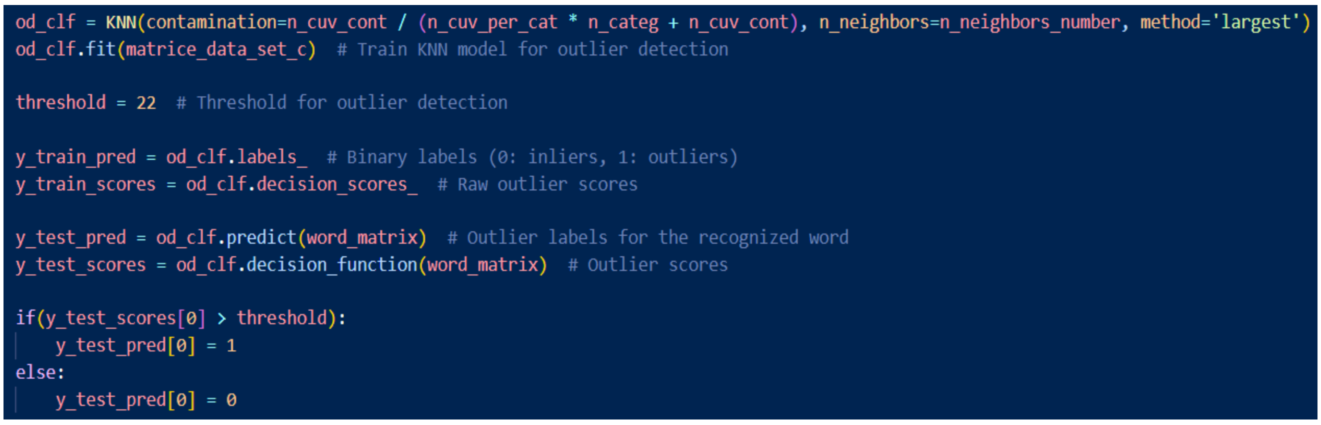
```
# Reading the dataset
with open(DataSetName) as csvfile:
    Counter = Counter + 1
    audio_data = csv.reader(csvfile)
    for row in audio_data:
        # Retrieve the dataset and convert from string to float
        cleaned_row = [re.sub("\n", "", item) for item in row]  # Remove '\n' from data
        data_list.append(cleaned_row[0:n_mfcc_number])  # Store training data
        for j in data_list:
            float_list = list(map(float, j))  # Convert to float
        data_set = np.array(float_list)
        matrice.append(data_set)  # Add data to the matrix
        matrice_data_set = np.array(matrice)
```

5.1.5 Anomaly detection

By "anomalies" we must understand those words that the voice recognition system must not identify. Basically, if the algorithm has to recognize only four words (Forward, Backward, Left, Right), as in the case of this work, any other word spoken by the one who will give a command to the robot, will be classified in a separate class from to the algorithm, and the robotic platform will not move.

The principle on which this anomaly detection algorithm works is as follows: words are classified into two categories: not-outlier and outlier. The input data is compared with the data set and thus a threshold is established, which makes the difference between the two categories. If the value of the input data is lower than the respective threshold, the word is classified as not-outlier, and if it is higher, it will be assigned to the outlier category.

The algorithm that performs the detection of these anomalies is as follows:



```
od_clf = KNN(contamination=n_cuv_cont / (n_cuv_per_cat * n_categ + n_cuv_cont), n_neighbors=n_neighbors_number, method='largest')
od_clf.fit(matrice_data_set_c)  # Train KNN model for outlier detection

threshold = 22  # Threshold for outlier detection

y_train_pred = od_clf.labels_  # Binary labels (0: inliers, 1: outliers)
y_train_scores = od_clf.decision_scores_  # Raw outlier scores

y_test_pred = od_clf.predict(word_matrix)  # Outlier labels for the recognized word
y_test_scores = od_clf.decision_function(word_matrix)  # Outlier scores

if(y_test_scores[0] > threshold):
    y_test_pred[0] = 1
else:
    y_test_pred[0] = 0
```

5.1. 6 Application of the KNN classification algorithm

The main purpose of audio signal processing is to identify the best class into which the voice command can be assigned. To achieve this distribution, the k-NN (k-nearest neighbors) classification algorithm was used, the principle of which is to calculate the distance between the queried point and other points in its vicinity. The algorithm will classify the queried point into the class that contains a larger number of neighbors, which are at a close distance to the object to be assigned.

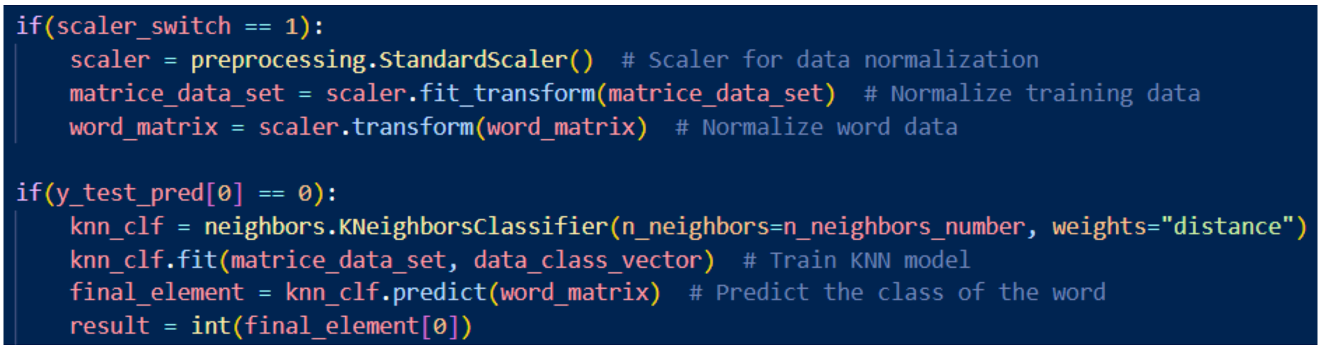
The data vector containing the voice command features has only 13 elements, which represent the 13 cepstral coefficients. In order to be able to process with the KNN algorithm, it is necessary to equalize its size with the size of the data set matrix. Starting from this idea, I made the data vector of the word to be recognized a matrix containing the 13 cepstral coefficients and the rest null values.

As we mentioned in the sub-chapter "KNN Algorithm (K-Nearest Neighbors)", KNN algorithms assign an object to a certain class by calculating the distance between it and the points in its vicinity. In addition to the distance criterion, the weight criterion can also be applied, which involves assigning an equal value to all neighbors. The application of the distance criterion requires a value equal to the inverse of the distance between the object and another point. The value assigned to a certain representative of a class is the greater the closer it is to the element to be classified, and when the decision is made as to whether the object belongs to a certain class, the elements that have high value weights can have a stronger influence in decision making.

In the software program, the classification criterion based on the calculation of the distance from the object to the elements in its vicinity was used. The calculated distance was made vis-à-vis 3 elements, located in the vicinity of the object whose classification is desired.

Data training is done with a method specific to the Python language, namely the **fit** method. This method has two parameters: the dataset and the vector containing the classification classes. When this function is called, the process of learning and assigning data to existing classes takes place. Basically, at this moment the classes with elements are created and populated, which will be the representatives of these classes at the time when it is desired to classify certain data. At this point, the entire dataset is assigned to classes, and all that's left to do is feed the data we want to classify into a method called predict , which predicts the class labels for the input data.

The code that performs the training and prediction of the classes is as follows:



```python
if(scaler_switch == 1):
    scaler = preprocessing.StandardScaler()  # Scaler for data normalization
    matrice_data_set = scaler.fit_transform(matrice_data_set)  # Normalize training data
    word_matrix = scaler.transform(word_matrix)  # Normalize word data

if(y_test_pred[0] == 0):
    knn_clf = neighbors.KNeighborsClassifier(n_neighbors=n_neighbors_number, weights="distance")
    knn_clf.fit(matrice_data_set, data_class_vector)  # Train KNN model
    final_element = knn_clf.predict(word_matrix)  # Predict the class of the word
    result = int(final_element[0])
```

In the final_element variable, the label of the class to which the recorded voice command will belong is stored. The above code also contains a series of conditions, which check which label was returned by the classification algorithm. The class label will be displayed on the GUI, so that the result obtained by the classification algorithm is known.

5.2 <u>Python programming language</u>

Programming languages are the way we can communicate and interact with computers or advanced technological systems. Whether we are talking about mobile applications, websites, intelligent robots or smart home, these programming languages are the basis of their realization. In essence, they are nothing more than a "vocabulary" that digital systems understand. They are represented by a series of rules and instructions that computing units can understand and execute. The number of these programming languages is very large nowadays, but we will present only two of them, because these two were also used in the development of the robotic system.

Python programming language is one of the most used and desired programming languages in the IT industry today. Its widespread use is due to the fact that it was created so that anyone can learn and use it, whether they are a beginner or an experienced programmer. In addition to this, what can be created with this language falls into a very wide range of applications. From simple functions found in structured programming, to the creation of platforms or websites, from the simplistic automation of some systems, to the programming of the most powerful and intelligent robots, Python is the language that allows them to be created easily. However, the ease with which this language can be used is not reduced to the syntax, but to its complexity.

Python comes bundled with a multitude of APIs (Application Programming Interface) and libraries, which give the developer the facility to create everything he wants with it. Suppose the programmer wants to develop an application that recognizes video or audio signals, using the audio-video systems of the computer on which the application is developed. In this situation, it is no longer necessary to create programs that take information from the laptop's microphone or camera, because this is already done in a number of libraries, which include APIs that python works with. The programmer's job will just be to identify the libraries corresponding to the respective APIs and include them in his project, using the methods he needs. The most common and often used APIs are NumPy, ploty, pandas, SciPy, Matplotlib.

In addition to Python, MicroPython was later created, which is not much different from the standard version of this language, but is designed to be used in working with smaller computing systems, such as microcontrollers and IoT devices (Internet of Things).

Capitolul VI. Concluzii

6.1 Rezumat

In this paper, we conducted an extensive investigation into how speech recognition systems work, the algorithms used for these systems, and the impact they can have on all of humanity. The concept and idea behind an application that aims to identify the words of the human voice turned out not to be so difficult, but has a clear and well-structured process. This process involves a series of steps, program with the creation of the data set and the membership classes, the processing of this data and the voice signals that should be recognized with the help of MFCC algorithms, and then the classification of the voice command into a certain class. The user, through the application, can say any word he wants as a command, but it is programmed to recognize a limited range of words, five in number, all in English: "FORWARD", "BACKWARD", "LEFT", "RIGHT", "STOP". Any other word spoken by the user will be routed to a class called "Other". During the course of the project, several tests were carried out on the speech recognition system, which made the project reach a high maturity of its functionality. A practical application of this application could be a miniature example of a wheelchair, used by people with locomotor disabilities. Such a system for a person who cannot use his limbs would be useful, giving him the possibility of moving to desired places. Considering the level of complexity of the project is not high, the number of words it can identify is perhaps a low one, the plan is to expand this number of words, as well as to perform more movements of the robotic platform, providing people who use it a wider range of functionality.

Capitolul VII. Bibliografie

[1]https://playtech.ro/2020/recunoasterea-vocala-e-acum-banala-dar-cand-a-luatnastere/

[2]https://koaha.org/wiki/Riconoscimento_vocale#cite_note-1
[3]http://math.etc.tuiasi.ro:81/rosu/didactic/MS%20II_Curs_Transformarea %20Fourier.pdf

[4]https://www.analyticsvidhya.com/blog/2021/06/mfcc-technique-for-speechrecognition/

[5]http://practicalcryptography.com/miscellaneous/machine-learning/guide-melfrequency-cepstral-coefficients-mfccs/

[6]https://towardsdatascience.com/how-to-apply-machine-learning-and-deep-learningmethods-to-audio-analysis-615e286fcbbc

[7] https://wiki.aalto.fi/display/ITSP/Cepstrum+and+MFCC

[8]https://www.researchgate.net/publication/221616659_Choice_of_Mel_filter_bank_in_computing_MFCC_of_a_resampled_speech

[9]https://www.oracle.com/ro/artificial-intelligence/machine-learning/what-is-machine-learning

[10] https://www.thc.ro/blog/ce-este-machine-learning/

[11] https://www.interviewbit.com/blog/cnn-architecture/

[12] https://scikit-learn.org/stable/modules/neighbors.html#classification

[25] https://towardsdatascience.com/dynamic-time-warping-3933f25fcdd

[26] https://medium.com/mlearning-ai/what-is-dynamic-time-warping-253a6880ad12

[24] https://www.youtube.com/watch?v=_K1OsqCicBY

[25] https://towardsdatascience.com/dynamic-time-warping-3933f25fcdd

[26] https://medium.com/mlearning-ai/what-is-dynamic-time-warping-253a6880ad12

[27] https://www.cs.unm.edu/~mueen/DTW.pdf [28] https://towardsdatascience.com/k-nearest-neighbors-knn-for-anomaly-detection fdf8ee160d13