uc3m | Universidad Carlos III de Madrid

MASTER'S THESIS

A robust speech-based stress detector designed for smartphone integration

Author:
Alba MÍNGUEZ SÁNCHEZ

Supervisor:
Carmen PELÁEZ MORENO

Master in Telecommunications Engineering

Signal Theory and Communications Department

February 26, 2019

UNIVERSIDAD CARLOS III DE MADRID

Abstract

Signal Theory and Communications Department

Master in Telecommunications Engineering

A robust speech-based stress detector designed for smartphone integration

by Alba MÍNGUEZ SÁNCHEZ

Unfortunately, gender-based violence has become one of the most up-to-date problems. Findings from different studies indicate a clear need to develop solutions to detect, prevent and combat this kind of situation.

BINDI, a project from the UC3M4Safety multidisciplinary team, was born with this purpose in mind. Its objective is to develop a system composed of a pendant that includes a microphone and a bracelet that captures physiological signals that are connected to an smartphone application. Therefore, these devices allow us to collect both voice and physiological measures of the victim of gender violence to analyze them and try to detect dangerous situations so as to warn a group of close people that could provide help.

The present work will develop the part of BINDI responsible for processing speech from the microphone, extracting features and generating a machine learning algorithm that decides between stress or neutral speech frames. Heart rate measures are employed to provide the labels for training the models. In addition, the effect of the different mismatched sources of the system, i.e. environmental noise and microphone distortions, will be considered with the contamination of a sample database in order to implement a system as robust as possible in real environments.

As for evaluating the performance and robustness of the system, several experiments are proposed for different conditions and speech enhancement phases, obtaining, on the one hand, accuracy values of around 90% for matched conditions; and on the other hand, maximum results of 90% accuracy for high SNR levels and 58% and 72% accuracy for low SNR, with and without using speech enhancement techniques, respectively.

Acknowledgements

Thank you, Carmen, for having counted on me for a project with such an important goal for everyone. For everything I have learned and for your guide whenever I needed it.

To my family, to make me always remember that now was the moment to do this: everything has its proper time.

To my friends, I hope you will recognize me when I have time to meet again.

Jesús, thank you for letting me mention you here again. You are still my example to follow.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The aim of this first chapter is to introduce the objectives and motivation of the master thesis that will be presented in this document, offering a global vision of the context in which it is set. It also contains a description of the main contributions and scope within this context. Finally, it details the structure of the document in order to ease as much as possible its reading.

1.1 Motivation

If we ask ourselves what biometrics, machine learning, cyberphysical systems and Internet of Things (IoT)(smartphones, wearables, smartwatches...) have in common, we would answer that they are current research topics that are evolving faster and faster every day. According to Deloitte's 2018 Global Mobile Consumer Survey [1], 65 percent of smartphone owners across 16 developed markets have used an application featuring machine learning in the past. Furthermore, many of these applications are powered or need biometric data such as voice, fingerprint and heart rate, sometimes provided by other smart devices.

However, these elements do not only stand out only for themselves, but because the combination of them can give rise to innovative solutions to detect, prevent and combat another of the most up-to-date problems: gender-based violence.

The World Health Organization presented in 2015 the first global systematic review [2] which aggregated, for the first time, global and regional prevalence estimates of two forms of violence against women: violence by an intimate partner (intimate partner violence) and sexual violence by someone other than a partner (non-partner sexual violence). This report was generated using population data from all over the world that have been compiled in a systematic way. The findings were striking:

- Overall, 35% of women worldwide have experienced either physical and/or sexual intimate partner violence or non-partner sexual violence.
- Most of this violence is intimate partner violence. Worldwide, almost one third (30%) of all women who have been in a relationship have experienced physical and/or sexual violence by their intimate partner. In some regions, 38% of women have experienced intimate partner violence.
- Globally, as many as 38% of all murders of women are committed by intimate partners.

There is a clear need to scale up efforts across several areas, both to prevent violence from happening in the first place and to provide necessary services for women experiencing violence.

1.2 The context: BINDI, a smart solution for women safety

All these are the reasons why BINDI [3], the research project of which this master thesis is a part, was born. BINDI consists of a smart solution for women safety developed and implemented by the UC3M4Safety [4] team. Specifically, the system is based on preventing and alerting about situations of danger or gender-based violence.

UC3M4Safety

UC3M4Safety multidisciplinary team was created in 2016, with researchers from different research groups in the Technology branch (Electronics Technology, Signal Theory and Communications, Telematics Engineering) and in the Social Sciences branch, with the Institute of Gender Studies of the Carlos III University of Madrid, as well as the Industrial Electronics Centre of the Polytechnic University of Madrid.

The main objective of this team was to combine the knowledge of all its researchers to use the most modern and disruptive technology in the detection, prevention and investigation of Gender Violence (GV), without losing sight of the social approach, both protection, care and recovery of victims as well as visibility of the problem and awareness of society. The idea consisted on the detection of the emotions experienced by victims of GV through intelligent sensors and machine learning algorithms, while incorporating wireless communications, body area networks and portable devices (wearable) to generate an invisible solution that would help autonomously, discreetly and immediately to women who are victims of an aggression.

In a continuous search for external funding, two European project proposals were presented in the topic "Human Factor for the Prevention, Mitigation and Investigation of criminal and terrorist acts", in subtopic 5 ("New methods to prevent, investigate and mitigate high impact domestic violence"), 2016 and 2017, within an international consortium with 5 National Police Agencies, several universities and research centres and the Leuven Institute of Criminology as project leader. A proposal was also submitted to the Gendernet Plus programme in 2018. Although these proposals have not been successful, the team continues seeking and proposing projects to fund the ongoing research.

Recently, substantial funding has been obtained from the Community of Madrid with a proposal awarded in the Synergic Projects 2018 call which also has the support of the Family and Women's Unit of the National Police and more than 10 associations of Victims of Gender Violence (VGV).

In addition, this team aligned efforts to participate in the international contest of the organization XPrize Women's Safety "Developing technology for safer societies" [5], which reached the semifinal in April 2018. In this competition, a portable, cheap

and precise solution to prevent sexual aggressions against women (BINDI) was presented.

Convinced of the enormous dimension of the social problem and the great potential impact of actions such as this, we have also submitted a patent pending to protect our conception of the technology needed to help the VGV.

Currently, we continue with the investigation of the specific problem of GV, the underlying and observable cognitive phenomena through current and future devices, its computational modeling with the most modern techniques of signal processing and machine learning and the use of communications and social networks to protect the VGV, actively participating in the redesign of protocols and protection policies in light of the facilities offered by technology. Without losing sight of the need to advance in the design and implementation of a portable, attractive and discreet solution so that it can be comfortably carried by users, it is necessary for the system to be able to detect automatically, quickly and reliably the situations of danger in the VGV; and, for this, **it is necessary to deepen the design and training of intelligent algorithms that detect emotions of fear, stress, or even panic.**

BINDI system

As shown in Figure 1.1, BINDI consists of two wearable devices and a smartphone application. One of the wearables, the bracelet, is in charge of measuring biometrics such as heart rate, galvanic skin response, temperature and provides a panic button. The second device is the pendant, which also has a panic button and a microphone that records voice from the victim. This information is collected and sent via bluetooth to the mobile phone, whose application, under certain conditions, sends an alert to a selected circle of people of the victim in order to alert them of the situation.

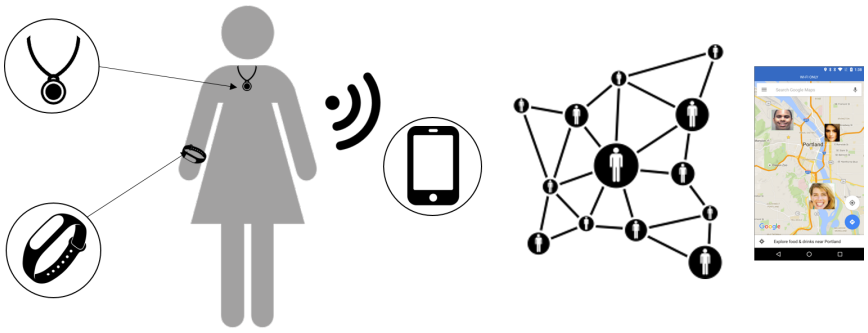

FIGURE 1.1: Bindi System

As mentioned above, this thesis is part of the research project BINDI. The following are the four big disciplines involved in the design and development of BINDI:

- **Electronics Technology**: for the development and implementation of the devices and sensory data collection. It includes the machine learning algorithm in charge of managing the bracelet data that provides the alert that switches on the microphone and starts stress detection block.

- **Signal Theory and Communications**: for
  1. processing speech from the microphone, extracting features and generating a machine learning algorithm that classifies between stress or neutral based on physiological features such as heart rate and speech. This master thesis belongs to this branch of Signal Theory and Communications discipline.
  2. identifying the victim's voice and possibly the aggressor's to be able to execute voice controlled commands and perform speaker adaptation.
  3. processing the environmental audio to detect violent acoustic events to confirm or discard the previous physiology based alert.
- **Telematics Engineering**: This area is in charge of embedding the previous areas in an smartphone with an Android system and sending alert messages to a previously configured circle of trusting people. It is also in charge of managing communications among the devices.
- **Institute of Gender Studies**: This branch is in charge of providing information on the social and real needs of women who are in gender violence situations. This helps on the development of the system to have a realistic and practical approach and to collect data to be used for training the algorithms.

1.3 Stress detection in speech signals: A preliminary study

This Master Thesis is preceded by a preliminary study on stress detection from speech signals [6] in which we tested the feasibility of a speech-based stress detection system in MATLAB and Python using the popular scikit-learn tool.

In that study, the VOCE Database[7] (see subsection 2.2.1) was explored and processed in depth adapting it to the needs of the study. In particular, stress labels were obtained from the electrocardiogram signals (ECG) provided that were transformed into consistent heart rate (HR) values in beats per minute. In addition, the database files were also classified in sets according to different quality criteria. This was useful to obtain results indicating that audio quality did affect stress detection. Experiments were also carried out distinguishing between gender (women and men) and age, with no remarkable differences.

The results of this study about stress classification based on HR values resulted in F-scores greater than 80%, which indicated that there was a strong relationship between stress in speech and heart rate, so the development of a system with this objective was feasible. Also, the experiments revealed that there was a certain speaker dependence on the stress detection system.

1.4 Objectives

The main goal of this work is to develop the stress detection system from voice signals as a part of BINDI project. This goal can be divided into three specific subobjectives:

1. To implement a recording procedure to collect a contaminated database with the purpose of evaluating the noise robustness of the stress detection system in different noise levels. In our particular case, this procedure will let us consider a proof-of-concept in which both the specific device (microphone) that provides the speech and the different real environmental noise levels will be tested in an anechoic chamber with the collection of a sample database.
2. To design and implement the stress detection system in Java to allow its integration in the smartphone application. To achieve this, some specific requirements are taken into account: first, those related with the robustness of the system in real life environments and second, those from the integration or embedding of the system in a smartphone device.
3. Test the robustness of the models obtained with the new data collected using the new implementation.

1.5 Contributions and scope

As mentioned before, BINDI is a multidisciplinary project involving several broad areas. These areas work together towards the common objectives of the project, taking into account the requirements and limitations of the others.

In this thesis, the main contributions consist in the design and implementation of a stress detection system based on speech signals and biometric measurements, capable of being integrated or embedded in a smartphone device in BINDI system in order detect and avoid, in real time, future gender violence situations.

However, for the purpose of this academic work it is important to make clear the scope of this work. Therefore, the aspects related to the study of the social needs of victims of gender violence, the choice of the different devices of the system, the communication protocols and the integration or embedding of the Android application in the smartphone device are outside the scope of the project.

1.6 Document Structure

Next, a description of the content of each chapter of this report will be briefly presented in order to facilitate its understanding.

Chapter 1 consists in introducing the context of the study: the research project it belongs and preliminary research. Motivation and objectives are also explained, as well as the author's contributions.

Chapter 2 includes the state of the art, that is, the investigation of previous works similar or related to the present one. Its results, technologies, methods and resources

used in them will be detailed, as well as all the necessary concepts to understand the technical implementation of the system.

The design and implementation of the system are presented in Chapter 3. Subsequently, the proposed solution is detailed and justified on the objectives listed above.

The purpose of Chapter 4 is to present the experiments carried out during the evaluation of the system as well as the results obtained from them.

Chapter 5 presents the planning and budget followed during the development of the project.

In Chapter 6, the corresponding conclusions are extracted from the previous results as well as from the general project, in order to evaluate the fulfilment of the objectives and propose future lines of work.

A breakdown of the results associated with the experiments in Chapter 4 shall be given in Appendix A so that they can be consulted if further details are required.

In Appendix B the user manual for the complete execution of the system is collected.

Chapter 2

State of the art

This chapter includes the state of the art, that is, the investigation of previous works similar to the present one. These related projects will be analysed in several aspects: results, technologies, methods and resources used in them. Also, this chapter contains all the necessary concepts to understand the technical implementation of the present work.

2.1 Biometrics

Biometrics is the science of analysing physiological or behavioural characteristics of the human body specific to each individual, usually for authentication tasks (identification, verification), among others [8]. There are multiple types of sources or biometric measurements, categorized in physiological (fingerprint, facial, iris, voice, DNA, Galvanic Skin Response (GSR), voice, electrocardiogram (ECG) signals...) and behavioural (signature, keystroke dynamics...).

In the case of this work, physiological measurements such as voice or speech and heart rate values derived from ECG signals will be used in different ways as information sources to detect or classify stress or neutral from the individual.

2.2 The Machine Learning problem

Machine learning (ML) is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience. It mainly focuses on the development of computer programs or algorithms that can access data and use it learn for themselves.

This process of *learning* begins with observations, called training data, in order to look for patterns in it and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly.

In this work, ML will be the basis for solving our problem. As we will see all along the chapter, machine learning techniques are used to solve a wide range of problems related to speech systems (speech recognition, speaker identification, emotion classification...). In our specific case, we will focus on *supervised* machine learning algorithms, that is, methods that apply what has been learned in the past to new data using *labeled* examples, to predict whether speech contains stress or not.

Figure 2.1 shows the block diagram of this common pipeline for solving a supervised machine learning problem in the context of speech technologies.

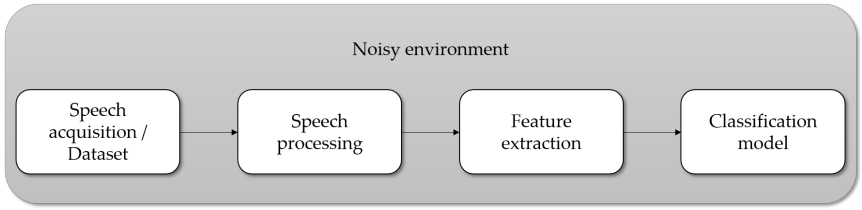

FIGURE 2.1: Common pipeline for a machine learning problem on a classification task.

Therefore, there are 4 main challenges to be considered: speech acquisition and dataset conditioning, speech pre-processing to clean or prepare the input signal, features extraction and the classification itself. Some of the most common techniques used in the literature to implement them will be explained below. In chapter 3, a detailed in depth description of our the current solutions will be exposed.

2.2.1 Stress Databases

Ververidis and Kotropoulos [9] include in their article a table with information from a total of 64 existing databases related to emotions, stress and various biometric measures. However, there are few that can be used for a problem like the one presented in this thesis, either because data is taken in artificial environments, because of the lack of availability and documentation, or simply because it is collected for other purposes and, therefore, it no longer makes sense to use them in other applications.

Below we present only some of the most well-known voice stress databases used in projects related to the detection of stress or emotions. Before defining these databases, it is important to point out the different types of stress under which speech may be. Some of the most present in the databases are [10]:

- **Physical task stress**: Physical stress includes factors such as G-force experienced in aircraft cockpits, stress experienced due to high speeds in racing cars etc.
- **Lombard effect**: although this is not a type of stress itself, it is commonly called stress because it refers to the variations - in terms of pitch, intensity, duration, formant location - in speech production due to a noise environment.
- **Cognitive stress**: This stress is related to situations where demanding tasks such as driving or giving a speech are required. This is the type of stress this work is focused on.

SUSAS: Speech Under Simulated and Actual Stress

This database[11] was compiled for the purpose of formulating and implementing speech algorithms under stress and noise. One of the characteristics of the recordings is that the voice is recorded under stress. both simulated and real. For this reason and because of the great variety of data this is the database most commonly

used in the study of the variability of voice production under conditions of stress and tension.

SUSAS divides data into five different domains including (i) psychiatric analysis data (speech under depression, fear, anxiety), (ii) different conversation styles (angry, calm, fast, high, slow, low), (iii) recordings during simple tasks, high workloads and noise situations (Lombard effect), (iv) files with double response between computer and user, and (v) audios during the performance of tasks to combat fears (G-force, Lombard effect, noise).

UT-Scope Corpus

UT-Scope Corpus [10] contains speech data under 3 types of stress: Lombard effect, cognitive stress and physical stress. Depending on the type of stress, the data have different characteristics:

Regarding the Lombard effect, speech under three noise types at three levels is recorded: Pink noise (PNK), large crowd noise (LCR) and noise in a car travelling at 65 mph on a highway (HWY). Speech samples from 59 speakers were collected. Noise presentation levels varied from 65 dB-SPL to 90 dB-SPL, representing 3 levels for each of the 3 noise types.

As for the cognitive stress, speech samples from 60 speakers while driving a car-driving simulator that requires extensive concentration were collected. Also, heart rate and blood pressure were also recorded during the task.

On the other hand, speech undar physical stress is collected while a person operates a stair stepper. Video and biometric data are also recorded for this task as well. Speech collection from 60 speakers was compelled.

VOCE Corpus

VOCE is the dataset that we will use in our study [7]. The last updated version of this dataset includes a total of 135 voice recordings that result from a set of 45 students (21 men, 17 women and 7 unidentified) from the University of Porto, with ages between 19 and 49 years.

These voice files (.wav format) correspond to three different recording settings: *pre-baseline*, reading a standard text at least 24 hours before the event in public (Public Speaking, PS); *baseline*, reading the same text as in the *pre-baseline* setting approximately 30 minutes before the PS; and *recording*, free speaking from a public event of free duration.

Together with these audio files, 117 files (.xml format) are collected, each of them matching a voice file, containing 2 measures to estimate the heart rate (Heart Rate, HR). These measurements, taken with the Zephyr HxM BT2 device, are (i) Z_{ecg} representing an averaged and filtered HR value and (ii) Z_{ts} values that refer to the instants of time in which R peaks (shown in Figure 2.2 occur in the electrocardiogram obtained with the apparatus, measured with an internal clock of 16 bits. Each of

these values is accompanied by the Universal Time Coordinated (UTC) corresponding moment.

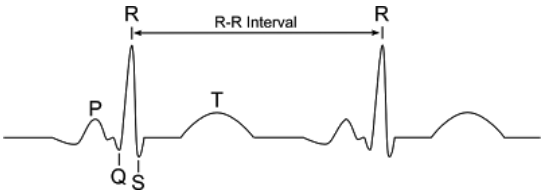

FIGURE 2.2: R peaks in ECG signal [12]

Additionally, the database has a metadata file (xml format) that includes data on gender, age, health information, experience in public speaking, STAI[13] test scores and information about the quality of the recordings (energy level, saturation ...). Unfortunately, this is only provided for 38 out of the 45 individuals in the database. It is worth noting that the database only gathers complete information[1] from 33 individuals.

2.2.2 Speech processing

As shown in Figure 2.1, one of the tasks to solve in the pipeline is speech processing. As speech systems can be used in a variety of environments, we must make sure their performances is maintained at the different noise levels. As we will see in more detail in subsection 2.3.3, noise robustness is one of the classical problems to face when designing a speech system. Although not the only one, signal processing, or *speech enhancement*, is one of the well-known techniques to combat this.

Speech enhancement is the technique that focuses on improving quality (intelligibility and/or overall perceptual quality) of speech signal [14]. As mentioned before, speech signal gets degraded because of the noisy environment in which it is framed, so it is necessary to enhance the corrupted signal by reducing noise. Usually, it is assumed that the noise is additive and that noise characteristics change very slowly as compared to the signal. This is the underlying assumption in speech enhancement methods.

2.2.3 Feature Extraction

Feature extraction methods are those algorithms and techniques that aim at computing a set of feature vectors able to provide a compact representation of the most relevant aspects of the input data to the task to solve, in our case, determining if there is stress in the speech signals.

Here follow some of the best known features extracted from speech for stress detection systems or speech/audio systems in general.

[1] In this work, complete means to have the 3 audio files and its corresponding HR values

Mel Frequency Cepstral Coefficients (MFCC)

MFCCs are one of the best-known sets of coefficients in speech systems. These were designed taking into account both human phonation and aural perception. As usual in speech processing, the MFCC calculation is performed on a short-term basis in which the operations shown in Figure 2.3 are performed.

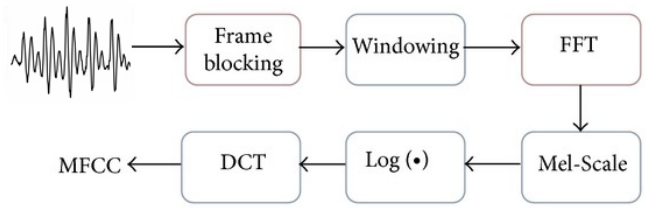

FIGURE 2.3: Process to calculate MFCC from a signal [15] where FFT means Fast Fourier Transform and DCT is Discrete Cosine Transform

Although a large number of coefficients can be calculated, it is customary to only take the first 13 into account. As a matter of fact, the 0th coefficient can be regarded as the average energies of the frame being analyzed [16].

Fundamental frequency and pitch

These are some of the most common features when designing a system within speech technologies, specially in speaker identification problems. Although they are often times used indistinctly, fundamental frequency f_0 and pitch are not exactly the same.

On the one hand, fundamental frequency (f_0) is a physical property of sound (in the case of speech, the number of glottal pulses in a second) and it is measured in Hz. On the other hand, pitch is a perceptual quality of frequency (i.e. the way our auditory system perceives different frequencies). f_0 and pitch are not identical, but the term 'pitch' is often used when talking about measured frequencies for simplicity.

In addition to this, it is interesting to know that tone and intonation are both achieved through changes of f_0. Tone refers normally to the lexical level - tonal languages use f_0 changes to differentiate word meaning. Intonation is used at the phrase level to create supra-lexical effects (e.g. differentiating between a declarative sentence and a question).

This feature will be useful in our system for two reasons: (1) to take into account speaker dependence and (2) to ease future speaker identification implementation in BINDI system.

Root Mean Square (RMS)

RMS is a good feature to measure the power of a signal, in terms of amplitude. This value is calculated by summing the squares of each sample and dividing this by the number of samples of the corresponding window or signal.

$$RMS = \sqrt{\sum_{n=1}^{N} x(n)^2}$$

Energy or power of the signal is a classical feature in emotions or stress classification, as will see in section 2.4.

Zero-Crossing Rate

In the context of discrete-time signals, a zero crossing is said to occur if two consecutive samples have different algebraic signs. The rate at which zero crossings occur is an indication of the frequency content of a signal. The zero-crossing rate is a measure of number of times in a given time interval or frame the amplitude of the speech signals passes through a value of zero [17].

This feature will be useful to discriminate between voiced and unvoiced parts of speech [18], given that zero crossing rates are low for voiced parts and high for unvoiced part whereas the energy is high for voiced portions and low for unvoiced.

Linear Predictive Coding (LPC)

Although not used for the purpose of our system, LPC is one of the classical state-of-the-art analysis used in speech recognition systems to extract the relevant information that represents the signal. The fundamental idea of the linear predictive coding is that a given speech sample can be estimated as a linear combination of the past speech samples.

This analysis is also usually used in speech processing for spectral envelope estimation. To do this, LPC analysis simplifies vocal tract model (speech production system) as a source-filter model. In this way, the obtained LPC coefficients of the filter describe the formants. Formants are the peaks of the spectral envelope correspond to the resonant frequencies [19], as shown in Figure 2.4.

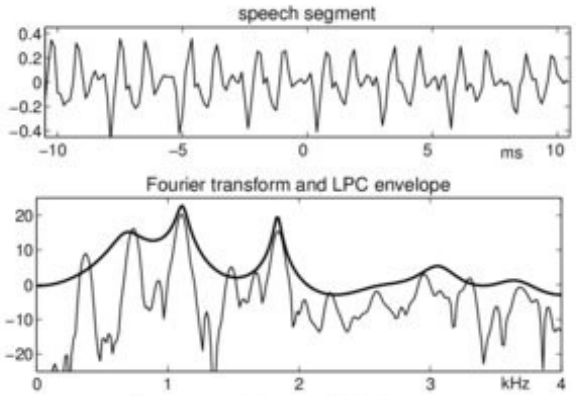

FIGURE 2.4: LPC analysis and formants

Teager Energy Operator (TEO)

New features derived from the nonlinear Teager Energy Operator (TEO) are getting more and more importance in speech processing, even in noisy environments [20]. The idea behind using TEO instead of the commonly used instantaneous energy, is to take advantage of the modulation energy tracking capability of the TEO. This leads to a better representation of the formant information in the feature vector [21].

As presented in section 2.4, TEO features is also used in emotion or stress classification systems.

2.2.4 Classification models

Gaussian Mixture Model (GMM)

Gaussian Mixture Models (GMMs) have been used extensively in speech processing. The reasons are that: (1) univariate Gaussian densities have a simple and concise representation, depending uniquely on two parameters, mean and variance, and (2) the Gaussian mixture distribution is universally studied and its behaviors are widely understood.

A Gaussian mixture model is a probabilistic model that assumes the data instances are generated from a mixture of a finite number of Gaussian distributions with unknown parameters, as shown in Figure 2.5. In principle, GMM can approximate any probability density function to an arbitrary accuracy as long as the proper number of Gaussians is provided.

In a classification problem, a GMM will be generated for the training samples of each class. After that, new samples will be classified as the class associated to the model which maximizes the likelihood.

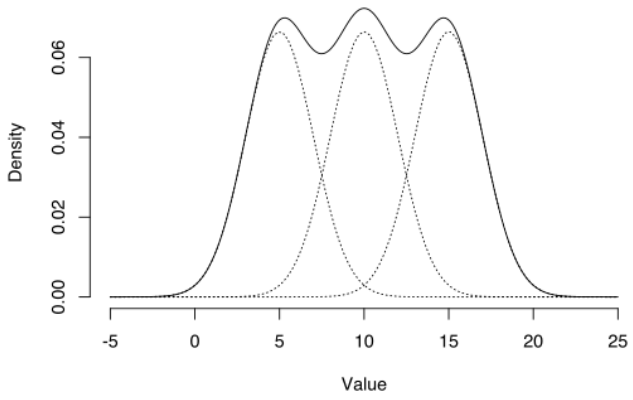

FIGURE 2.5: GMM diagram

Hidden Markov Model (HMM)

Hidden Markov Models (HMMs) provide a simple and effective framework for modelling time-varying vector sequences. As a consequence, HMMs are used in speech recognition because a speech signal can be viewed as a piecewise stationary signal or a short-time stationary signal. In a short time-scale (e.g., 10 milliseconds), speech can be approximated as a stationary process.

For example, if we consider a phoneme as the acoustic unit, each phoneme q will be represented by a continuous density HMM of the form illustrated in Figure 2.6 with transition probability parameters a_{ij} and output observation distributions $b_j(y_k)$. In operation, an HMM makes a transition from its current state to one of its connected states every time step. The probability of making a particular transition from state s_i to state s_j is given by the transition probability a_{ij}. On entering a state, a feature vector is generated using the distribution associated with the state being entered, $b_j(y_k)$ [22]. This results in the observation sequence of phones, in this case.

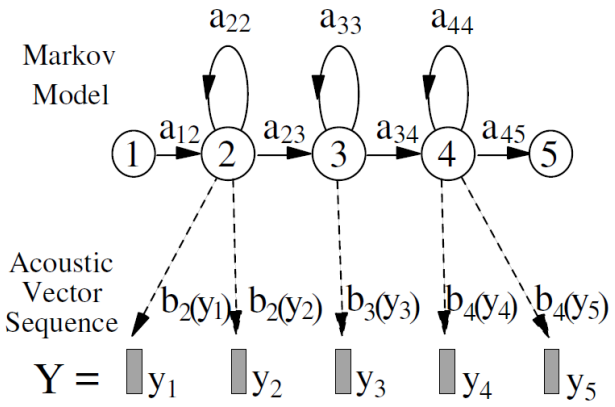

FIGURE 2.6: HMM diagram

Although this model is usually employed for speech recognition, in classification problems can be addressed as well as GMM. In this case, a HMM model would be trained for each class, having the corresponding gaussian mixture in each state s_i.

Multi-layer Perceptron (MLP)

Multilayer Perceptron (MLP) is one of the most popular network architectures for both classification and regression. MLP is a network (Artificial Neural Network, ANN) which is usually composed of several layers of nodes with unidirectional connections, often trained through backpropagation. The MLP network learning process consists of data samples that form the N-dimensional input vector x and the desired output vector M-dimensional d, called target. By processing the input vector x, the MLP produces the output signal vector $y(x, w)$ closest to the desired one by adjusting the adapted weights vector w, as reflected in Figure 2.7.

In our particular case, input vector will consists of the several speech features extracted and, therefore, output vector will be a single value representing the predicted class 0 or 1, that is neutral or stress.

The MLP learning algorithm is based on the minimization of the error function defined in the learning set (x_i, d_i) for $i = 1, 2, 3, \ldots N$ using, for example, the Euclidean standard Equation 2.1. Minimizing this error leads to optimal weight values.

$$E(W) = \frac{1}{2} \sum_{n=1}^{N} \|y(x_i, w) - d_i)\|^2 \tag{2.1}$$

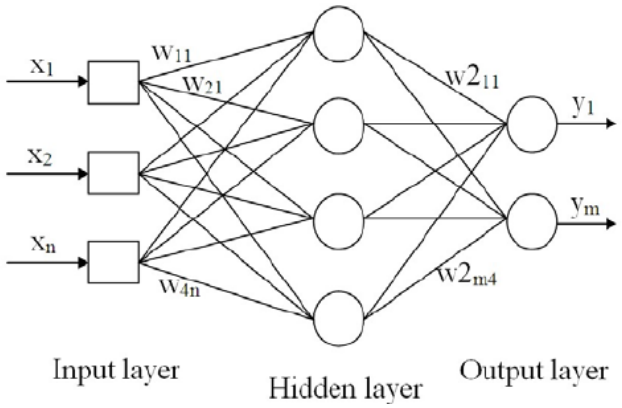

FIGURE 2.7: MLP diagram

2.3 Noise robustness

As shown in Figure 2.1, real environments are characterized by the presence of noise. Noise, regardless of its origin, is always a drawback which affects most of the speech processing systems by generating mismatched conditions which translates in a degradation of performance and quality [23]. This means that it is necessary to develop techniques for improving noise robustness.

2.3.1 Mismatched sources

Depending on the origin of the mismatched cause, we can distinguish two categories:

- **External**
  - **Environmental noise**: It consists of the noise that acoustic environment produces.
  - **Transmission channel**: it includes channel noise, transmission errors, frame/packet losses...
  - **Reverberation** is created when the signal is reflected on the surfaces or objects from rooms or cabins.

- **Speaker**
  - **Intraspeaker variability**: It includes speaking styles, emotion, Lombard effect or stress as individual's own characteristics.
  - **Interspeaker variability**: It comprises physiological aspects, dialect...

2.3.2 Model of acoustic environment

Considering the previous mismatched sources, we can define a model of acoustic environment:

$$y(t) = s(t) * h(t) + n(t)$$

where:

- $y(t)$ represents the noisy speech signal
- $s(t)$ is the clean speech signal
- $h(t)$ represents the transmission channel modelled as convolutional noise
- $n(t)$ is the background/environmental modelled as additive noise

In our case, as we will describe in chapter 3, reverberation will not be included in the model above due to the fact that recordings will be done inside an anechoic chamber.

2.3.3 Approaches

The field of noise robustness is one of the widest areas in speech technologies and, nowadays, there are multiple solutions and techniques to face this problem. If we look at one of the classic books in literature, Techniques for Noise Robustness in Automatic Speech Recognition [24], we can classify the different approaches into three main categories:

Signal enhancement

As mentioned in subsection 2.2.2, signal enhancement or speech processing is one of the approaches to face noise robustness problem. Speech enhancement aims to improve speech *quality* (clarity, intelligibility or compatibility with some other method in speech processing) by using various algorithms.

Some of this algorithms are based on multichannel mixtures and the use of microphone arrays [25], which are out of the scope of this project. However, there are also several techniques of our interest focused on acoustical signal enhancement that can be grouped into the following:

- **Voice Activity Detection (VAD):** To improve the robustness of speech systems, the detection of talk spurts is of critical importance. This can be accomplished by the model itself, for instance, by including a silence or a background noise model. In this way, the model can deal with roughly end-pointed utterances that may still contain long noise-only segments.

However, in many cases, it is more practical to evaluate the voice activity to exclude noise-only segments for further processing [26]. As a result, a large variety of different approaches are available, which are usually tailored to their specific types of application [27].

In our case, VAD algorithm will detect voice activity at frame level based on a dynamic noise estimation all along the signal, as it will be presented in chapter 3.

- **Noise Power Spectrum Estimation:** Most noise-reduction algorithms require an estimate of the background noise power. In most systems, short utterances are processed as a whole and the noise power spectral density of a particular utterance is obtained from the first few signal frames which are assumed to be noise only. This simple off-line processing approach is certainly suitable for short utterances and fairly stationary noise. However, in general the noise power needs to be tracked over time. Many other approaches make use of a VAD and acquire the noise power during periods of speech absence.

- **Adaptive filters:** While the VAD enables the end pointing and dropping of noise-only frames, the enhancement of speech segments requires an adaptive filter which, in most cases, is based on noise power and SNR estimates. One of the classical approaches is spectral subtraction (SS) family algorithms for reducing additive noise. In these methods, an average signal spectrum and average noise spectrum are estimated in parts of the recording and subtracted from each other, so that average signal-to-noise ratio (SNR) is improved.

  Other well-know methods are: nonlinear SS, which follow the general idea of applying a small overestimation factor in high SNR regions and a large factor in low SNR regions, and Wiener filtering, a linear estimator that minimizes the mean-squared error between the original and enhanced speech [28].

Feature enhancement

Speech feature-enhancement techniques require that the speech waveform is processed to get a sequence of feature vectors —the so called speech features— of a relative small number of dimensions. This reduction is necessary to not waste resources by representing irrelevant information and to prevent the curse of dimensionality. The transformation of the speech waveform into a set of dimension-reduced features is known as speech feature extraction.

Therefore, the set of transformations done to the features so that the resulting ones will contain only relevant information is what it is called feature enhancement. Some of the well-know transformations are based on spectral and cepstral processing or feature compensation [24].

Model enhancement

The main weakness speech systems resides in their lack of robustness to variability. Usually, the problem resides in the dependence of probabilistic models on their training corpora as, in practice, it is common for probabilistic models to be used in

application contexts that differ considerably from the context of their training data.

Such mismatch between training data and application context (test data) causes the models to lose some of their precision and predictive power, in turn degrading the performance of the system. Therefore, model adaptation consists in reducing the mismatch between probabilistic models and the data against which they are used. This mismatched is usually caused by the factors we mentioned in subsection 2.3.3.

2.4 Related work

As seen all along this chapter, there are many techniques and methods used to solve projects similar to ours. It is not only important to know these techniques, but also the performances of that system in order to compare and value the results of the present one.

One of the most similar ones found in the literature is StressSense [29]. This study puts forward an unobtrusive stress detection system using human voice from smartphones. In this project, the SUSAS database is used to develop a first baseline system. After that, speech and GSR is collected from 14 participants. This data is collected in several environments (indoor and outdoor) and features such as pitch, MFCC, TEO and speaking rate are extracted from it. Finally, the study uses a Gaussian Mixture Model and model adaptation techniques that results F-score of 64.4%-83.6% for indoor and 60.1%-78.9% for outdoors (for different sets of features). The system was embedded in an Android application implemented in C, C++ and Java.

Another example that takes into account the integration of the system in several devices is [30], a library for speech analysis on mobile phones implemented in C is presented. In this study, a comparison between two speech analysis libraries is done: the well-known openSMILE [31] library for PC and AMMON. They compare the results of extracting features from SUSAS and training a Support Vector Machine (SVM) classifier. The results are among 75.51%-93.60% accuracy, depending on the feature set.

Regarding the problem of stress detection, it is interesting to mention the study by Mariana Dimas of the University of Porto [32], whose objective was to find the most revealing set of features for stress detection in speech. To do this, she proposed extracting an initial set of 6365 features, including non-linear ones such as TEO. Subsequently, feature selection step is done, obtaining a reduced set as input for the classification between stress and neutral. For this last step, SVM classifiers are used to achieve classification rates around 60%-75%.

Another less similar but relevant example is the project carried out at the University of Alcalá de Henares on stress detection through the analysis of emotions in speech [33]. In the study they used the EMO-DB database (Berlin Database of Emotional Speech [34]) to extract a set of standard features: MFCC, energy, pitch, jitter and shimmer. The classification of stress is made through the clustering of 7 emotions in groups of low, medium and high content of stress. The results obtained for the classification of 3 classes were around 80-85% success rate, while for 7 classes,

they obtained performances around 63%-68% accuracy.

Finally, as for noise robustness, it is interesting to mention UT-Scope[10] project, which not only focuses on collecting its own dataset (mentioned in subsection 2.2.1) but implements a Speaker Identification system. This system uses MFCC's and some spectral coefficients to train a GMM in different levels and types of noise and conditions (matched and mismatched). The average accuracy in this case results in 46% for the worst case and 83% for the best one.

Chapter 3

Design and Implementation

3.1 Solution design

The challenges of this project are designing and implementing a stress detection system from voice signals, with heart rate measurements used to label the data, suitable for integration into the BINDI system and able to properly work in real environments.

The solution to this problem will be divided into 2 independent parts or phases in order to meet the objectives mentioned in section 1.4.

The first one will consist on implement a recording procedure which let us generate a contaminated database that includes the distortions produced by the device and environmental noise. This database will allow us to train and evaluate the robustness of the stress detection system against different noise levels as a first proof-of-concept.

The second will focus on the Java and Python implementation of the code of the stress detection system based on speech and measurements of heart rate values, meeting the necessary requirements to be integrated into the smartphone used in BINDI.

It is important to mention that, as a common aspect of both phases, VOCE Corpus will be used. There are several reasons why this database is chosen: first, for having data taken in real and not stress induced environments; second, for offering data from sensors similar to the BINDI system bracelet (heart rate measurements); and third, for having previous studies [6] confirming the feasibility of a project that directly relates a heart rate metric with stress in speech.

As mentioned in section 2.2.1, original version of the dataset consists of voice recordings and ECG data measurements that result from a set of 45 different speakers.

Voice files (.wav format) correspond to three different recording settings: *pre-baseline*, reading a standard text at least 24 hours before the event in public (Public Speaking, PS); *baseline*, reading the same text as in the pre-baseline setting approximately 30 minutes before the PS; and *recording*, free speaking from a public event of free duration. According to VOCE's own documentation, the *pre-baseline* and *baseline* files would correspond to a neutral state, that is, what we consider label 0. On the other hand, the *recording* file would correspond to the state of stress or label 1.

Together with these audio files, 117 files (.xml format) are collected, each of them matching a voice file, containing measures to estimate the heart rate (Heart Rate, HR).

As a starting point, we will not use these original data, but the already processed database of the preliminary study [6]. The main differences between original and processed VOCE Corpus are:

- As it will be later detailed in subsection 3.3.2, the labels used for classification phase will not be the ones associated from original VOCE Corpus to each recording moment (0 for *pre-baseline* or *baseline* and 1 for *recording*), but they will be generated based on HR values using a unique threshold for each individual based on a 75% percentile. These labels have 1 second resolution.
- Because not all individuals in the database have all the necessary files (three recording settings and the three corresponding HR data files), a subset of VOCE Corpus, previously selected in the preliminary study [6], will be used in this work. In this way, the data subset to be worked with will consists of 18 speakers, each of them associated to 2 audio files, one of the neutral states (*pre-baseline* or *baseline*) and the *recording* state, and the corresponding two heart rate data files.

3.2 Data collection phase

As mentioned above, the main objective of this phase is to implement a recording procedure or system which let us generate a database of contaminated audio files at several noise levels in order to evaluate the performance and robustness of the stress detection system with the device distortions in different realistic noisy situations.

Once we choose the database we want to work with, that is, processed version of VOCE Corpus, the next step is to select the noise we want to contaminate the audio files with. In order to make the system as realistic as possible, we decided to distort the database with an outdoors noise, specifically with environmental noise recorded from a park (NPARK noise) from the well-known noise database DEMAND [35]. This will be considered as a proof-of-concept that the collection of the distorted database is possible: the employment of a variety of noises is planned as future work.

To fulfil with the aspect of implementing a realistic and reliable system, we should take into account the distortion produced by the device that includes the audio coder (OPUS [36], with sampling frequency of 16000 kHz.), the microphone acquisition distortion and the bluetooth channel possible errors, mainly, i.e. the transmission noisy channel $h(t)$.

With all these requirements, the recording procedure is automatised for a noise level range between -5 dB to 20 dB. Figure 3.1 represents the block diagram of the pipeline for contaminating the audio files at each of the noise levels. This entire phase has been implemented in MATLAB.

FIGURE 3.1: Pipeline for data collection process

3.2.1 Step 1: Contaminated audios generation

This first step consists in generating contaminated audio files at a specific SNR level with noise chosen, in our case, park outdoors noise. We will work with a range of values from -5dB to 20dB, with a step of 5dB, and considering 20dB as clean speech.

In order to control these SNR values in the new audios, the following process is repeated for each of the VOCE Corpus audio files.

1. First, the original audios are downsampled, both VOCE Corpus files and noise, to 16 Khz, which is the sampling frequency of the audio codec we use.

2. By definition, SNR means the ratio between signal (S) and noise (N). The objective now is to fix a specific SNR value and calculate the scaling factor for the noise value required so that subsequently, the sum of the signal and the scaled noise results in the corresponding SNR.
   Furthermore, to calculate the *signal* mean power, we must avoid the silence areas from speech since the duration of them could heavily influence this value. To do this, a Voice Activity Detector from Matlab Voicebox Toolbox [37] is used to detect these areas in order to correctly calculate the *signal* term of SNR.

3. Once we have calculated the proper scaling factor and multiply it by the noise, we can add it to the signal, obtaining an audio contaminated with additive noise at the prescribed SNR levels, in mono mode at 16 kHz.

This process is done for each of the downsampled audios from VOCE Corpus subset. Next, in order to automate the subsequent recording process (subsection 3.2.2), and since the duration of the audios is relatively short (between 3-15 minutes), we decided to concatenate all the files, already contaminated, into a single file with longer duration.

This new long audio consists of the concatenation of all the contaminated files for a given SNR value, separated by a clapper generated as a Dual-Tone Multi-Frequency (DTMF) signal. Specifically the sequence "123", surrounded by a silence gap of 3 seconds was selected in order to detect it easily and recover the individual audios automatically in the final step. This operation is critical since the labels need to be properly synchronized with the audios to allow the proper training process. Figure 3.2 shows an example of the result of this first step.

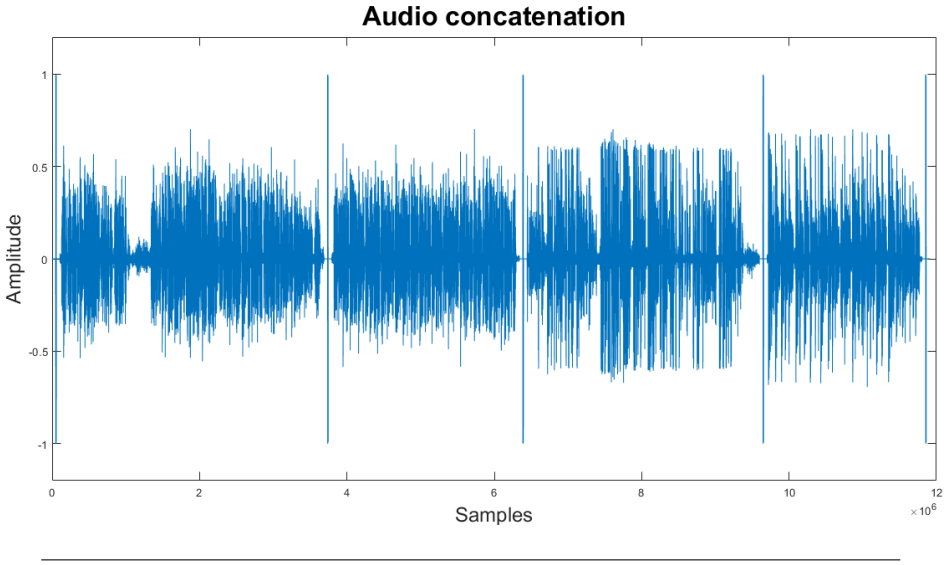

FIGURE 3.2: Audio concatenation at 20 dB

3.2.2 Step 2: Audio Recording

Once contaminated audios are generated at different noise levels, the next step consists in recording the audios with the BINDI's pendant microphone that codes and transmits it to the smartphone to take into account the acquisition and transmission distortions.

With the purpose of making the recordings with the highest possible quality avoiding room echoes, these audios are recorded inside an anechoic chamber. There, a single speaker reproduces the contaminated audios while BINDI's microphone, connected to a computer via USB with the outside, captures it. This process was first thought to be done via Bluetooth. However, due to several problems related with packet loss during the recording and connection loss between the microphone inside the anechoic chamber and the computer outside, it ended up being done via USB. This will be one of the future work lines proposed in chapter 6.

This process is automatically done from MATLAB, controlling microphone drivers. Drivers configuration and microphone connections were carried out by the Team from the Electronics Technology Department.

When the recording finishes, a text file is generated with the recorded data and saved in the computer via USB. This text file contains the concatenated audio, taking into account environmental, acquisition and transmission noise. Figure 3.3 and Figure 3.4 show the configuration inside the anechoic chamber during the recording procedure.

FIGURE 3.3: Anechoic Chamber

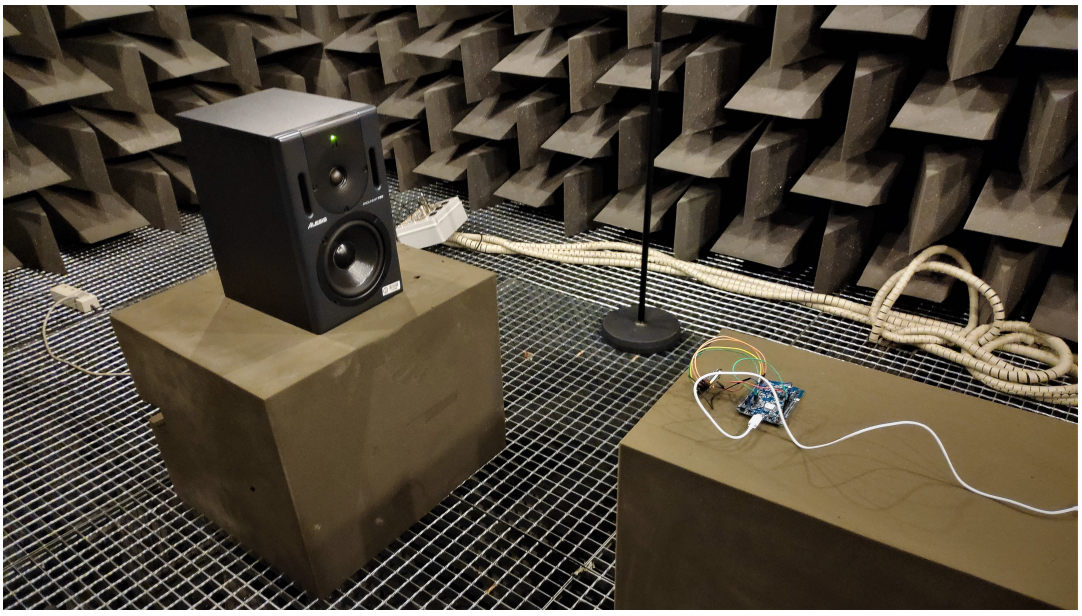

FIGURE 3.4: Speaker and microphone set

3.2.3 Step 3: Audio Recovering

Finally, last step consists in recovering the contaminated audios. To achieve this, we read the text files and detect the clapper with a DTMF decoder [38].

The data between two clappers detected is saved into a *wav* file. In this way, after recording, reading and splitting all the data, we recover the audio files, now contaminated with the different noises and distortions.

With this final step, the first of the sub-objectives of the project is fulfilled (see 1.4). The resulting dataset contains 36 contaminated audio files (2 per speaker by 18 speakers) with their corresponding ECG sensor files from the VOCE Corpus aligned for each of the six different SNR levels from 20 dB to -5 dB, taking into account environmental noise and distortions from the device. In summary, 216 contaminated audio files are now available. As mentioned before, each speaker is associated with 2 files corresponding with 2 different moments when capturing the audios and sensor in original VOCE Corpus: either *prebaseline* or *baseline*, and *recording*.

3.3 Implementation phase

3.3.1 Requirements

Before presenting the solution implemented, the technical requirements taken into account in the design should be enumerated:

- **The device should be low cost and wearable.**
- **The consumption of power should be low**: this implies that the intelligence (signal processing, feature extraction, classification model...) of the stress detection system will be allocated in the smartphone in order to save battery in the pendant. Therefore, the system must be implemented in Java, in order to be integrated in an Android application.
- **It should be speaker dependent**: The system must be capable of adapting to the user. A JSON file containing the user's profile is kept with the speaker dependent configuration.
- Although the training phase takes into account the ECG measurements for labelling stress, the classification must be done using **speech** as the sole input. The other physiological signals captured by the bracelet are employed for triggering the start of the speech module (given they are considerably less expensive in terms of battery consumption) but are not considered as inputs in it.
- Execution time must be as short as possible, due to the fact that the system must work in **real-time**.

3.3.2 Pipeline

The implementation will be divided in two main phases: training and testing. Training phase will consists in training the models with the new collected data. This process is done off-line. On the other hand, testing phase will be the part which integrates in the smartphone. For this reason of embedding or integrating the system

in the device, most of the code will be in Java excepting model training, that will be in Python and later exported to Java. The reason why Python is used in this model training phase is due to the lack of advanced machine learning libraries in Java, so, in order to obtain results as reliable as possible, it is decided to use this language.

The complete proposed solution is explained below. For each of the two phases a block diagram to provide an overall view will be provided. Then, each block will be detailed including its inputs, outputs, methods and techniques used.

Training phase

Training phase consists in generating a classification model capable of predicting stress or neutral labels from speech input. To do this, we will follow a conventional machine learning pipeline as shown in Figure 3.1.

As mentioned before, most of the processes will be implemented in Java. However, due to the lack of advanced libraries for machine learning in Java, the training of the model is implemented in Python and later exported to Java. Figure 3.5 and Figure 3.7 represent the complete training phase in Java and Python, respectively.

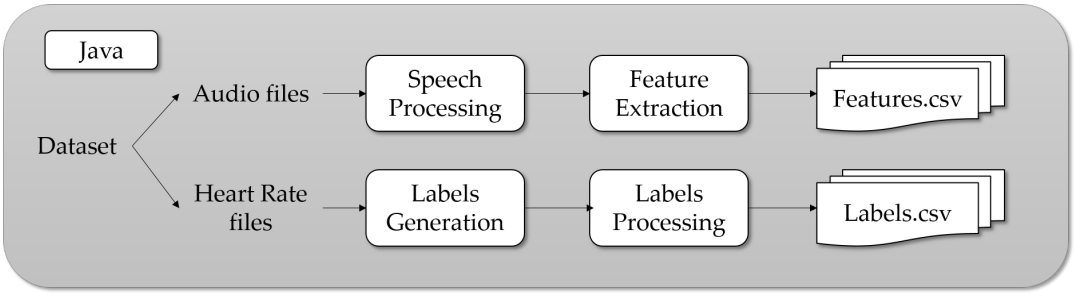

FIGURE 3.5: Training phase: Java processes

The start point is the dataset. Although in subsection 3.2.3 we finally got 36 contaminated audios (2 per speaker, 18 speakers) for every SNR value between -5 dB and 20 dB, we will only be able to use the recordings with length up to 15 minutes, due to lack of memory in the Java implementation. So, in this way, the dataset used for training will be composed of 32 audio files (from 16 speakers) and its corresponding sensor data (HR values) for each SNR value.

Next step is speech processing. This block consists of several processes with the aim of enhancing the signal before extracting its features. The processes are the following:

1. **Normalization** by its maximum value, in order to have an amplitude range between -1 and 1.

2. **Generate VAD vector**: this means we generate a binary vector which represents active ($label = 1$) and silence ($label = 0$) areas in the signal.

   For the implementation of this block in the system, different open-source libraries were tested. However, several problems were found that prevented

its use: too long execution times, lack of adaptation to our use case and errors due to the fact that the libraries were too complex. For these reasons, and taking into account the limitations it could carry out, a simplified VAD algorithm based on *Efficient voice activity detection algorithms using long-term speech information* study [39] was manually implemented in Java.

This VAD algorithm is designed for improving speech detection robustness in noisy environments. The algorithm measures the long-term spectral divergence (LTSD) between speech and noise and formulates the speech/non-speech decision rule by comparing the long-term spectral envelope to the average noise spectrum, thus yielding a high discriminating decision rule and minimizing the average number of decision errors. The decision threshold is adapted to the measured noise energy while a controlled hang-over is activated only when the observed signal-to-noise ratio is low.

3. **Spectral Substraction**: The spectral subtraction family algorithms for speech enhancement is the most widely used conventional method for reducing additive noise. In these methods, an averaged signal spectrum and averaged noise spectrum are estimated in different parts of the recording and subtracted from each other, so that average signal-to-noise ratio (SNR) is improved.

   As happened with VAD algorithm, no suitable open-source libraries in Java were found for our purpose and requirements. In this way, an own improved SS technique based on the one *Spectral over-subtraction* algorithm proposed in [40] was implemented in Java from scratch.

   The main idea of this method is to re-estimate noise spectrum every time silence areas are detected instead of using the initial estimation from the first 150 ms all along the analysis. In this way the estimation of the noise is updated along with that of the signal, taking into account that noise can affect non-uniformly. In other words, our algorithm starts with an initial estimation of the noise from the first 150 ms, as the original one. After that, if an non-speech frame is detected, noise is re-estimated based on the noise spectrum of that silence frame. In case the previous frame was silence too, a mean estimation from both frames is performed. This new noise estimation will be later used to denoise the subsequent speech frames until a new non-speech frame is detected.

4. **Removing non-speech areas**: After denoising the signal, VAD vector is also used to remove silence areas from the signals as we assume they do not contain information of our interest. The most important thing of this step, as we will see later, is that we must eliminate non-speech areas in 1 second chunks to avoid losing the synchronization with the labels, whose resolution is 1 second too.

Figure 3.6 represents an example of the complete process applied to a sample audio as it transverses the speech processing block.

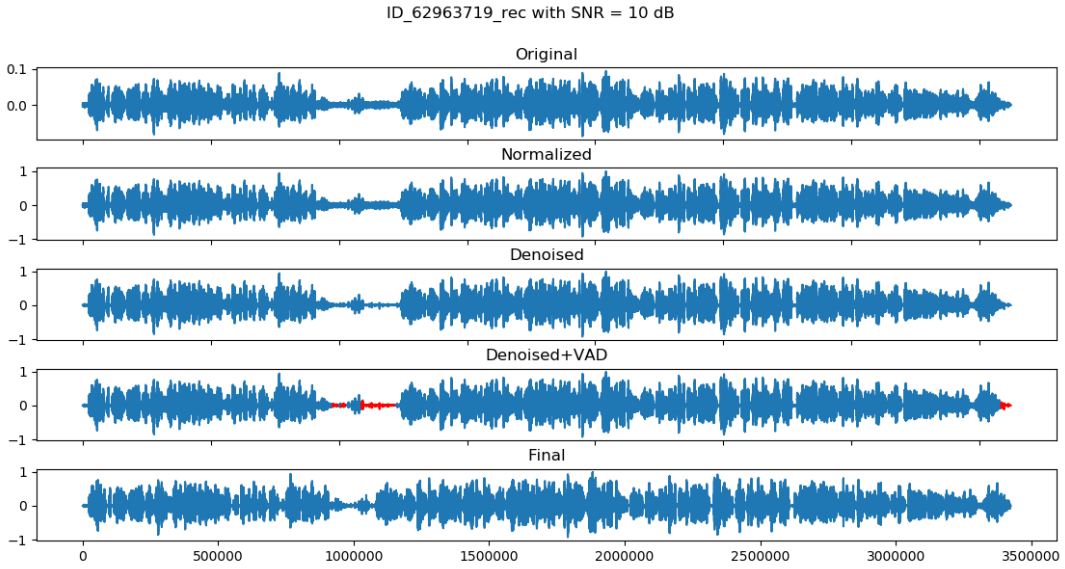

FIGURE 3.6: Speech processing pipeline example for speaker ID 62963719, *recording* moment, $SNR = 10dB$, where red areas represent silence frames

After the speech processing block, the signal is ready for feature extraction. In this step, software from several public repositories [41][42] are used to extract the whole set of features. For each of them, a window of 20 ms with 50% overlapping is used. After that, mean and standard deviation are calculated to aggregate them into 1 second resolution vectors. At the end of the extraction, we obtain a matrix as shown in Table 3.1. Feature matrices are later exported as CSV files.

Row Index	**Feature**
0 - 12	Mean MFCC
13 - 25	Standard Deviation MFCC
26 - 30	Mean RMS, Zero Cross Rate, Spectral Centroid, Spectral Flux, Spectral Roll Off
31 - 35	Standard Deviation RMS, Zero Cross Rate, Spectral Centroid, Spectral Flux, Spectral Roll Off
36	Mean Pitch
37	Standard Deviation Pitch

TABLE 3.1: Feature Matrix

Once we have finished the audio pipeline, we need to prepare the labels, In most machine learning datasets the labels are provided as part of the collection. In our case, the labelling provided was too loose for our purposes since one label was assigned to the whole audio file (pre-baseline, baseline and recording) corresponding to the three recording conditions (see section 3.1). Therefore, in our case, we not only have to generate the labels from other data, but we also have to process them based on speech processing information, as we explain below.

Thus, the first step is to establish a decision threshold, that is, to choose how we are going to determine whether, within an audio 1 second long chunk, there is stress or not.

As studied in [6], we propose a decision threshold, unique for each individual, based on heart rate values from *baseline* file (taken as neutral moment). This threshold establishes a percentage of frames that will be considered as stressed and therefore corresponds to the 75% percentile, in this particular case. As our sensory data resolution is 1 second, we apply the criterion that if a HR value exceeds the threshold, calculated for each particular speaker, the corresponding label will be 1 (stress) for that corresponding second and, conversely, if the value is below the threshold, the label will be 0 (neutral).

After labels are generated, we use the VAD vector generated in speech processing in order to eliminate the samples corresponding to silence in the speech signals. In this way, labels vector and feature matrices dimensions should match. Finally, labels are also exported as CSV files. This would be the end of the training data preparison in Java.

As shown in Figure 3.7, the last processes of the training phase are implemented in Python. We use the CSV files generated in Java as inputs in this Python phase. Once feature matrices and labels of interest are loaded, we divide the data in training (80%) and test (20%) sets.

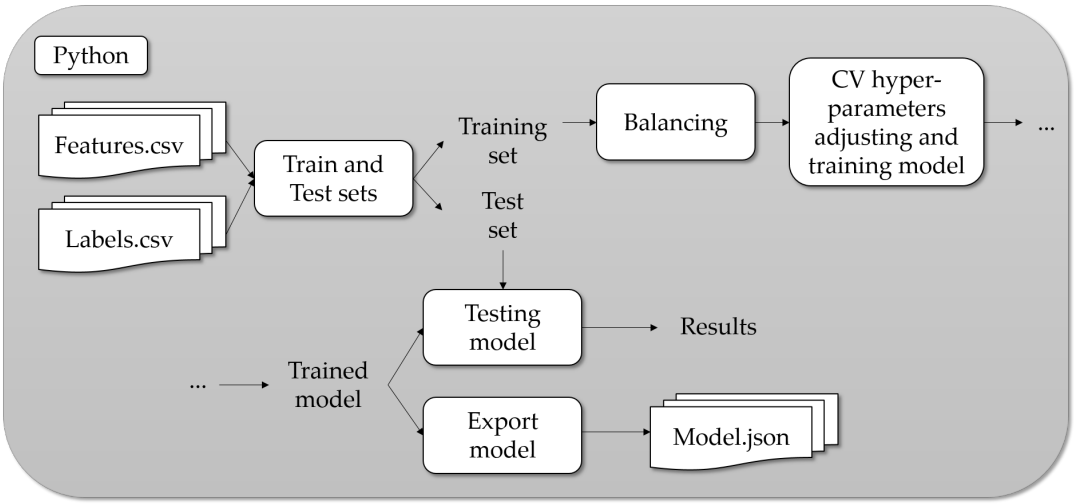

FIGURE 3.7: Training phase: Python processes

After analyzing the number of training samples for each class, we check the problem of possible *imbalance* in the data: we have much more samples of class 1 than class 0. To solve this, we balance the problem using *Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning*[43], specifically the algorithm *RandomOverSampler*, which over-samples the minority class by picking samples at random with replacement.

Once training set is balanced, we can start generating the model. To do this, we use Cross Validation to adjust the hyper-parameters of the model, using *Scikit-learn: Machine Learning in Python* API [44].

As we would later on need to export the model to Java, the choice of the classifier depends on the restriction of the Java library we are using. So, after trying several models and the way to export them, we decided to use Multi-Layer Perceptron (MLP) classifier. Therefore, the hyper-parameters to be adjusted are: hidden layer size, activation function for the hidden layer and regularization term.

Once the classifier is trained with the optimal parameters, we predict test labels and get the corresponding score, as we will see in chapter 4 results. Besides that, we export the trained model to Java with *sklearn-porter*, an open source API which transcribes trained scikit-learn models to several programming languages such as C, Java, JavaScript and others [45], generating a JSON file with the corresponding parameters, and a Java class for the classifier.

This way of exporting the model allows us to define different profiles or even improve the model by changing only this JSON file. In this way, we generate a baseline or general profile with VOCE Corpus that can be later personalized for the target speaker.

In this way, the training phase would be ended, having a trained model ready to be integrated in the smartphone.

Test phase

As shown in Figure 3.8, test phase shares common blocks with training phase. However, there are some differences to mention.

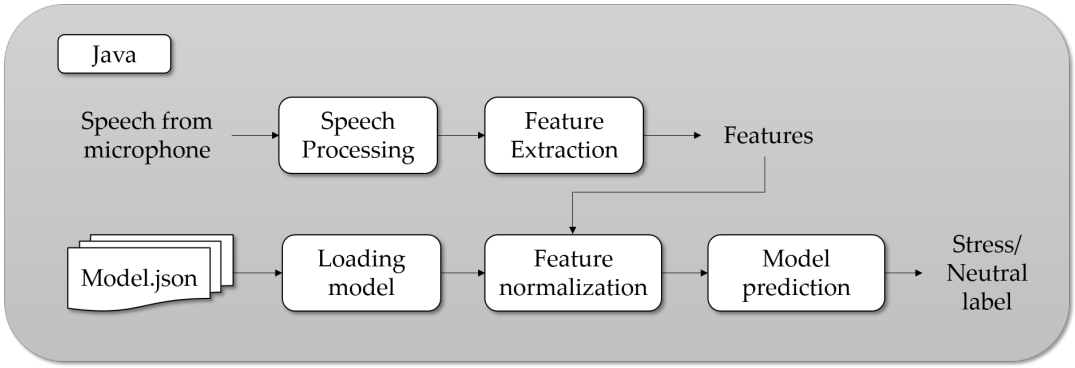

FIGURE 3.8: Test phase in Java

Now, the input signal will be originated on-line in the microphone of the device and not a stored off-line database. As we explained before, speech signals go through several processes such as VAD and spectral substraction to finally extract their corresponding features.

In contrast to the training phase, we do not use heart rate data now and therefore the stress detection is only based on the speech features. Once we have them extracted, we normalized them with trained model parameters (mean and standard deviation), exported from Python.

We also import the optimal hyper-parameters from the JSON file to have the complete trained model in Java. In this way, we can predict the corresponding labels of the streaming speech that comes from the microphone, getting an output of stress (1) or neutral (0) per second.

This output is finally used in the whole BINDI system to define a threshold (e.g.: accumulate 10 consecutive seconds of stress) which determines whether or not to send the alert message to the selected group of people.

In this way, the second sub-objective related with designing and implementing the stress detection system in Java to allow its integration in the smart phone application would be completed.

Chapter 4

Results

In this chapter, we focus on the achievement of the third and last sub-objective presented in section 1.4 which consist in testing the robustness of the models obtained before with the new data collected using the new implementation described in chapter 3.

In this way, we collect the different experiments carried out during the work, together with their respective results, in order to evaluate different aspects and conditions of the stress detection system as a first proof-of-concept, as we contaminated the data with one kind of noise.

4.1 Noise influence on VAD

In order to verify that the experiments done in this chapter are comparable regardless of the SNR at which they are performed, a preliminary study is carried out to analyse the influence or effect of noise on the performance of the Voice Activity Detector. We need to take into account that stress/non-stress decisions are only taken for frames that contain speech. Therefore, if due to noise, the VAD detects a different number of non-speech frames, the precision figures might not be comparable. To assess this influence, recall that the resolution of our system and labels is 1 second. Then we calculate the rate between non-speech seconds detected by the VAD and the total number of seconds for the whole set of audio files, for each of the SNR values considered. As a result, we obtain the following rate values represented Figure 4.1. This results are later detailed in Appendix A.

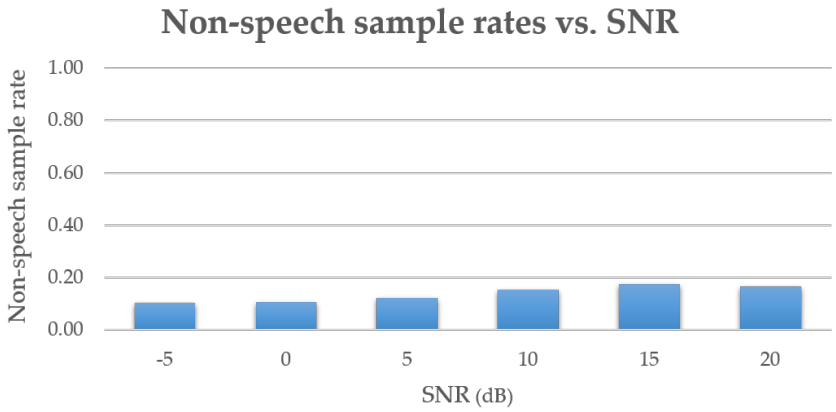

FIGURE 4.1: Non-speech sample rates vs. SNR

As we can observe in the figure, non-speech samples rate values are kept almost constant for the different SNR levels, indicating that Voice Activity Detector maintains its performance almost independently of noise level. If we calculate maximum difference between the maximum and the minimum rate, we obtain 7.25%, which seems to be a reasonable value to work with.

In view of the previous results, we can conclude that the system performances at different SNR values can be compared, since the samples that the VAD will eliminate in speech processing phase should be very similar for the different SNR levels.

4.2 Noise influence on stress detection

Once we know that the results we obtain from the system are comparable along the different noise levels, we will distinguish two main types of experiments based on the conditions of training and test sets used: **matched** and **mismatched conditions**.

In addition to this, the following experiments will evaluate two different scenarios: with and without the Spectral Substraction (SS) block, in order to determine how much the noise affects the performance of the system if we try to enhance the signal quality.

As mentioned in chapter 3, the implementation of the classification model has been done in Python with the Scikit-Learn library. This library allows us to adjust the hyper-parameters of the classification model in a simple way: indicating the candidate values of the hyper-parameters in a vector to be evaluated and the metric to be optimized during the adjustment. Therefore, a 5-fold cross validation has been done in every experiment to adjust the following hyper-parameters ranges of the MLP[1] :

- **Hidden layer size**: by default (100,), which means one hidden layer with 100 neuros. The values tested were [(100,), (150,), (200,), (250,), (300,)].
- **Activation function for the hidden layer**: ['identity', 'logistic', 'relu', 'tanh']
- **Regularization term L2**: a 10 samples log-spaced vector from 10^{-4} to 10^{-1}.

4.2.1 Experiments in matched conditions

As first experiments, matched conditions are analysed. The purpose of the matched conditions experiments is to evaluate the performance of the stress detection system in an ideal situation, that is, in the case where we evaluate how well the detection is when the model already knows the environmental noise in which it would operate. Therefore, in these cases, each classifier uses only samples of the corresponding SNR to evaluate, splitting 80% for training and 20% for testing. Thus, when predicting the test samples, the classifier already knows the noise conditions, as it has been trained with samples in the same environment.

[1]More information in: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

As a first approximation, and because we are evaluating a detection system, that is, in our problem we have labelled the neutral state with 0 and the event or detection of stress as 1, we decided to optimize the F-Score metric. This is because this metric, due to the way it is calculated, takes more into account cases of true positives, i.e. cases where the true label is 1 and the prediction is 1 too (detection).

In particular, this measure consists of the harmonic mean between Precision (P) and Recall (R), as defined below:

$$f\text{-}score = 2 \times \frac{P \times R}{P + R} \qquad \text{where:}$$

$$P = \frac{T_p}{T_p + F_p} \qquad \text{and} \qquad R = \frac{T_p}{T_p + F_n} \qquad \text{where:}$$

- T_p: True positives (true label=1, predicted label=1)
- F_p: False positives (true label=0, predicted label=1)
- F_n: False negatives (true label=1, predicted label=0)

Once the metric to evaluate the performance is defined, we obtain the f-score results for the different SNR values, shown in Figure 4.2.

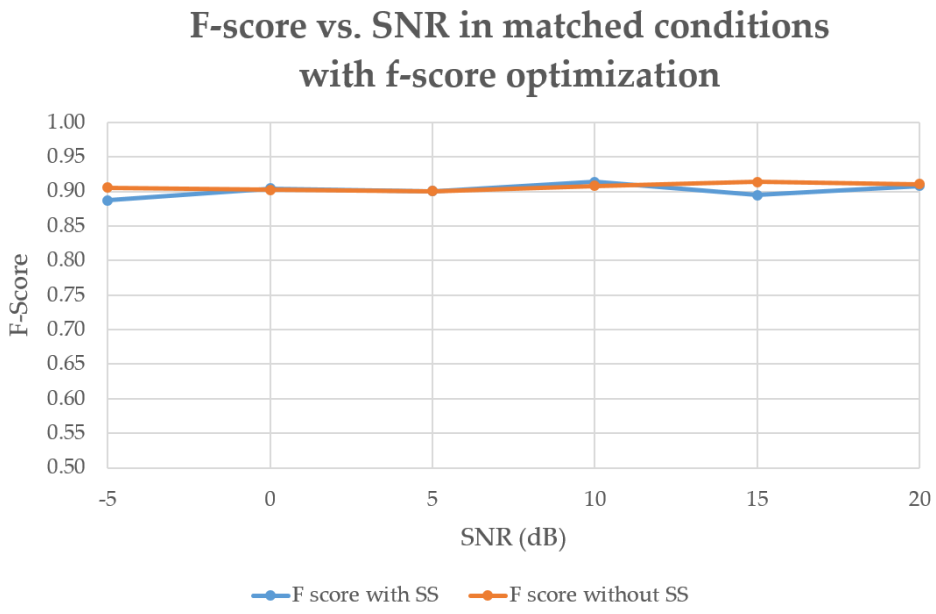

FIGURE 4.2: F-score vs. SNR in matched conditions with f-score optimization

Since the confusion matrices were also generated during the calculation of the results (see Appendix A), we decided to calculate the accuracy from them. We wanted to explore whether or not there are differences in the behaviour of the results for the different SNRs. These results should be reliable because during the generation of the classification model the training samples had been balanced. This accuracy values are represented in Figure 4.3.

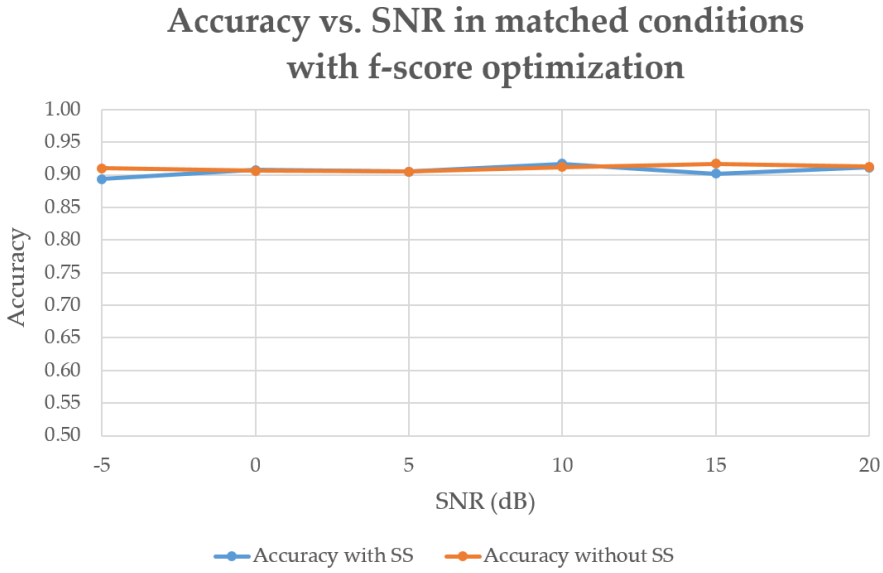

FIGURE 4.3: Accuracy vs. SNR in matched conditions with f-score optimization with and without SS.

As we can observe in Figure 4.2 and Figure 4.3, the system detects the presence of voice stress with high success rates for test samples in matched conditions. We see that for both the f-score metric and accuracy, the values behave similarly regardless of the SNR level. As expected, these values are high as the model is making predictions on samples for which it already knows the conditions, i.e. in matched conditions, the training and test samples belong to the same SNR value data or same noise conditions.

In this way, we can conclude the model detects stress or neutral quite correctly when it knows the conditions of the environment, that is, when it has been trained with the same noise in training and test phases. In this case, SS does not improved nor deteriorate the performance at any of the SNRs considered except for -5 and 15 dBs with an insignificant decrease when we use SS.

4.2.2 Experiments in mismatched conditions

After observing how the system behaves under ideal conditions, i.e. when the model knows the conditions of the test samples a priori, we continue to analyse the system under mismatched conditions.

In contrast with previous experiments where the test samples were a subset (20%) of the total samples for a specific SNR value, we will now train our model with samples that we will consider clean speech, in our case, the samples associated with the SNR value = 20 dB.

Once we have the classifier trained in clean conditions, using *scikit-learn* cross-validation method to adjust the hyper-parameters again, we evaluate the performance of the system for each of the different SNR values. Now the classifier will not know the conditions of the test samples (except in the case of 20 dB) and therefore

this is considered more realistic.

As we did in the previous case, we trained the model to optimize the f-score metric, the results of which are shown in Figure 4.4. After this, we calculated the accuracy values (Figure 4.5) from the confusion matrices in order to evaluate the behavior of the system completely.

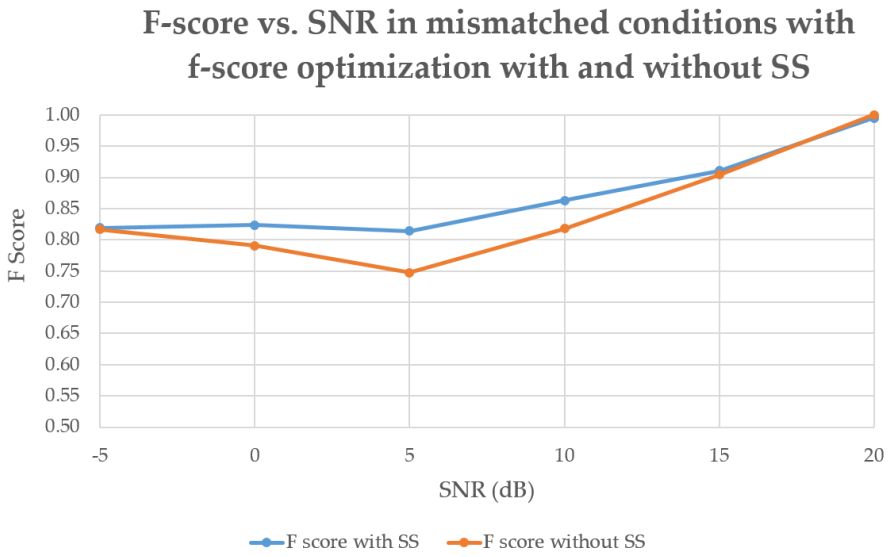

FIGURE 4.4: F-score vs. SNR in mismatched conditions with f-score optimization with and without SS.

If we compare the results of the experiments in mismatched conditions (Figure 4.4 and Figure 4.5), we detect two aspects that stand out:

On the one hand, in the interval between 5 dB and 20 dB we observe that both the f-score and the accuracy worsen as the noise level increases, which is what we expected.

However, if we observe the span between -5 dB and 5 dB, both for the cases in which we use SS and we do not, an unexpected behaviour appears: the performance in this case seems to be better for lower SNR values.

This strange behaviour leads us to wonder if optimizing the f-score metric is the best option. As mentioned above, the f-score metric focuses on true positives, i.e. detection, which coincides with the objective of our system.

However, if we analyse the data we are working and generating the models with, we observe that, regardless of the noise level, there is always an imbalance between the two classes (which is later artificially corrected) in which the majority class is 1 (stress, $\approx$ 75%) and the minority class is 0 (neutral, $\approx$ 25%). This is just the opposite of what is expected from a detection system, in which the minority class should be 0 and eventually the occurrence to be detected would happen, i.e. class 1.

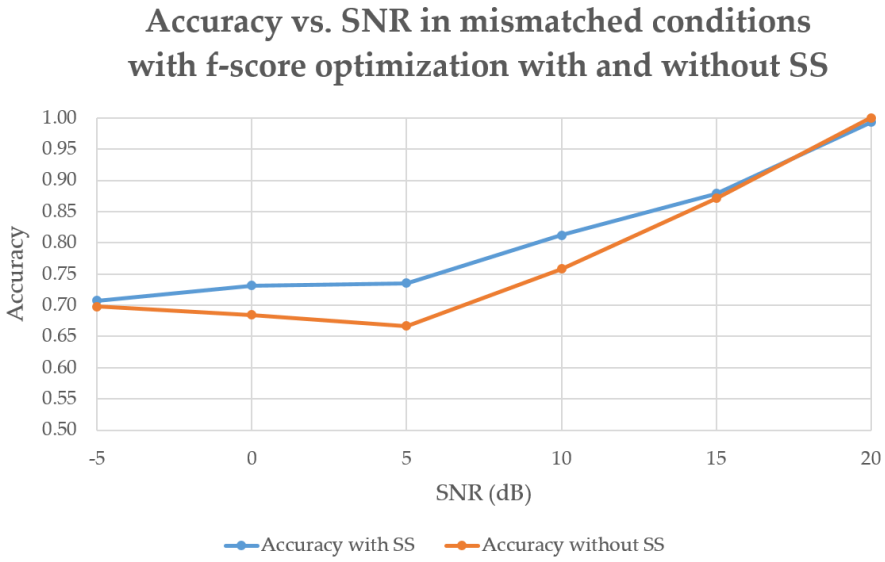

FIGURE 4.5: Accuracy vs. SNR in mismatched conditions with f-score optimization with and without SS.

As a way of exploring this problem, we decided to train the model again in clean conditions, this time optimizing accuracy metric instead of the f-score. These new results are collected in Figure 4.6. The results are more detailed in Appendix A.

Now, as shown in Figure 4.6, accuracy values follow the behaviour expected. As we can observe, regardless of the use of SS or not, the greater the noise level, the worse the stress gets detected.

In addition, it is interesting to mention that at low SNR values the effect of using the Spectral Subtraction phase to enhance the signal becomes more noticeable, as the detection rates improves when using SS than not using it reaching a maximum relative improvement of 24.6% for -5 dB.

Therefore, the results obtained from the experiments in mismatched conditions reflect that the detection system presents relatively high robustness against noise, obtaining maximum accuracy values around 90% for the case of 15 dB and, in the worst case, 72% using the SS block and 58% without using it. These results can be later checked in Appendix A.

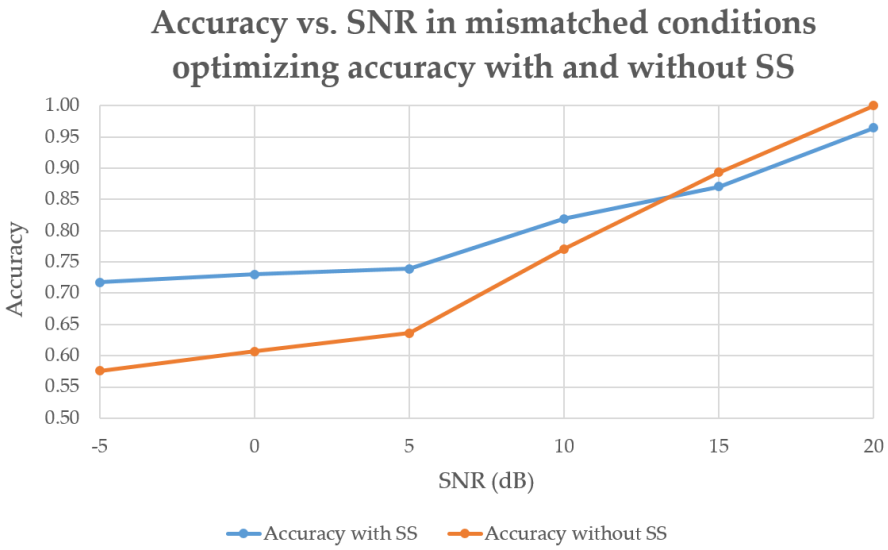

FIGURE 4.6: Accuracy vs. SNR in mismatched conditions optimizing accuracy with and without SS.

Chapter 5

Planning and Budget

The objective of this chapter is to explain the human, material and time resources needed for the complete development of the present master's thesis.

5.1 Planning

Below follows a detailed presentation of the time resources spent and the general steps or tasks in which the project has been organized.

A summary of the different tasks with their duration in weeks, description and section associated is presented in Figure 5.1.

In order to facilitate the understanding of the temporal process, this information is accompanied with a Gantt chart in Figure 5.2, that graphically indicates the duration of each task and the sequence followed, as well as the time dedicated to each task within the total duration of the project.

Task Name	Start Week	Duration (weeks)	Description
1. Requirements definition	0	4	Programming language, database, pipeline definition
2. First prototype	4	8	Translation of preliminar study to Java
3. Recording system implementation	12	4	Pendant integration, audio generation, automatic splitting system
4. Database contamination	16	6	Recordings collection
5. Second prototype	22	6	VAD and SS programming, database integration, code refactoring
6. Testing and improvements	28	5	Metrics evaluation, model adjusting
7. Report	33	6	Project documentation

FIGURE 5.1: Tasks summary

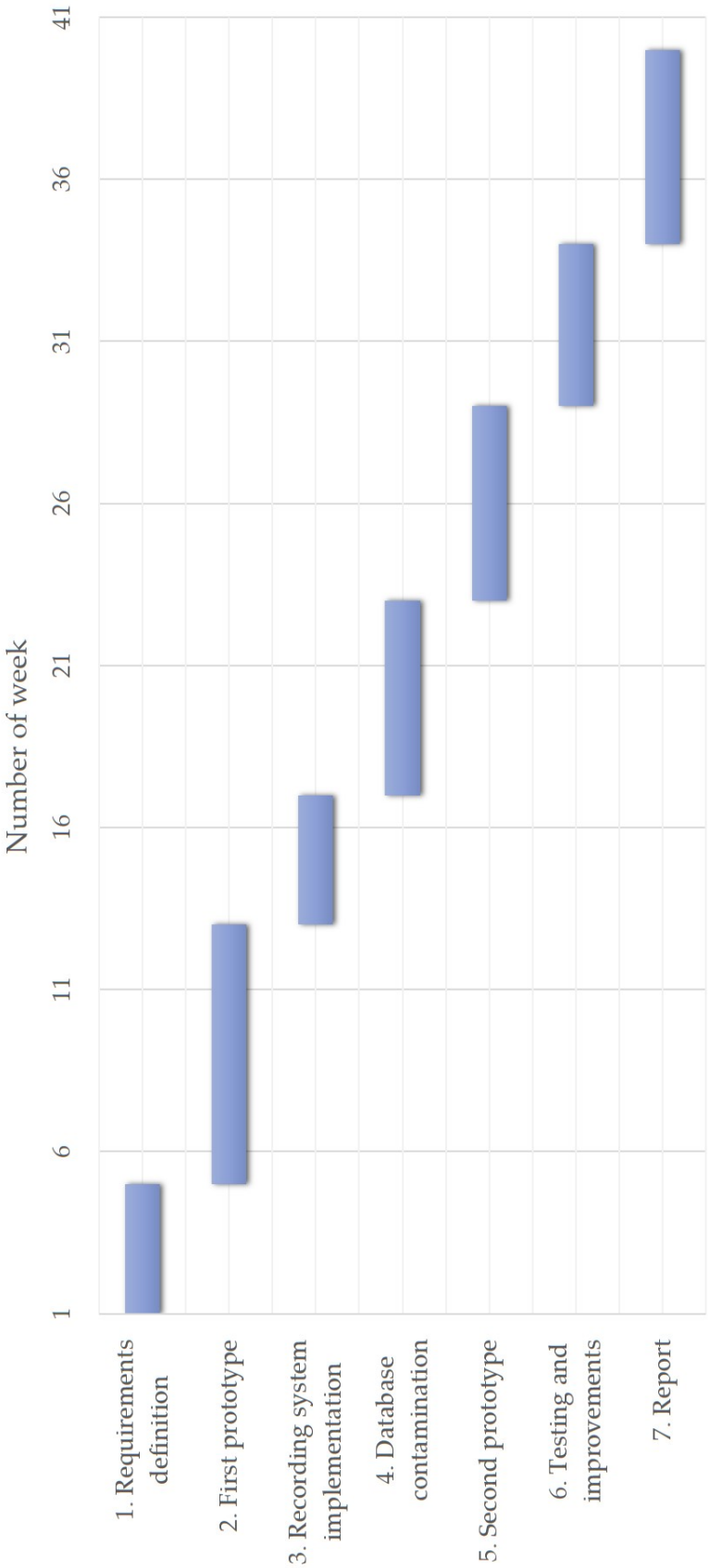

FIGURE 5.2: Gantt diagram

5.2 Budget

The following figures are an estimated budget of the project taking into account the materials, equipment, software and human resources for its complete development.

First, we list in Figure 5.3 and Figure 5.4 the resources related with material, equipment and necessary software to design and implement the whole system divided in the two same phases the system was developed: recordings collection and code implementation.

Materials and equipment for recording system	
Item	**Price**
NRF52-DK, Dev Kit para nRF52832 multiprotocol radio System on Chip (SoC)	33.15 €
STEVAL-MKI155V2 (4 microphone array MEMS)	38.68 €
Laptop for recording	Free for members of the university community
Anechoic room with equipment (speaker, connectors, wires...)	Free for members of the university community
Cable USB alcance 10 metros	19.50 €
BINDI pendant	32.04 €
Subtotal	**123.37 €**

FIGURE 5.3: Materials and equipment for recording system budget

Equipment and software for developing				
Item	**Total cost**	**Months of operation**	**Payback time (months)**	**Cost**
Laptop Acer Aspire E1-571	393.39 €	12	36	131.13 €
Microsoft Windows 10 Home 64 bits	-	-	-	114.55 €
MATLAB 2018 Student Licence	-	-	-	69.00 €
Eclipse IDE	-	-	-	- €
Anaconda IDE	-	-	-	- €
Subtotal				**314.68 €**

FIGURE 5.4: Equipment and software for developing budget

In addition, we present in Figure 5.5 human resources which consist of the author of the thesis, who has developed the system, and the supervisor.

Human resources (39 weeks)			
Item	**Hours dedicated (hours/week)**	**Cost/hour**	**Total**
Developer	18	22.50 €	15,795.00 €
Supervisor	2	27.00 €	2,106.00 €
Subtotal			**17,901.00 €**

FIGURE 5.5: Human resources budget for 39 weeks

Finally, we summarize all the necessary resources and theirs costs in Figure 5.6.

Total resources	
Item	**Subtotal**
Materials and equipment for recording system	123.37 €
Equipment and software for developing	314.68 €
Human resources	17,901.00 €
TOTAL	**18,339.05 €**

FIGURE 5.6: Total resources budget

Chapter 6

Conclusions and future work

This chapter presents the final conclusions of this thesis. To do this, we will focus mainly on the fulfilment of the objectives set out in Chapter 1 (see section 1.4). After this, we propose the future lines of work that would indicate the path for continuation of this work.

6.1 Conclusions

In this work, a stress detection system from voice signals has been developed as a part of BINDI project within Signal Theory and Communications discipline.

In order to meet the first objective, a complete recording procedure to collect a contaminated database which let us evaluate noise robustness has been implemented.

First, we require that the environmental noise (modelled as additive) and the acquisition and transmission noise (by empirically recording using a prototype of the pendant) are taken into account. Then, the pipeline for data collection was designed in order to generate contaminated audio files from VOCE Corpus at different SNR levels. With the aim of being as automatic as possible, the process was divided in three steps: concatenation and contamination of the original files at the different noise levels, audio recording inside an anechoic chamber using BINDI's pendant and splitting these recordings in order to recover audio files and aligning it to the corresponding labels.

In this way, the objective has been accomplished with the following limitations or aspects to improve:

- As for the automation of the recording process, due to the laptop used to manage the recordings, the chunks were a maximum of 20 minutes long. This limitation could be simply solved with more powerful equipment, so the recordings could be done at once.
- DTMF decoder needed to be initially set in order to adjust the threshold to detect the corresponding frequencies of the clapper. Due to this threshold, there are also very few rare cases in which decoder detects more frequencies than it should. These cases had to be split by hand.
- Another limiting aspect has been the kind of connection between the microphone inside the anechoic chamber and the laptop outside it while recording the audios. USB connection was used and therefore the Bluetooth transmission channel distortion was not recorded.

The second objective was about designing and implementing the stress detection system in Java to allow its integration in the smartphone application. Although some limitations related with libraries and code implementation were found in Java, we can say the goal has been successfully achieved for several reasons:

- We have developed a system which completely works in Java for the test phase, which is the one that will be integrated in the smartphone for the user. This system is capable of adapting to the user by a JSON file that could be substituted or updated either off-line or connecting to a server with Python implementation.
- The system classifies using speech as the sole input and it is fast enough to work in real-time.
- The implementation has taken into account many of the techniques used in similar works of literature, especially those related to noise robustness (speech enhancement) and feature extraction, so, within the limitations of libraries and implementation, we could take into account as many options as possible from the ones presented in the state of the art with the aim of developing a reliable and robust system.

As for the third objective about testing the robustness of the system with the new data collected and the new implementation, several experiments have been proposed. As a first approach, we checked our VAD component detected similar non-speech sample rate independently of the noise level. This let us conclude that the system performances at different SNR values could be compared, since the samples that the VAD would eliminate in speech processing phase would be very similar for the different SNR levels.

Next experiments tested the system with and without using Spectral Subtraction algorithm to enhance the signal, and were divided in two categories depending on the training and test conditions of the model: matched and mismatched conditions.

Matched conditions experiments showed us the system detects stress or neutral with high accuracy values independently of using or not SS block and, also, of the noise level. This means the system works quite satisfactorily when it has to detect stress in an already known environment.

On the other hand, mismatched conditions experiments showed us the difference between optimizing different metrics (f-score and accuracy) due to the imbalance of our database. We finally got the behaviour expected with accuracy optimization, that is, the system performance got worse as we increased the noise level, from a mean accuracy of 90% at 20dB to a 72% using SS and 58% without it at -5dB.

In this way, we checked the importance of using SS block when the system works in unknown condition, getting a maximum relative improvement of 24.6% when using it.

As presented in chapter 2, these values are within the results obtained in similar studies, so the objective can be also considered successfully achieved.

6.2 Future work

Once we finish this stage of the work, we propose future lines of work in which we can improve or expand different aspects of the work.

As for the implementation of the code, specifically the speech processing and feature extraction blocks, other libraries and algorithms could be tested in the future, in order to obtain higher quality signal after speech processing and, also, lighten the feature extraction process.

Regarding label generation, different percentile values could be analysed in order to study the optimal threshold that allows stress to be detected in the voice.

In terms of noise robustness, the next classic step would be to contaminate the database with new and different noises in order to be able to train the model in the future as a multi-conditional training case. This would result in reliable conclusions about the robustness of the system, as in this present work we could only check the feasibility of the it with a proof-of-concept that tested one type of noise.

As far as the models are concerned, it would be interesting to analyse the performance of other classifiers and other libraries to export from Python to Java. Also, researching and testing the performance of future releases of machine learning libraries in Java in order to integrate the entire system into the smartphone.

Finally, another interesting option would be to identify the most influential features in stress detection in order to better understand the origin of stress and its reflection in the speech.

Appendix A

Results summary

A.1 Noise influence on VAD results

The following table presents the results related with the preliminary experiment in which we analyse the influence of noise on performance of the Voice Activity Detector to verify that experiments are comparable regardless of the SNR at which they are performed.

SNR	Non-speech samples rate
-5	0.1028
0	0.1073
5	0.1231
10	0.1534
15	0.1754
20	0.1646

TABLE A.1: Non-speech sample rates vs. SNR results

A.2 Noise influence on stress detection results

This section collects the results related with noise influence on stress detection experiments.

A.2.1 Experiments in matched conditions

The following tables contain the f-score and accuracy values in matched conditions with f-score optimization with (Table A.2) and without (Table A.3) Spectral Subtraction block.

SNR	F score with SS	Tn	Fp	Fn	Tp	Accuracy with SS
-5	0.8867	927	48	161	818	0.8930
0	0.9039	913	47	132	842	0.9074
5	0.9002	888	41	138	807	0.9045
10	0.9136	855	43	105	782	0.9171
15	0.8949	832	44	126	724	0.9015
20	0.9081	834	33	123	771	0.9114

TABLE A.2: F-score and Accuracy results in matched conditions with f-score optimization with SS

SNR	F score without SS	Tn	Fp	Fn	Tp	Accuracy without SS
-5	0.9056	934	41	135	844	0.9099
0	0.9023	912	48	134	840	0.9059
5	0.9004	886	43	136	809	0.9045
10	0.9081	852	46	111	776	0.9120
15	0.9136	827	49	94	756	0.9171
20	0.9108	821	46	108	786	0.9125

TABLE A.3: F-score and Accuracy results in matched conditions with f-score optimization without SS

A.2.2 Experiments in mismatched conditions

In addition, the results of f-score and accuracy in mismatched conditions with f-score optimization are presented in Table A.4 and Table A.5, with and without Spectral Subtraction block, respectively.

SNR	F score with SS	Tn	Fp	Fn	Tp	Accuracy with SS
-5	0.8192	297	1659	345	4540	0.7071
0	0.8233	725	1249	578	4255	0.7316
5	0.8137	1045	956	816	3869	0.7350
10	0.8632	1420	573	639	3823	0.8122
15	0.9105	1650	325	437	3876	0.8788
20	0.9949	1955	13	32	4370	0.9929

TABLE A.4: F-score and Accuracy results in mismatched conditions with f-score optimization with SS

SNR	F score without SS	Tn	Fp	Fn	Tp	Accuracy without SS
-5	0.8166	175	1781	285	4600	0.6980
0	0.7903	617	1357	789	4044	0.6847
5	0.7473	1164	837	1391	3294	0.6668
10	0.8175	1394	599	963	3499	0.7580
15	0.9043	1646	329	482	3831	0.8710
20	1.0000	1968	0	0	4402	1.0000

TABLE A.5: F-score and Accuracy results in mismatched conditions with f-score optimization without SS

Finally, accuracy values when optimizing the models for accuracy metric are collected in next tables, both using SS block (Table A.6) and without using it (Table A.7).

SNR	Accuracy with SS	Tn	Fp	Fn	Tp
-5	0.7173	352	1604	330	4555
0	0.7303	830	1144	692	4141
5	0.7390	1202	799	946	3739
10	0.8189	1503	490	679	3783
15	0.8702	1688	287	529	3784
20	0.9644	1940	28	199	4203

TABLE A.6: Accuracy results in mismatched conditions with accuracy optimization with SS

SNR	Accuracy without SS	Tn	Fp	Fn	Tp
-5	0.5755	590	1366	1538	3347
0	0.6069	909	1065	1611	3222
5	0.6360	1285	716	1718	2967
10	0.7707	1480	513	967	3495
15	0.8933	1698	277	394	3919
20	1.0000	1968	0	0	4402

TABLE A.7: Accuracy results in mismatched conditions with accuracy optimization without SS

Appendix B

Code Repository

In order to share the project with the scientific community, it has been decided to publish the code used during the project in the following public GitHub repository:

https://github.com/minguezalba/A-robust-speech-based-stress-detector-designed-for-smartphone-integration

The README file attached to the project is shown below:

A robust speech-based stress detector designed for smartphone integration

---

The challenges of this project are designing and implementing a stress detection system from voice signals, with heart rate measurements used to label the data, suitable for integration into the BINDI system and able to properly work in real environments.

The solution to this problem will be divided into 2 independent parts:

Part 1: Recording system for data collection

The first one will consist on implement a recording procedure which let us generate a contaminated database that includes the distortions produced by the device and environmental noise. This database will allow us to train and evaluate the robustness of the stress detection system against different noise levels as a first proof-of-concept.

Requirements and packages dependencies:

- MATLAB >= 2016b
- VOICEBOX: Speech Processing Toolbox for MATLAB
- BINDIs pendant drivers
- str2doubleq for Fast String to Double Conversion

Directory tree:

```
1_generate_data:
  |-- originals: contains original audio files from VOCE Corpus,
  previously processed in a preliminary study [1].
  |-- recordings: contains clapper and noise files, and concate-
  nated audios ready for recording phase.
  |-- step1: generates concatenated audio files contaminated
   at different SNR values. They are saved in recordings/ directory.
  |-- step2: contains necessary code for connecting with BINDI
  microphone and capture audio to text file.
  |-- step3: reading the text files, converting to wav format
  and splitting them based on the clapper (DTMF Decoder).
```

Part 2: Stress detector implementation

The second will focus on the Java and Python implementation of the code of the stress detection system based on speech and measurements of heart rate values, meeting the necessary requirements to be integrated into the smartphone used in BINDI.

Requirements and packages dependencies:

- Java 1.8
- JAR files included: commons-math3-3.7.1, TarsosDSP, gson-2.6.2, opencsv-2.2
- Python 3.7
- Packages needed: numpy (>=1.8.2), pandas (>=0.23.0), scipy (>=0.13.3), scikit-learn (>=0.20), imbalanced-learn 0.4.2

Directory tree:

```
2_system:
  |-- TrainStress: contains Java project code to implement
  speech processing, feature extraction and label generation
  with audio files from data/data_recordings/
  |-- TestStress: contains Java project code to implement
  speech processing and feature extraction for test files
  in data/data_test/
  |-- data: contains necessary files and directories for
  TrainStress and TestStress projects. It contains audio
  files and features and labels generated from Java projects.
  |-- model: contains the necessary files for the training
  and test phase of the models (MLP classifiers) in Python.
```

Bibliography

[1] D. Insights, "Technology, media and telecommunications predictions 2019", 2018. [Online]. Available: https://www2.deloitte.com/content/dam/insights/us/articles/TMT-Predictions_2019/DI_TMT-predictions_2019.pdf.

[2] L. S.o. H. World Health Organization and S. A.M.R. C. Tropical Medicine, "Global and regional estimates of violence against women: Prevalence and health effects of intimate partner violence and non-partner sexual violence", 2015. [Online]. Available: https://bit.ly/2B5F3x6.

[3] (2018). Bindi: Smart solution for women's safety xprize. [Online; accessed 22-Jan-2019].

[4] (2018). Uc3m4safety, [Online]. Available: http://portal.uc3m.es/portal/page/portal/inst_estudios_genero/proyectos/UC3M4Safety (visited on 02/02/2019).

[5] (2018). Xprize, [Online]. Available: https://safety.xprize.org/prizes/womens-safety (visited on 02/02/2019).

[6] A. Minguez-Sanchez, "Detección de estrés en señales de voz", 2017, [Online; accessed 22-Jan-2019]. [Online]. Available: http://hdl.handle.net/10016/27535.

[7] A. Aguiar, M. Kaiseler, M. Cunha, J. Silva, H. Meinedo, and P. R. Almeida, "Voce corpus: Ecologically collected speech annotated with physiological and psychological stress assessments.", *Language Resources and Evaluation Conference*, 2014. [Online]. Available: http://paginas.fe.up.pt/~voce/docs/lrec2014-aguiar.pdf.

[8] M. P. Down and R. J. Sands, "Biometrics: An overview of the technology, challenges and control considerations", *Information Systems Control Journal*, vol. 4, pp. 53–56, 2004.

[9] D. Ververidis and C. Kotropoulos, "A review of emotional speech databases", [Online]. Available: https://bit.ly/2GnfsD1.

[10] A. Ikeno, V. Varadarajan, S. Patil, and J. H. L. Hansen, "Ut-scope: Speech under lombard effect and cognitive stress", in *2007 IEEE Aerospace Conference*, 2007, pp. 1–7. DOI: 10.1109/AERO.2007.352975.

[11] (1999). Susas database, [Online]. Available: https://catalog.ldc.upenn.edu/LDC99S78 (visited on 01/26/2019).

[12] (2017). R peaks in ecg signal figure, [Online]. Available: https://goo.gl/hejNyf (visited on 02/16/2019).

[13] C. D. Spielberger, R. L. Gorsuch, and R. E. Lushene, *State-Trait Anxiety Inventory (STAI): Test Manual for Form X*. Consulting Psychologists Press, 1968.

[14] A. Chaudhari and S. B. Dhonde, "A review on speech enhancement techniques", in *2015 International Conference on Pervasive Computing (ICPC)*, 2015, pp. 1–3. DOI: 10.1109/PERVASIVE.2015.7087096.

[15] (2017). Mfcc figure, [Online]. Available: https://goo.gl/NwnCFy (visited on 02/16/2019).

[16] F. Zheng, G. Zhang, and Z. Song, "Comparison of different implementations of mfcc.", *J. Comput. Sci. Technol.*, vol. 16, pp. 582–589, Nov. 2001. DOI: 10.1007/BF02943243.

[17] C. Panagiotakis and G. Tziritas, "A speech/music discriminator based on rms and zero-crossings", *IEEE Transactions on Multimedia*, vol. 7, no. 1, pp. 155–166, 2005, ISSN: 1520-9210.

[18] B. R, A. B.K., K. S, and B. Barkana, "Separation of voiced and unvoiced speech signals using energy and zero crossing rate", Mar. 2008.

[19] B. Belean, "Comparison of formant detection methods used in speech processing applications", *AIP Conference Proceedings*, vol. 1565, pp. 85–89, Nov. 2013. DOI: 10.1063/1.4833702.

[20] F. Jabloun, A. E. Cetin, and E. Erzin, "Teager energy based feature parameters for speech recognition in car noise", *IEEE Signal Processing Letters*, vol. 6, no. 10, pp. 259–261, 1999.

[21] F. Jabloun, A Cetin, and E. Erzin, "Teager energy based feature parameters for speech recognition in car noise", *Signal Processing Letters, IEEE*, vol. 6, pp. 259 –261, Nov. 1999. DOI: 10.1109/97.789604.

[22] M. Gales and S. Young, "The application of hidden markov models in speech recognition", *Foundations and Trends in Signal Processing*, vol. 1, pp. 195–304, Jan. 2007. DOI: 10.1561/2000000004.

[23] J. Ramirez, J. M. Górriz, and J. C. Segura, "Voice activity detection. fundamentals and speech recognition system robustness", in *Robust speech recognition and understanding*, InTech, 2007.

[24] T. Virtanen, R. Singh, and B. Raj, *Techniques for noise robustness in automatic speech recognition*. John Wiley & Sons, 2012.

[25] J. Ortega-Garcia and J. Gonzalez-Rodriguez, "Overview of speech enhancement techniques for automatic speaker recognition", in *Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP '96*, vol. 2, 1996, 929–932 vol.2. DOI: 10.1109/ICSLP.1996.607754.

[26] D. Malah, R. V. Cox, and A. J. Accardi, "Tracking speech-presence uncertainty to improve speech enhancement in non-stationary noise environments", in *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No. 99CH36258)*, IEEE, vol. 2, 1999, pp. 789–792.

[27] J. Ramirez, J. C. Segura, J. M. Gorriz, and L. Garcia, "Improved voice activity detection using contextual multiple hypothesis testing for robust speech recognition", *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 8, pp. 2177–2189, 2007, ISSN: 1558-7916. DOI: 10.1109/TASL.2007.903937.

[28] N. Upadhyay and R. K. Jaiswal, "Single channel speech enhancement: Using wiener filtering with recursive noise estimation", *Procedia Computer Science*, vol. 84, pp. 22 –30, 2016, Proceeding of the Seventh International Conference on Intelligent Human Computer Interaction (IHCI 2015), ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2016.04.061. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050916300758.

[29] H. Lu, D. Frauendorfer, M. Rabbi, M. S. Mast, G. T. Chittaranjan, A. T. Campbell, D. Gatica-Perez, and T. Choudhury, "Stresssense: Detecting stress in unconstrained acoustic environments using smartphones", in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, ACM, 2012, pp. 351–360.

[30] K. hao Chang, D. Fisher, and J. F. Canny, "Ammon : A speech analysis library for analyzing affect , stress , and mental health on mobile phones", 2011.

[31] F. Eyben, M. Wöllmer, and B. Schuller, "Opensmile: The munich versatile and fast open-source audio feature extractor", in *Proceedings of the 18th ACM international conference on Multimedia*, ACM, 2010, pp. 1459–1462.

[32] M. D. Julião, "Feature sets for stressed speech discrimination", 2014.

[33] I. Mohino, R. Gil-Pita, and L Alvarez, "Stress detection through emotional speech analysis", *Advances in Computer Science*, pp. 233–237, 2011.

[34] F. Burkhardt, A. Paeschke, M. Rolfes, W. F. Sendlmeier, and B. Weiss, "A database of german emotional speech", in *Ninth European Conference on Speech Communication and Technology*, 2005.

[35] J Thiemann, N Ito, and E Vincent, "Demand: A collection of multi-channel recordings of acoustic noise in diverse environments", in *Proc. Meetings Acoust.*, 2013. DOI: 10.5281/zenodo.1227121.

[36] J.-M. Valin, K. Vos, and T. Terriberry, "Definition of the opus audio codec", Tech. Rep., 2012.

[37] M. Brookes *et al.*, "Voicebox: Speech processing toolbox for matlab", *Software, available [Jan. 2019] from www. ee. ic. ac. uk/hp/staff/dmb/voicebox/voicebox. html*, vol. 47, 1997.

[38] J. Loomis. (2010). Dtmf decoder, [Online]. Available: http://www.johnloomis.org/ece303L/lab7/lab7.html (visited on 02/02/2019).

[39] J. Ramirez, J. C. Segura, C. Benitez, A. de la Torre, and A. Rubio, "Efficient voice activity detection algorithms using long-term speech information", *Speech Communication*, vol. 42, no. 3, pp. 271 –287, 2004, ISSN: 0167-6393. [Online]. Available: https://bit.ly/2G7Y3iI.

[40] N. Upadhyay and A. Karmakar, "Speech enhancement using spectral subtraction-type algorithms: A comparison and simulation study", *Procedia Computer Science*, vol. 54, pp. 574 –584, 2015, ISSN: 1877-0509. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050915013903.

[41] F. Uva. (2013). Soundbites, environment recognition through sounds, [Online]. Available: https://github.com/filipeuva/SoundBites (visited on 02/10/2019).

[42] J. Six, O. Cornelis, and M. Leman, "TarsosDSP, a Real-Time Audio Processing Framework in Java", in *Proceedings of the 53rd AES Conference (AES 53rd)*, 2014.

[43] G. Lemaître, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning", *Journal of Machine Learning Research*, vol. 18, no. 17, pp. 1–5, 2017. [Online]. Available: http://jmlr.org/papers/v18/16-365.html.

[44] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python", *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[45] D. Morawiec, "Sklearn-porter", Transpile trained scikit-learn estimators to C, Java, JavaScript and others, [Online]. Available: https://github.com/nok/sklearn-porter.