Análisis de datos útiles en predicción de trastornos emocionales

Ismael Setti Alonso - Jose Manuel Pinto Lozano

Trabajo de fin de grado del Grado en Ingeniería Informática

Facultad de Informática

Universidad Complutense de Madrid

Año académico 2018/2019

Directoras: María Victoria López López - Matilde Santos Peña

Índice

Índice figuras

Índice de ecuaciones

Resumen

El trastorno bipolar es una afección mental en la que una persona tiene unos cambios de ánimo muy marcados. Cuando se produce uno de estos cambios se denomina episodio. La finalidad de este trabajo es predecirlos utilizando *Machine Learning* (Aprendizaje Automático) sobre unos datos que se obtendrán de forma no intrusiva.

La primera fase del trabajo consiste en la obtención de los datos mediante dispositivos tecnológicos que utilizamos en el día a día, como son los smartphones y las pulseras de actividad, y con ellos tratar de predecir episodios en un paciente con trastorno bipolar y detectar si entrará, en un futuro cercano, en un estado de depresión o manía. Con esto el médico sabrá de antemano cuando un paciente puede sufrir un episodio y reaccionar con antelación de la forma que considere oportuna.

En concreto se ha trabajado con señales de voz, obtenidas de repositorios públicos, y con datos de actividad física para identificar estos estados. Las señales de voz servirán para predecir emociones. Esto ayuda a detectar cambios de ánimo, usado como indicador de un posible episodio. Los registros de actividad física también proporcionan información del estado anímico de un paciente. Por ello se consideran que son fuentes relevantes de datos y han sido elegidas para realizar proyecto.

Para la predicción se tratarán los datos a los que después se aplicarán distintos algoritmos de *Machine Learning* con el objetivo de determinar cuál de estos algoritmos produce mejores y más fiables resultados a la hora de predecir un episodio.

Además de la predicción de crisis, otros resultados que se pueden extraer es la importancia de las distintas variables a la hora de prever si el paciente tendrá o no un episodio próximamente. Con esto se podría profundizar más en la investigación de estos trastornos aislando los factores de influencia más relevantes para la detección de los cambios.

En la presente memoria se describe un análisis exhaustivo del proceso de obtención y preparación de los datos, además de la selección de algoritmos de reconocimiento y detección que tienen mejor precisión. Se discuten los resultados para proporcionar una herramienta de análisis a los especialistas que les ayude en la toma de decisiones.

Palabras clave

Aprendizaje Automático, análisis de voz, actividad física, procesamiento de datos, predicción, trastorno bipolar, depresión.

Abstract

Bipolar disorder is a mental condition in which a person has very marked mood changes. When one of these changes occurs, it is called an episode. The purpose of this work is to predict them using Machine Learning on data that will be obtained in a non-intrusive way.

The first work phase is to obtain data through technological devices that we use daily, such as smartphones and activity wristbands, and with them, try to predict episodes in a patient with bipolar disorder and detect if he will suffer, in the near future, a state of depression or mania. With this, the doctor will know beforehand when a patient may suffer an episode and react in advance in the way he considers appropriate.

In particular it has been worked with voice signals, obtained from public repositories, and physical activity data to identify this states. The voice signals will serve to predict emotions. This helps to detect changes in mood, used as an indicator of a possible episode. The physical activity records also provide information of patient mood. Therefore it is considered that they are relevant data sources and have been chosen to carry out the proyect.

For the prediction, the data that will be applied to different Machine Learning algorithms will be treated in order to determine which of these produces better and more reliable results when predicting an episode.

In addition from the crisis prediction, other results that can be extracted is the importance of the different variables when it comes to predicting whether or not the patient will have an episode soon. This could further deepen the investigation of these disorders by isolating the most relevant factors of influence for the detection of changes.

In the present memory an exhaustive analysis of the process of obtaining and preparing the data is described, in addition to the selection of recognition and detection algorithms that have better accuracy. The results are discussed to provide a analysis tool to the specialist that helps them in the decisionmaking.

Keywords

Machine Learning, voice analysis, physical activity, prediction, bipolar disorder, unipolar depression.

1. Introducción

1.1. Contexto de la investigación

Este trabajo surge como continuación de la tesis de fin de grado *Application of Machine Learning Algorithms for Bipolar Disorder Crisis Prediction* de Axel Junestrand, el cual trataba de anticipar los episodios de crisis en enfer bipolares para evitar los síntomas antes de que los pacientes empiecen a sufrirlos. Este proyecto utilizaba datos subjetivos cedidos por los propios pacientes a través de sesiones con el médico. El problema que esto planteaba es que los datos pueden estar falseados y no terminan de ser precisos, además de ser necesaria una sesión presencial para obtenerlos.

Con este trabajo se pretende analizar la posibilidad de realizar la predicción de estos episodios utilizando datos objetivos obtenidos a través de dispositivos móviles, como pueden ser los smartphones o pulseras de actividad y monitorización. Así los datos son más precisos, y pueden ser obtenidos de forma remota y no intrusiva.

Estos datos que se pretenden extraer son grabaciones de la voz y la actividad física monitorizada, debido a que se piensa que podrían determinar en parte el estado de ánimo de los pacientes y teniendo en cuenta cómo evoluciona éste podría desarrollarse un sistema de ayuda a la decisión para los especialistas.

1.2. Objetivos

El objetivo general de este trabajo es predecir crisis en pacientes bipolares en base al análisis de la actividad física y señales de voz. Esto serviría como sistema de ayuda en la toma de decisiones del médico especialista. Para ello se ha utilizado las siguientes fuentes de información:

- Con las señales de voz se pretende identificar emociones, en concreto, seis estados de ánimo que son: alegría, tristeza, miedo, asco, ira y neutral.

- Con los datos de actividad el objetivo es identificar dos tipos de enfermedades mentales, la depresión y el trastorno bipolar.

Para conseguirlo se plantean una serie de objetivos específicos que marcarán el desarrollo del trabajo.

- Estudio del estado del arte, para conocer las técnicas más utilizadas y recomendaciones.
- Análisis de dispositivos de medida de actividad física y de bases de datos de audios para obtener conjuntos con los que entrenar a los modelos.
- Procesamiento de las señales de voz y actividad para tener los datos listos para la modelización.

- Aplicar aprendizaje automático para clasificar los estados de ánimo a través de la voz y para detectar los trastornos mediante la actividad física.
- Comparación de diversas técnicas de aprendizaje automático para determinar la mejor.
- Discutir los resultados y comparar con otros trabajos relacionados.

1.3. Plan de trabajo

Para la realización de este proyecto se ha seguido un proceso de 5 etapas: obtención de los datos, limpieza y preparación de los datos, análisis exploratorio de los mismos, comparación de algoritmos y selección del algoritmo más adecuado.

En la fase de obtención de información se recolectaron todos los datos necesarios para el proyecto, de los que se obtuvo una versión final en formato csv para facilitar su tratamiento en futuras fases.

En la fase de limpieza y preparación se realizó un estudio de los datos obtenidos anteriormente, se detectan valores incorrectos o vacíos, se eliminan outliers y se normalizan los datos; todo ello con la intención de tener los datos preparados para realizar un análisis exhaustivo en la siguiente fase.

En el análisis exploratorio de los datos se determinó qué variables eran más relevantes a la hora de aplicar los algoritmos. Para esto se obtuvo la importancia que daban los árboles de decisión a cada variable y, estudiando la correlación existente entre las variables, se pudo eliminar las redundancias y reducir el conjunto de variables a las esenciales para así mejorar el tiempo de cómputo y la complejidad de los datos de entrada.

Con esos datos ya preparados se probaron distintos algoritmos de machine learning variando los parámetros de cada uno de ellos para realizar una comparación de los resultados obtenidos. Por último, en base a estos resultados se determinó que algoritmo sería el más idóneo para este problema.

Los resultados obtenidos son satisfactorios y aportan una información que puede resultar muy útil para la predicción de episodios en pacientes bipolares.

1.4. Tecnologías utilizadas

En el proyecto se ha utilizado Python 3 como lenguaje de programación, para procesar los datos y aplicar los algoritmos de aprendizaje automático, haciendo uso de Jupyter Notebook como entorno. Además, se ha utilizado el programa Praat para obtener las variables a partir de las ondas de sonido procedentes de los audios grabados.

1.4.1. Python

Python es un lenguaje de programación interpretado administrado por Python Software Foundation. Su uso en aprendizaje automático está muy extendido ya que cuenta con

librerías que ayudan al tratamiento (Pandas, Scikit-learn) y visualización (Matplotlib, Seaborn) de los datos, además de otras que proporcionan algoritmos de predicción (Scikit-learn).

1.4.2. Jupyter Notebook

Jupyter Notebook es una aplicación web que consiste en "notebooks" que incluyen código, el cuál puede ser ejecutado por fragmentos. También permite el uso de Markdown, lo que es de gran utilidad para comentar ciertos fragmentos de código. Otra ventaja que aporta el uso de Jupyter Notebook es la posibilidad de visualizar resultados de forma rápida, clara y sencilla.

1.4.3. Praat

Praat (http://www.fon.hum.uva.nl/praat/) es un programa de escritorio disponible para OSX, Windows y Linux, creado por Paul Boersma y David Weenink, profesores de la Universidad de Amsterdam. El programa permite analizar una onda de audio, poniendo a disposición del usuario varias herramientas para obtener el valor de distintas variables como: el tono (máximo, mínimo, medio) o la intensidad (máxima, mínima, media); permite modificar el rango de frecuencias a editar, obtener frecuencias de momentos concretos de la onda, obtener los formantes (pico de intensidad en el espectro de un sonido ya que la máxima concentración de energía, amplitud de onda, se da en una determinada frecuencia), y algunas funcionalidades más avanzadas para expertos.

1.5. Repositorio

El proyecto se encuentra publicado en un repositorio en GitHub: https://www.github.com/MrDevoid/TFG. En él se puede encontrar el siguiente contenido:

1. Archivos csv utilizados en el proyecto
2. Los scripts de Praat utilizados para obtener los archivos csv a partir de los audios
3. El notebook utilizado para generar los archivos csv a partir de los scripts de Praat y los audios
4. El notebook utilizado para entrenar y probar los algoritmos de predicción

El proyecto está protegido bajo una licencia GNU General Public License v3.0

1.6. Estructura del trabajo de fin de grado

En este documento se describe el proceso seguido en durante el proyecto. Este ha sido estructurado en capítulos para facilitar su lectura:

Capítulo 1: Introducción del proyecto en la que se comentan los aspectos generales de éste, tecnologías y datos utilizados, metodología de trabajo, y objetivos de la tesis.

Capítulo 2: Estado del arte en el que se mencionan algunos proyectos similares y las tecnologías con las que están trabajando. También se hace un análisis de las aplicaciones y dispositivos actuales que permiten monitorizar ciertas variables.

Capítulo 3: Descripción del preprocesamiento de los datos. Primeramente, se explica cómo se obtienen los datos necesarios para el proyecto, y se realiza un análisis de cada conjunto de datos, describiendo las variables de cada uno de estos. Tras esto se describen los procedimientos aplicados a los datos para prepararlos para el entrenamiento, como puede ser normalizar o tratar valores atípicos.

Capítulo 4: Análisis de los datos. En este apartado se estudian las variables de los datos obtenidos en el capítulo 3. Se observa la correlación que existe entre las variables, qué variables son más relevantes a la hora de predecir y cuales son menos significativas, todo ello con el objetivo de limpiar los conjuntos de datos y entrenar solo con las variables necesarias.

Capítulo 5: Algoritmos de machine learning. Descripción de los algoritmos utilizados en el proyecto para predecir y cómo se han configurado sus parámetros para este problema.

Capítulo 6: Resultados obtenidos al aplicar los algoritmos vistos en el capítulo anterior. Se analizan los resultados y se realizan ciertas modificaciones para mejorar su funcionamiento.

Capítulo 7: Conclusiones y trabajos futuros. Se comentan los resultados obtenidos finalmente y si es útil utilizar machine learning para estos problemas reales, explicando qué algoritmo o algoritmos serían los más adecuados. Además se detallan algunas vías en las que se podría avanzar en un futuro.

1.7. Asignaturas de la carrera relacionadas

A lo largo de la carrera hemos cursado una serie de asignaturas relacionadas con la inteligencia artificial y el tratamiento de datos e información que nos han proporcionado las competencias necesarias para la realización de este trabajo. Las principales materias son:

- Inteligencia Artificial, la cual daba una visión general de la inteligencia artificial simbólica y subsimbólica presente a día de hoy.

- Inteligencia Artificial Aplicada al Control, trataba de la utilización de algoritmos de inteligencia de artificial (computación evolutiva, sistemas expertos, redes neuronales y lógica fuzzy) aplicados a sistemas de control.

- Aprendizaje Automático y Big Data, asignatura que enseña el fundamento de algunos algoritmos como regresión lineal y logística, redes neuronales, SVM y otros, utilizados en problemas actuales de análisis de datos.

- Minería de Datos y el paradigma Big Data, en la cual se aprenden técnicas para un correcto manejo, procesamiento y/o explotación de grandes volúmenes de datos, así como la utilización de una adecuada representación de estos.

- Cloud y Big Data: principalmente se centra en el procesamiento de grandes cantidades de datos utilizando servicios cloud, frameworks como Hadoop o Spark sobre la plataforma de servicios cloud de Amazon.

1’. Introduction

1.1’. Research Context

This work arises as a continuation of the thesis *Application of Machine Learning Algorithms for bipolar Disorder crisis Prediction [11]* by Axel Junestrand, which tried to anticipate the episodes of crisis in bipolar disorder patients to avoid the symptoms before that patients begin to suffer. This project used subjective data given by the patients themselves through sessions with the doctor. The problem that this posed is that the data may be distorted and not being accurate, in addition, is required to have a face to face session to obtain them.

With this work is intended to analyze the possibility of making the prediction of this episodes using objetive data obtained through mobile devices, such as smartphones and smartbands. That way the data is more accurate and can be obtained remotely and in a non-intrusive way.

These data to be extracted are recordings of the voice and physical activity monitored, because it is thought that they could determine, in part, the mood of the patients and considering how it evolves, it could be possible to develop a system of aid to the decision for specialist.

1.2’. Objetives

The general objetive of this work is to predict crisis in bipolar patients based on physical activity and voice signals analysis. This will serve as a system to help in the medical specialist makingdecision. For this, the following sources of information have been used:

- With voice signals it is intended to identify emotions, specifically, six moods that are: happiness, sadness, fear, disgust, anger and neutral.

- With physical activity data the objetive is identify two types of mental deseases, unipolar depression and bipolar disorder

To achieve this, a series of specific objetives that will mark the deveopment of the work are propounded.

- Study of state of the art, to know the most used and recommended techniques.
- Analysis of phisical activity monitoring devices and audio databases to obtain sets to train the models.
- Procesing activity and voice signals to obtain prepared data to create models.
- Apply machine learning to classify moods through voice and to detect disorders through physical activity.
- Comparation of various tecniques of machine learning to determine the best.
- Discuss the results and compare with other relatated projects.

1.3’. Work plan

To carry out this project, a 5-stage process has been carried out: data collection, data cleaning and preparation, exploratory data analysis, algorithm comparison and the most appropriate recognition of patterns algorithm selection.

In the data collection phase, all the necessary data were collected for the project, from which a final version was obtained in CSV format to facilitate its treatment in future phases.

In the cleaning and preparation phase study of the previously obtained data was carried out, incorrect or empty values are detected, outliers are eliminated, and the data are standardized; all this with the intention of having the data prepared to carry out a thorough analysis in the next phase.

In the exploratory analysis of the data, it was determined which variables were more important when applying the algorithms. For this we looked at the importance that the decision trees gave to each variable and, studying the correlation between the variables, redundancies could be eliminated, and the set of variables reduce, to improve the computing time and complexity of input data.

With these data already prepared, different machine learning algorithms were tested varying the parameters of each one of them to make a comparison of the results obtained. Finally, based on these results was determined which algorithm would be the most suitable for this problem.

The results obtained are satisfactory and provide an information that can be very useful to the episodes prediction in bipolar patients.

1.4’. Technologies used

In the project it has been used Python 3 as programming language, to process the data and apply the algorithms of machine learning, making use of Jupyter Notebook as environment. In addition, the Praat program has been used to obtain the variables from the sound waves from the recorded audios.

1.4.1’. Python

Python is an interpreted programming language administered by the Python Software Foundation. Its use in automatic learning is very widespread because it has libraries that help the treatment (pandas, Scikit-Learn) and visualization (Matplotlib, Seaborn) of the data, in addition to others that provide prediction algorithms (Scikit-Learn).

1.4.2’. Jupyter Notebook

Jupyter Notebook is a web application that consists of "notebooks" that include code, which can be executed by fragments. It also allows the use of Markdown, which is very

useful to comment on certain snippets of code. Another advantage that brings the use of Jupyter Notebook is the ability to visualize results quickly, clearly and easily.

1.4.3'. Praat

Praat (http://www.fon.hum.uva.nl/praat/) is a desktop program available for OSX, Windows and Linux, created by Paul Boersma and David Weenink, professors of the University of Amsterdam. The program allows to analyze an audio wave, making available to the user several tools to obtain the value of different variables like: The pitch (maximum, minimum, medium) or the intensity (maximum, minimum, average); It allows to modify the range of frequencies to edit, to obtain frequencies of specific moments of the wave, to obtain the formants (peak of intensity in the spectrum of a sound since the maximum concentration of energy, amplitude of wave, is given in a certain frequency), and some more advanced features for experts.

1.5'. Repository

The project is published in a repository on GitHub: https://www.github.com/MrDevoid/TFG. The following content can be found here:

1. CSV files used in the project.
2. The Praat scripts used to get the CSV files from the audios.
3. The notebook used to generate the CSV files from the Praat scripts and the audios.
4. The notebook used to train and test the prediction algorithms.

The project is protected under a GNU General Public License v 3.0

1.6'. Final degree project structure

This document describes the process followed during the project. This has been structured in chapters to facilitate reading:

Chapter 1: Introduction of the project which discusses the general aspects of this, technologies and data used, methodology of work, and objectives of the thesis.

Chapter 2: State of the art in which some similar projects are mentioned and the technologies with which they are working. It also makes an analysis of the current applications and devices that allow to monitor certain variables.

Chapter 3: Description of data pre-processing. Firstly, it explains how the data needed for the project is obtained, and an analysis of each dataset is performed, describing the variables of each one of these data. This describe the procedures applied to the data to prepare them for training, such as normalizing or treating atypical values.

Chapter 4: Analysis of the data. In this section we study the variables of the data obtained in Chapter 3. It is observed the correlation that exists between the variables, which variables are more relevant when predicting and which are less relevant, all with the objective of cleaning the datasets and train only with the variables necessary.

Chapter 5: Machine learning algorithms. Description of the algorithms used in the project to predict and how their parameters have been configured for this problem.

Chapter 6: Results obtained when applying the algorithms seen in previous chapter. The results are analyzed, and certain modifications are made to improve its performance, such as modifying the number of exit classes.

Chapter 7: Conclusions and future work. The results and if it is useful to use machine learning for these real problems are discuss, explaining which algorithm or algorithms would be the most appropriate. In addition, there are some ways in which one could move forward in the future.

1.7’. Related career subjects

Throughout the degree we have studied a series of subjects related to Artificial Intelligence and the data and information processing that have given us the necessary skills to carry out this work. The main subjects are:

- Artificial Intelligence, which gave an overview of the symbolic and subsymbolic artificial intelligence present today.

- Artificial Intelligence applied to Control, it was about the use of artificial intelligence algorithms (evolutionary computation, expert systems, neural networks and fuzzy logic) applied to control systems.

- Automatic learning and Big Data, a subject that teaches the foundation of some algorithms such as linear regression and logistics, neural networks, SVM and others used in current problems of data analysis.

- Data mining and the Big Data paradigm, in which we learn techniques for the correct handling, processing and/or exploitation of large volumes of data, as well as the use of an adequate representation of these.

- Cloud and Big Data: mainly focuses on processing large amounts of data using cloud services, frameworks such as Hadoop or Spark on Amazon's cloud services platform.

2. Estado del arte

2.1. Proyectos similares

Se han encontrado algunos trabajos de reconocimiento del habla para detectar el estado del ánimo. Estos utilizan habitualmente una herramienta software para procesar el audio, como PRAAT [7, 8, 17, 22]. Como algoritmo de clasificación lo más habitual es utilizar redes neuronales [5, 7, 12, 14, 15], SVM [7, 9, 12, 14], K-NN [9, 15] y árboles de decisión [14]. Además, en estos trabajos se hace uso de algoritmos de selección de variables y, a diferencia de lo que sucede con los algoritmos de clasificación, no existe un algoritmo que se use de forma habitual.

Sobre el tema de predicción del estado de ánimo a partir de variables como el movimiento o el sueño existen escasos trabajos que traten de determinar las emociones o posibles episodios mediante estás señales.

Algunos de los trabajos que tratan contenidos similares a los que se abordan en este TFG son:

1. **Application of Machine Learning Algorithms for Bipolar Disorder Crisis Prediction [11]:**

Trabajo realizado el año anterior por un compañero de la facultad de Informática de la UCM. Este trabajo utiliza fuentes de datos subjetivas.

2. **Prominence features: Effective emotional features for speech emotion recognition [9]:**

En este artículo se comparan los algoritmos K-Nearest Neighbor (K-NN), Naïve Bayes (NB), Support Vector Machine (SVM) y Partial Least Squares-Discriminatory Analysis (PLS-DA). Utiliza la base de datos Chinese Dual-mode Emotional Speech Database (CDESD). Los mejores resultados son obtenidos por SVM y NB, en torno a un 73% de aciertos.

3. **Emotion in speech: recognition and application to call centers [15]:**

En este artículo se comparan los resultados de los algoritmos K-Nearest Neighbors (KNN), Neural Networks (NN) y Ensembles of Neural Networks. Crean una base de datos pidiendo a varias personas que graben una serie de frases simulando diferentes estados de ánimo. Los resultados obtenidos son de un 55% de aciertos con el algoritmo K-NN, un 65% con NN y un 70% con ENN.

4. **Speech Emotion Recognition: Methods and Cases Study [12]:**

En este artículo se trata de determinar las emociones presentes en el habla, utilizando redes neuronales recurrentes (RNN), regresión lineal multivariable (MLR) y support

vector machines (SVM). Utilizan variables como los MFCC, características de modulación espectral y coeficientes de predicción lineal (LPC). Como bases de datos utilizan Emotional speech synthesis database [4], con audios en español, y Berlín Emotional Database [1]. Los resultados obtenidos son de un 90,05% para la base de datos de audios españoles utilizando las RNN y un 75,90% para la base de datos de audios alemanes utilizando MLR.

5. **The production and recognition of emotions in speech: features and algorithms [14]:**

Estudio donde se comparan una gran cantidad de algoritmos de machine learning, k-NN, decision trees, Kernel density, KStar, Linear regression, LWR, SVM, AdaBoost… Utilizan como datos de prueba una base de datos japonesa con unos 4800 audios. Se centra en estudiar el sentimiento que produce una voz en las personas, con la finalidad de modular los sentimientos de las voces robóticas. Obtienen porcentajes de acierto en torno al 94% con árboles de decisión, SVM y AdaBoost. En este caso las clases posibles para la clasificación son calma, ira, tristeza, confort y alegría.

6. **Statistical Evaluation of Speech Features for Emotions Recognition [8]:**

Artículo donde se utiliza la Berlin Emotional Database, y realizan una clasificación en dos clases, alta excitación (alegría, miedo, ira) y baja excitación (neutral, aburrimiento, asco y tristeza). Se comentan algunas de las variables más relevantes como energía, tono, velocidad del habla, y variables espectrales como los coeficientes cepstrales de mel. Utiliza el algoritmo MLP, obteniendo un 83,17% de aciertos al clasificar en 7 clases, y un 95% al clasificar en alta o baja excitación.

7. **Speech Emotion Classification Using SVM and MLP on prosodic and voice quality features [7]:**

Trabajo donde se compara la clasificación realizada por SVM y MLP de la emoción presente en el habla, haciendo uso de características prosódicas y de la calidad de la voz. La agrupación se hace en 7 clases: alegría, asco, miedo, ira, aburrimiento, tristeza y neutral. Hacen uso de la Berlín Emotional Database y consiguen unos porcentajes de acierto de un 76,82% con SVM y un 78,69% con MLP.

8. **Analysis on speech signal features of manic patients [10]:**

Trabajo que trata de determinar si es posible utilizar la voz para diferenciar pacientes sanos y pacientes maniacos. Para crear el conjunto de datos que utilizan, graban a 30 pacientes maniacos y a 30 sanos para más adelante estudiar el habla con modelos como Random Forest y SVM. Para determinar el tipo de paciente que era cada uno utilizaron BRMS (Bech-Rafaelsdn Mania Rating Scale) y CGI (Clinical Impression Rating Scale).

El objetivo que se persigue con nuestro trabajo consiste en estudiar la posibilidad de predecir los episodios de manía o depresión que sufren las personas con trastornos

bipolares, de forma no intrusiva, es decir, mediante la utilización de dispositivos de uso cotidiano como son los smartphones y las pulseras de monitorización.

Además de estudiar trabajos que realizan un análisis de la voz para determinar las emociones o posibles episodios, es importante saber como se está afrontando actualmente la detección y prevención de estos episodios.

En el artículo *Psychosocial interventions for the prevention of relapse in bipolar disorder: systematic review of controlled trials* [2] se habla de que para prevenir estos episodios se utiliza medicación y añade que una ayuda psicosocial, como son las terapias cognitivo-conductuales y familiares presentan beneficios como complemento de la medicación. Por lo tanto, si se puede predecir que un paciente sufrirá un episodio en un futuro cercano, el médico podría comentar a la familia la necesidad de aplicar dichas terapias para tratar de prevenir la crisis. De esta forma, quizá, podría evitarse el episodio sin ser necesaria una terapia constante con el paciente.

Además, actualmente los métodos para diagnosticar manía se han basado en juicios clínicos como YMRS (Young Mania Rating Scale), BRMS (Bech-Rafaelsdn Mania Rating Scale), MIDI (Mini-International Neuropsychiatric Interview) y MDQ (Mood Disorder Questionnaire) [10], los cuáles utilizan datos subjetivos y, por lo tanto, es posible que no sean del todo certeros o precisos.

2.2. Dispositivos de monitorización

Los dispositivos actualmente en el mercado que pueden utilizarse para la monitorización son: smartbands (pulseras inteligentes) y smartwatches (relojes inteligentes).

Smartbands: algunos de los más utilizados y sus funcionalidades se muestran en la *Figura 1*.

Smartbands	Fitbit Charge 3	Huawei Band 3 Pro	Xiaomi Mi Band 3	Samsung Gear Fit 2 Pro	Garmin Vivosport
					
Marca	Fitbit	Huawei	Xiaomi	Samsung	Garmin
Compatibilidad	Android + iOS	Android + iOS	Android + iOS	Android + iOS	Android + iOS
Autonomía	Hasta 7 días	Hasta 20 días	Hasta 20 días	Hasta 3 días	Hasta 7 días
Precio Medio	150.00 €	99.00 €	29.00 €	199.00 €	199.00 €
Conectividad	Bluetooth 4.0	Bluetooth 4.2	Bluetooth 4.2	Bluetooth 4,2	Bluetooth
	NFC	GPS	NFC	Wifi	ANT
				GPS-GLONASS	GPS
Sensores	Acelerómetro de tres ejes	Sensor óptico de frecuencia cardí	Acelerómetro	Acelerómetro	Acelerómetro
	Monitor óptico del ritmo cardíaco	Acelerómetro en seis ejes	Sensor de frecuencia carc	Barómetro	Altímetro
	Altímetro	Sensor infrarrojos		Giroscopio	Sensor óptico de frecuencia cardíaca
	Motor de vibración			Sensor de frecuencia cardíaca	
	Sensor SpO2 relativo				
Contador de pasos	SI	SI	SI	SI	SI
Calorías quemadas	SI	SI	SI	SI	SI
Distancia recorrida	SI	SI	SI	SI	SI
Pisos subidos	SI	NO	NO	SI	SI
Monitorización del sueño	SI	SI	SI	SI	SI
Medición niveles de estrés	NO	NO	NO	NO	NO
Medición del ritmo cardíaco	SI	SI	SI	SI	SI
Ingesta de calorías	NO	NO	NO	NO	NO
Niveles de hidratación	NO	NO	NO	NO	NO

Figura 1. Tabla comparativa smartbands

Smartwatches: Los relojes inteligentes que permiten la monitorización de una serie de variables de interés para este estudio se muestran en la *Figura 2*.

Smartwatch	Apple Watch Series 4 40mm	Samsung Galaxy Watch 42mm	Huawei Watch 2	Garmin Fenix 5	Fitbit Versa	Amazfit Pace
						
Marca	Apple	Samsung	Huawei	Garmin	Fitbit	Huami
Sistema Operativo	WatchOS 5.0	Tizen OS 4.0	Wear OS	Sistema propio	Fitbit OS	Sistema propio
Compatibilidad	iOS	Android + iOS	Android + iOS	Android + iOS	Android + iOS	Android + iOS
Autonomía	18 horas	3 días	2 días	Hasta 14 días	Hasta 4 días	Hasta 11 días
Precio Medio	429.00 €	309.00 €	285.00 €	549.00 €	199.00 €	125.00 €
Conectividad	Bluetooth 5.0	Bluetooth 4.2	Bluetooth 4.1	Bluetooth Smart	Bluetooth 4.0	Bluetooth 4.0
	Wifi	Wifi	Wifi	Wifi	Wifi	Wifi
	NFC	NFC	NFC	ANT+	NFC	GPS
	GPS	A-GPS	GPS	GPS		GLONASS
	GLONASS	GPS	GLONASS	GLONASS		
	Galileo	GLONASS		Galileo		
	QZSS					
Sensores	Altímetro barométrico	Sensor de frecuencia cardíaca	Altímetro	Acelerómetro	Acelerómetro de tres ejes	Acelerómetro
	Sensor óptico de frecuencia cardíaca	Acelerómetro	Sensor de ritmo cardíaco	Altímetro barométrico	Giroscopio de tres ejes	Sensor de frecuencia cardíaca
	Sensor eléctrico de frecuencia cardíaca	Altímetro barométrico	Acelerómetro	Giroscopio	Monitor óptico del ritmo cardíaco	Giroscopio
	Acelerómetro	Giroscopio	Giroscopio	Brújula	Altímetro	Brújula
	Giroscopio	Sensor de luz ambiental	Barómetro	Termómetro	Sensor de luz ambiental	
	Sensor de luz ambiental		Brújula digital	Sensor de frecuencia cardíaca	Sensor SpO2	
Contador de pasos	SI	SI	SI	SI	SI	SI
Calorías quemadas	SI	SI	SI	SI	SI	SI
Distancia recorrida	SI	SI	SI	SI	SI	SI
Pisos subidos	SI	SI	SI	SI	SI	NO
Monitorización del sueño	NO	SI	SI	SI	SI	SI
Medición niveles de estrés	NO	SI	NO	SI	NO	NO
Medición del ritmo cardíaco	NO	SI	SI	SI	SI	SI
Ingesta de calorías	NO	SI	SI	SI	SI	NO
Niveles de hidratación	NO	SI	NO	NO	NO	NO

Figura 2. Tabla comparativa smartwatches

2.3. Aplicaciones de monitorización

Las aplicaciones que existen a día de hoy para la monitorización se suelen dividir en aplicaciones para monitorizar el sueño, la actividad física y el audio. También se realiza un análisis de algunas aplicaciones que ayuden a obtener los datos de la forma más adecuada.

2.3.1. Aplicaciones de monitorización de la actividad física

De la misma manera que para el sueño, se han analizado diversas aplicaciones que monitorizan la actividad física para seleccionar la más adecuada.

- **Google Fit:** Aplicación gratuita de Google. Monitoriza en segundo plano las actividades que realice el usuario llevando el terminal consigo (actividades de cardio). Además, podrá introducir manualmente actividades si en ese momento no tenía consigo el teléfono, o bien, si va a realizar un entrenamiento específico.

Solo en el caso de añadir una actividad será necesario introducir datos adicionales, como hora de inicio y fin, tipo de actividad, calorías, pasos y kilómetros.

De forma gráfica la aplicación muestra los pasos, calorías consumidas y kilómetros recorridos, todo ello referido al día en curso. También muestra un resumen de las calorías gastadas en los últimos 7 días, información sobre la frecuencia cardiaca del usuario, y el peso; estos dos últimos pueden ser representados de forma gráfica en distintos niveles de especificidad como diaria, semanal, mensual o anual.

Ofrece la posibilidad de exportar un csv con los datos. Para ello el usuario ha de ir a la siguiente página: https://takeout.google.com/settings/takeout/downloads, crear un archivo, seleccionar sólo los datos de Fit, pulsar en siguiente, configurar las opciones del archivo al gusto del usuario y pulsar en continuar. Tras esto se obtiene una carpeta con

bastante información. Los csv, que se encuentran en Takeout > Fit > Agregaciones diarias, contienen las siguientes variables:

1. Hora de inicio (HH:MM:SS)
2. Hora de finalización (HH:MM:SS)
3. Calorías (kilocalorías)
4. Distancia (metros)
5. Frecuencia cardiaca media (ppm)
6. Frecuencia cardiaca máxima (ppm)
7. Frecuencia cardiaca mínima (ppm)
8. Latitud baja (grados)
9. Longitud baja (grados)
10. Latitud alta (grados)
11. Longitud alta (grados)
12. Velocidad media (m/s)
13. Velocidad máxima(m/s)
14. Velocidad mínima (m/s)
15. Recuento de pasos
16. Peso medio (kilogramos)
17. Peso máximo (kilogramos)
18. Peso mínimo (kilogramos)
19. Duración de inactividad (milisegundos)
20. Duración de andar (milisegundos)

Cada csv corresponde a un día concreto, y las horas de inicio y finalización corresponden a intervalos de 15 minutos, lo que implica que muchas casillas del archivo se encuentran vacías.

Las unidades de las variables son las mismas para todas las aplicaciones y dispositivos utilizados en este trabajo. Y en caso de existir alguna diferencia se especifica en la variable correspondiente.

- **MiFit:** Está aplicación permite la monitorización del ejercicio físico. La monitorización de actividad se realizará de forma automática, y se guardará en la aplicación como actividades, mostrando los pasos, distancia recorrida, y calorías quemadas. Por otro lado, es posible registrar un entrenamiento, como correr al aire libre, en cinta, caminar o ciclismo, y para ello será necesario ir a la pestaña de entretenimiento y pulsar en “Go”. Para registrar otro tipo de entrenamientos será necesario un dispositivo de Xiaomi (pulsera de actividad MiBand).

Para obtener los datos referentes a la actividad física bastará con realizar la operación comentada anteriormente para esta aplicación: Perfil > Ajustes > Acerca de > Ejercer los derechos del usuario > Exportar datos. Tras esto el usuario deberá introducir su cuenta y seleccionar los datos que desea obtener, en este caso los de actividad y deporte y, opcionalmente, los de frecuencia cardiaca. Tras unos minutos recibirá un correo con un link para descargar sus datos. En los distintos csv aparecerán las siguientes variables.

En el archivo de actividades:

1. Fecha
2. Último momento de sincronización (segundos desde el 1 de enero de 1970)
3. Pasos
4. Distancia
5. Distancia corriendo
6. Calorías quemadas

En el archivo de entrenamiento:

1. Tipo
2. Tiempo de inicio
3. Duración del ejercicio
4. Distancia
5. Velocidad máxima
6. Velocidad mínima
7. Velocidad media
8. Calorías quemadas

En el archivo de frecuencia cardiaca:

1. Fecha
2. Último momento de sincronización (segundos desde el 1 de enero de 1970)
3. Frecuencia cardiaca
4. Timestamp

- **Samsung Health:** Aplicación gratuita de Samsung. Monitoriza un conjunto de variables de forma automática (pasos, distancia, calorías quemadas. Es importante comentar que, al igual que sucedía con la aplicación anterior, la posesión de un dispositivo de la marca Samsung pondrá a disposición del usuario un conjunto más amplio de variables, aunque algunas de ellas están disponibles para introducirlas de forma manual.

Entre las variables que permite monitorizar encontramos algunas típicas como pasos, distancia, entrenamientos, el sueño (en caso de no disponer del dispositivo será necesario introducirlo manualmente) o tiempo de actividad. Otras más inusuales son la comida ingerida a lo largo del día, los vasos de agua o café, glucosa en sangre y tensión arterial (han de introducirse manualmente), y otras para las que será necesario un dispositivo Samsung como la saturación de oxígeno, estrés y frecuencia cardiaca.

Para motivar al usuario la aplicación otorga ciertas recompensas por realizar ejercicio diario, dormir bien, por el número de pasos…

Además de todo esto es posible extraer los datos en varios archivos csv. En cada uno de ellos se registra una de las variables anteriores. Como ofrece muchas variables que otras aplicaciones no monitorizan, se hará una descripción sólamente de aquellas más comunes:

Las variables más destacables de la frecuencia cardiaca son:

1. Fecha de creación
2. Número de latidos del corazón
3. Fin de la monitorización
4. Frecuencia cardiaca

De la cuenta de los pasos:

1. Fecha de creación
2. Distancia
3. Número de pasos
4. Velocidad
5. Calorías
6. Fin de la monitorización

De la actividad diaria:

1. Mayor tiempo inactivo
2. Fecha de creación
3. Distancia
4. Tiempo corriendo
5. Número de pasos
6. Tiempo de paseo
7. Hora del día
8. Tiempo activo
9. Mayor tiempo activo
10. Calorías

- **Strava:** Aplicación completamente gratuita. Está orientada a la monitorización de entrenamiento cardiovascular. Ofrece tres pestañas principales para carrera, bicicleta y natación.

A diferencia de las aplicaciones anteriores no es necesario la posesión de un dispositivo para disponer de todas las funcionalidades, aunque sí permite la sincronización con distintas pulseras para mejorar la monitorización.

En las actividades, la aplicación muestra el tiempo de la actividad, la distancia recorrida, el ritmo y la ruta seguida. Para exportar los datos se debe ingresar la sesion en strava.com, ir a configuración, mi cuenta, y en el apartado de descarga o elimina tu cuenta pulsar en comenzar; a continuación, introducir el correo y solicitar el archivo. Es

importante remarcar que sólo se podrá realizar una petición por semana. El archivo que se entrega al usuario contiene varios csv, pero de interés sería "activities.csv" que contiene las siguientes variables:

1. Identificador
2. Fecha
3. Nombre
4. Tipo
5. Descripción
6. Tiempo empleado
7. Distancia
8. Conmute (opción de viaje, actualmente en desuso)
9. Marcha

- **Runtastic:** Aplicación gratuita que goza de una versión premium que incluye creación de rutas, ausencia de anuncios y algunas funcionalidades extra.

Permite la monitorización de distintos tipos de actividades, pero siempre será necesario indicar el comienzo de la actividad. Si se desea que la aplicación monitorice en segundo plano sin necesidad de indicar el comienzo de una actividad puede descargarse otra aplicación de Runtastic que además ofrece la posibilidad de sincronizarse con Google Fit y dispositivos de la marca Runtastic.

Las variables monitorizadas en la aplicación principal son: distancia, duración, calorías, ritmo medio, velocidad media y máxima, elevación del terreno y hora de inicio. En el caso de que algunos de los datos anteriores no tengan sentido en alguna actividad estos datos no aparecerán.

En cuanto a la versión que permite la monitorización en segundo plano, mostrará información sobre la distancia recorrida, calorías quemadas, minutos activos y número de pasos. Además, presentará una gráfica con los distintos días de la semana sobre los que se podrá pulsar para ver información acerca de ese día en concreto.

Para extraer los datos se debe ir a la página de runtastic.com, iniciar sesión, ir a ajustes y pulsar en exportar. En esta pestaña podremos descargar los últimos datos solicitados o solicitar un nuevo paquete de datos, aunque sólamente se podrá solicitar un conjunto de datos por semana. En el archivo csv de las actividades aparecen las siguientes variables:

1. Fecha de inicio (segundos desde el 1 de enero de 1970)
2. Fecha de fin (segundos desde el 1 de enero de 1970)
3. Fecha de creación (segundos desde el 1 de enero de 1970)
4. Fecha de actualización (segundos desde el 1 de enero de 1970)
5. Distancia
6. Duración
7. Ganancia de elevación

8. Pérdida de elevación
9. Velocidad media
10. Calorías quemadas
11. Tiempo por kilómetro
12. Temperatura
13. Frecuencia cardíaca
14. Frecuencia cardiaca máxima
15. Manual/Automático (si la actividad ha sido añadida manualmente o no)
16. Completada
17. Tipo de deporte

Comentar que los datos descargados se almacenan en carpetas y cada actividad aparece en un archivo .json diferente. Será necesario transformarlo a csv utilizando algún convertidor como el siguiente: http://convertcsv.com/json-to-csv.htm

- **Garmin Connect:** Aplicación gratuita. Para la captura de las actividades la posesión de un dispositivo de la marca Garmin es necesaria debido a que, sin éste, la aplicación sólo permite añadir actividades de forma manual y muchas variables no pueden ser medidas por el usuario.

De forma gráfica la aplicación muestra un pequeño resumen de las tareas con algunos datos sobre ellas. El tipo de actividad puede representar un problema ya que en función de ésta cambian algunas unidades de medida de las variables, como es el caso de la velocidad media, que en actividades como "Carrera" está representada en minutos por kilómetro y en "Senderismo" aparece como millas por hora.

Para extraer los datos en formato csv bastará con iniciar sesión en la página de https://connect.garmin.com/es-ES/signin, pulsar en "Actividades" en el panel situado a la izquierda, y después pulsar sobre "Exportar a CSV" situado arriba a la derecha. Las variables que ofrece el CSV por cada actividad son:

1. Tipo
2. Fecha de inicio
3. Distancia
4. Duración
5. Velocidad media
6. Longitud media de zancada
7. Relación vertical media
8. Oscilación vertical media
9. Dificultad
10. Intervalo en superficie
11. Descompresión (Si/No)

Tras este análisis hay que comentar que en este trabajo no se ha utilizado ninguna de éstas, debido a la base de datos construida por los integrantes del grupo no era lo suficientemente grande como para que los modelos produzcan buenos resultados y por

ello se ha decidido hacer uso de una base de datos pública; pero la más adecuada para realizar un trabajo similar sería Google Fit, ya que agrega los datos cada 15 minutos, lo que permite tener un amplio margen para trabajar con distintos niveles de granularidad. En un futuro si esta base de datos creciese podrían utilizarse los datos adquiridos mediante esta aplicación.

2.4. Dispositivos de grabación

Para grabar la voz existen numerosas aplicaciones, por lo que será suficiente con utilizar una que capte el sonido de forma clara, sin añadir mucha distorsión, y que en el audio grabado se escuche en primer plano a la persona cuya voz va a ser analizada, con el menor ruido de fondo posible. Lo único relevante será el formato del archivo que debe estar en ".wav" para que pueda ser procesado por Praat.

Un ejemplo de aplicación que podría utilizarse es Grabadora de Voz Fácil (https://play.google.com/store/apps/details?id=com.coffeebeanventures.easyvoicerecorder&hl=es419), ya que extrae el archivo en .wav, que es necesario para poder tratarlo con Praat.

La obtención de las variables necesarias para el análisis de los datos se realizará mediante un script de Praat.

3. Preprocesamiento de datos

3.1. Obtención de los datos

Los datos utilizados proceden de diversas fuentes que se exponen a continuación:

Los datos de audio proceden de dos bases de datos públicas. Una de ellas contiene alrededor de 500 audios en alemán (Berlín database of emotional speech [1]) y la otra unos 1500 audios en español (Emotional speech synthesis database [4]), y ambas clasifican estos audios dependiendo de la emoción que se expresa en ellos.

Los datos de actividad física se obtienen de una página web (The Depresjon Dataset [25]), los cuales contienen el registro de actividad de 23 pacientes con trastorno bipolar y depresión de varios días.

3.1.1. Obtención de los datos de audio

Para obtener la información necesaria se han utilizado dos bases de datos (Berlín database of emotional speech [1] y Emotional speech synthesis database [4]). La base de datos de audios alemanes está formada por unos 500 audios, grabados por 10 personas (5 hombres y 5 mujeres) de entre 25 y 35 años. La clasificación se hace con 7 emociones (ira, aburrimiento, asco, miedo, felicidad, tristeza y neutral). Para realizar las grabaciones los participantes debían grabar 10 frases en cada uno de los estados de ánimo mencionados.

La base de datos de audios españoles tiene cerca de 1500 audios grabados por una mujer y un hombre que leían una serie de frases con distintos estados de ánimo (alegría, tristeza, asco, miedo, ira y distintas modalidades de neutral, como neutral rápido o lento). La duración total de los audios es de unas 8 horas.

Como resultado se tienen los siguientes conjuntos que se utilizarán en el entrenamiento y determinarán las diferencias que se producen en la predicción con cada uno de ellos: sonidosAleman.csv, sonidosMezclados.csv, y sonidosEspanol.csv.

Para obtener las variables a partir de un archivo de audio se utiliza el programa Praat mediante unos scripts que crearán el archivo csv correspondiente al conjunto de audios que se encuentren en las carpetas adecuadas. En total son utilizados 4 scripts: *ScriptCrearCsv.praat, TransformarAudio.praat, ObtenerVariables.praat y CalculoSpeechRate.praat*. El primero se utiliza para obtener el archivo .csv de los audios que se utilizan para la fase de entrenamiento; el segundo para obtener el archivo .csv correspondiente a los audios para realizar las pruebas, y los dos restantes son scripts auxiliares que obtienen las diferentes variables que se comentarán a continuación. Para el correcto funcionamiento del script *ScriptCrearCsv.praat* será necesario la creación de 6 subdirectorios, los cuáles serán denominados *Alegría, Tristeza, Miedo, Asco, Ira y Neutral*, y en cada uno de ellos se introducirán los audios correspondientes a esa emoción. Por último, se indicará la ruta del directorio que contiene esos 6 subdirectorios. Para

obtener el .csv de los audios de prueba bastará con indicar el directorio en el que se encuentran todos los audios.

Cabe comentar que los audios en español tenían un formato .l16, y para transformarlo a .wav se utiliza un script de praat: *TransformarAudiosEspanol.praat* que carga los archivos y los guarda en .wav en los directorios adecuados, de forma que basta con aplicar *ScriptCrearCsv.praat* a todos los audios de forma genérica para extraer las variables.

A continuación se realiza una breve descripción de todas las variables utilizadas en el análisis de audios:

- ➔ Amplitude (mean, min, max, std, range): Mide la amplitud de la señal, es decir, la diferencia entre el punto más alto de una onda y su base. Los valores de esta variable se encuentran en los siguientes intervalos:

  - ◆ Mean (Pascal): [-0.007, 4.5e-5]
  - ◆ Minimum (Pascal): [-1.3, -0.06]
  - ◆ Maximum (Pascal): [0.5, 1.3]
  - ◆ Standard deviation (Pascal): [0.008, 0.3]
  - ◆ Range (Pascal): [0.1, 2.31]
    *Pascal = kg /(m*s^2) = Newton/metro^2*

- ➔ Pitch (mean, min, max, std, range): Se refiere a la frecuencia fundamental (primer armónico) que se obtiene al aplicar el análisis de Fourier a la onda sonora. Los valores de esta variable se encuentran en los siguientes intervalos:

  - ◆ Mean (Hz): [90.5, 441]
  - ◆ Minimum (Hz): [65.5, 312]
  - ◆ Maximum (Hz): [108.2, 635.7]
  - ◆ Standard deviation (Hz): [6.6,198.2]
  - ◆ Range (Hz): [33.2, 556]

- ➔ Harmonicity (mean, min, max, std, range): Representa el grado de periodicidad acústica, expresada en dB. Los valores de esta variable se encuentran en los siguientes intervalos:

  - ◆ Mean (dB): [0.2, 20.2]
  - ◆ Minimum (dB): [-229, -220]
  - ◆ Maximum (dB): [10, 60]
  - ◆ Standard deviation (dB): [2,10]
  - ◆ Range (dB): [245, 290]

➔ Intensity (mean, min, max, std, range, quantile): Representa la intensidad en puntos de tiempo separados linealmente, expresada en dB. Los valores de esta variable se encuentran en los siguientes intervalos:

- Mean (dB): [40, 77.6]
- Minimum (dB): [12, 61]
- Maximum (dB): [61, 80]
- Standard deviation (dB): [3, 99]
- Range (dB): [14.5, 60]
- Quantile (dB): [40, 82]

➔ Speech rate (Sílabas/segundo): Mide la velocidad y densidad del habla por unidad de tiempo. Los valores de esta variable se encuentran en el intervalo [0, 7]

➔ Articulation rate (Sílabas/segundo): Obtiene una medida de la velocidad del habla donde se excluyen del cálculo todas las pausas. Los valores de esta variable se encuentran en el intervalo [0, 7]

➔ Energy (mean, min, max, std, range): Energía que transmiten las ondas sonoras, la cual procede de la vibración del foco sonoro. Los valores de esta variable se encuentran en los siguientes intervalos:

- Mean (Pascal^2 s): [400, 1600]
- Minimum (Pascal^2 s): [-600, 1000]
- Maximum (Pascal^2 s): [1100, 2200]
- Standard deviation (Pascal^2 s): [110, 600]
- Range (Pascal^2 s): [365, 2400]

➔ MFCC 1-12 (mean, min, max, std, range) [20, 22]: Representan los coeficientes cepstrales de frecuencia de Mel en función del tiempo. Estos coeficientes están basados en la percepción auditiva humana y se calculan a partir de la Transformada de Fourier (TF) o la Transformada del Coseno Discreta. Además, los MFCC son menos susceptibles a las variaciones provocadas por la condición física de los oradores. Los valores de esta variable, en la escala de Mel, se encuentran en los siguientes intervalos (ver *Figura 3)*:

	Mean	**Minimum**	**Maximum**	**Standard deviation**	**Range**
MFCC 1	[0,400]	[-350, 200]	[180, 600]	[40, 175]	[200, 750]
MFCC2	[-170, 93]	[-400, -25]	[70, 350]	[30, 155]	[170, 700]
MFCC3	[-35, 120]	[-230, -5]	[100, 360]	[25, 110]	[130, 500]
MFCC4	[-90, 80]	[-270, -40]	[40, 200]	[25, 80]	[150, 400]
MFCC5	[-60, 60]	[-220, -20]	[40, 200]	[20, 70]	[120, 360]
MFCC6	[-80, 35]	[-240, -50]	[10, 170]	[20, 70]	[110, 350]
MFCC7	[-100, 25]	[-210, -40]	[10, 140]	[20, 50]	[100, 300]
MFCC8	[-50, 30]	[-160, -30]	[25, 160]	[20, 45]	[95, 260]

MFCC9	[-70, 25]	[-180, -30]	[0, 125]	[15, 50]	[90, 250]
MFCC10	[-40, 30]	[-150, -20]	[25, 140]	[15, 40]	[80, 250]
MFCC11	[-50, 20]	[-160, -30]	[15, 130]	[10, 45]	[80, 250]
MFCC 12	[-25, 40]	[-120, -20]	[25, 140]	[10, 40]	[60, 225]

Figura 3. Rango valores variables MFCC

Como se puede observar los intervalos son bastante amplios, debido a la presencia de audios en diferentes idiomas, con una entonación más grave o más aguda.

La escala de Mel es una escala de tonos juzgados como intervalos equiespaciados, y su equivalencia es la siguiente, con f (Hz):

$$m = 1127{,}01048 \times log_e\,(1 + f/700) \qquad (1)$$

➔ Formantes 1-5 (mean, min, max, std, range, quantile): Representa la estructura espectral en función del tiempo. Los valores de esta variable se encuentran en los siguientes intervalos en Hercios (ver *Figura 4*):

	Formante 1	**Formante 2**	**Formante 3**	**Formante 4**	**Formante 5**
Mean	[500, 1100]	[1600, 2320]	[2600, 3300]	[3500, 4500]	[4400, 5200]
Minimum	[0, 300]	[100, 1300]	[1200, 2600]	[2200, 3800]	[3300, 4800]
Maximum	[1200, 3100]	[2600, 4400]	[3500, 5300]	[4800, 5700]	[5200, 5600]
Standard deviation	[150, 650]	[250, 720]	[250, 660]	[200, 700]	[150, 520]
Range	[1000, 3000]	[1500, 4000]	[1500, 3800]	[1500, 3400]	[750, 2100]
Quantile	[350, 1300]	[1400, 2500]	[2500, 3400]	[3500, 4600]	[4400, 5200]

Figura 4. Rango valores variables Formante

Al igual que sucedía en los MFCC, los rangos son muy amplios por el mismo motivo.

3.1.2. Obtención de los datos de actividad física

En relación con la actividad física se han utilizado los datos obtenidos de The Depresjon Dataset en formato CSV.

Los datos han sido recolectados mediante un reloj (*Actiwatch, Cambridge Neurotechnology Ltd, England, model AW4*) que mide los niveles de actividad. La frecuencia de muestreo es de 32 Hz y se registran movimientos por encima de 0.05 g. El número perteneciente a la actividad es proporcional a la intensidad del movimiento. El total de la actividad está siendo continuamente registrada en intervalos de 1 minuto.

Los datos de actividad corresponden a 23 pacientes con depresión o trastorno bipolar que usaron este dispositivo en su mano derecha.

A continuación, se muestra un análisis de los campos que contiene *scores.csv*:

- ➔ number: Identificador del paciente
- ➔ days: número de días de la medición
- ➔ gender: 1 (mujer), 2 (hombre)
- ➔ age: edad agrupada en grupos de edad
- ➔ afftype: 1 (bipolar II), 2 (depresivo unipolar), 3 (bipolar I)
- ➔ melanch: 1 (melancolía), 2 (no melancolía)
- ➔ inpatient: 1 (paciente interno), 2 (paciente externo)
- ➔ edu: (educación agrupada en años)
- ➔ marrieage: 1 (casado/a o con pareja), 2 (soltero/a)
- ➔ work: 1 (trabajando o estudiando), 2 (desempleado / baja por enfermedad / jubilado)
- ➔ madrs1: puntuación MADRS cuando empieza la medición
- ➔ madrs2: puntuación MADRS cuando finaliza la medición

Dentro de la carpeta *condition* hay 23 ficheros CSV relativos a cada uno de los pacientes analizados. A continuación se muestran los campos que contiene cada fichero:

- ➔ timestamp: fecha y hora en intervalos de 1 minuto en el siguiente formato: AAAA-MM-DD HH:MM:SS
- ➔ date: fecha de la medición en el siguiente formato: AAAA-MM-DD
- ➔ activity: medición del nivel de actividad

3.2. Limpieza de datos

3.2.1. Limpieza y preparación de los datos de audio

Una vez obtenidos los csv, como se ha explicado anteriormente, se procede a su carga haciendo uso de la función read_csv() proporcionada por la librería *Pandas*. Una vez cargados los datos, se separa cada csv en dos variables: variable X, que contendrá las columnas correspondientes a las variables de entrada con una fila por cada ejemplo, y la variable Y, que contendrá la clase a la que pertenece cada ejemplo. Para asociar un ejemplo con su clase se hará uso del número de la fila en la que se encuentre dicho ejemplo, por lo que es importante no modificar el orden de las filas durante el proceso de entrenamiento (*Ver Figuras 5 y 6*).

```
audiosMezclados = pd.read_csv('sonidosMezclados.csv', na_values='--undefined--')
audiosAleman = pd.read_csv('sonidosAleman.csv', na_values='--undefined--')
audiosEspanol = pd.read_csv('sonidosEspanol.csv', na_values='--undefined--')
```

Figura 5. Lectura archivos CSV

```
X_Mezclados = audiosMezclados.drop(['Result'], axis=1)
y_Mezclados = audiosMezclados.Result

X_Aleman = audiosAleman.drop(['Result'], axis=1)
y_Aleman = audiosAleman.Result

X_Espanol = audiosEspanol.drop(['Result'], axis=1)
y_Espanol = audiosEspanol.Result
```

Figura 6. Selección variables entrada y salida

Con las variables de entrada y salida preparadas, se aplica en todos los conjuntos de datos una función que eliminará los valores nulos sustituyéndolos por la media aritmética de la variable correspondiente, de forma que a partir de este momento el conjunto de datos pueda tratarse sin ese problema. La función se muestra en la *Figura 7.*

```
def deleteNAWithMean(df,mean):
    for i in df.columns.values:
        df[i].replace(np.nan, mean[i], inplace = True)
    return df
```

Figura 7. Implementación de la función deleteNAWithMean en código Python

Otro tipo de datos problemáticos son los outliers o valores atípicos que son valores numéricamente muy distantes al resto de datos de una muestra, y que pueden llevar a tener un peor resultado en la predicción. Estos pueden ser fruto de una mala medición de la variable o puede que ese ejemplo sea un caso excepcional, por lo que a la hora de tratarlos se deberá tener en cuenta estos posibles orígenes. Para detectarlos se utilizará el rango intercuartílico y los cuartiles 1 y 3, aplicando la siguiente fórmula:

$$\boldsymbol{Lower\ limit} = \boldsymbol{Q1} - \boldsymbol{1.5} \times \boldsymbol{IQR}$$
$$\boldsymbol{Upper\ limit} = \boldsymbol{Q3} + \boldsymbol{1.5} \times \boldsymbol{IQR} \qquad (2)$$

De esta forma un valor será atípico sí es menor que Lower limit o mayor que Upper limit, y dado su origen, se considera que lo más conveniente sería eliminar estos ejemplos del entrenamiento para no empeorar el resultado. La supresión de outliers se lleva a cabo de la forma que muestra la *Figura 8.*

```
def deleteOutliers(data,outdata):
    lowerOutliers = np.array([False])
    upperOutliers = np.array([False])
    for i in data.columns.values:
        quartiles = data[i].quantile([0.25,0.75])
        IQR = quartiles[0.75]-quartiles[0.25]
        lowerLimit = quartiles[0.25] - 1.5*IQR
        upperLimit = quartiles[0.75] + 1.5*IQR
        lowerOutliersAux = data[i].values < lowerLimit
        upperOutliersAux = data[i].values > upperLimit
        lowerOutliers = np.logical_or(lowerOutliers, lowerOutliersAux)
        upperOutliers = np.logical_or(upperOutliers, upperOutliersAux)
    outliers = np.logical_or(lowerOutliers, upperOutliers)
    data = data.drop(np.where(outliers)[0])
    outdata = outdata.drop(np.where(outliers)[0])
    data = data.reset_index(drop=True)
    outdata = outdata.reset_index(drop=True)
    return data, outdata
```

```
X_MezcladosNorWithoutOutliers,y_MezcladosWithoutOutliers=deleteOutliers(X_MezcladosReduced,y_Mezclados)
X_AlemanNorWithoutOutliers, y_AlemanWithoutOutliers=deleteOutliers(X_AlemanReduced, y_Aleman)
X_EspanolNorWithoutOutliers, y_EspanolWithoutOutliers=deleteOutliers(X_EspanolReduced, y_Espanol)
```

Figura 8. Implementación de la función deleteOutliers en código Python

Sin embargo, al aplicar esto se eliminan más del 50% de los datos, por lo que se decidió realizar modificaciones para no suprimir tal cantidad de información, puesto que en las pruebas el rendimiento de los algoritmos se reducía considerablemente. Una de las medidas tomadas consiste en realizar esta supresión tras eliminar las variables menos relevantes; con ello algunos de los ejemplos antes eliminados permanecían ahora en el conjunto de datos. Además de esto, dado que el número de ejemplos eliminados aún es cuantioso, se amplía el rango de valores no considerados outliers, multiplicando el IQR por valores mayores. Finalmente, tras realizar varias pruebas se determinó que 2.5 era un valor adecuado para este problema, puesto que de esta forma se eliminaban los ejemplos cuyos valores eran realmente extremos.

Uno de los problemas que aparecen con el conjunto de datos de audio es la heterogeneidad en las unidades de medida, lo que provoca grandes variaciones en los valores de las diferentes variables. Por lo tanto, se ha de aplicar una normalización previa al tratamiento. Se han analizado distintas alternativas de normalización, y a continuación se detallan y ejemplifican mostrando gráficamente el efecto que tiene cada una, con la ayuda de dos gráficas (azul: antes y amarillo: después).

- **Escalado de variables (MinMax Scaler):** Está normalización se realiza aplicando la siguiente fórmula:

$$X_{normalized} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (3)$$

Está normalización transforma los datos originales al dominio [0-1], siendo 0 el valor mínimo y 1 el valor máximo. Podría presentar algunos problemas debido a la presencia de outliers y además distorsionar mucho señales de audio estables. Las *Figuras 9 y 10* muestran unos datos de ejemplo antes y después de la normalización.

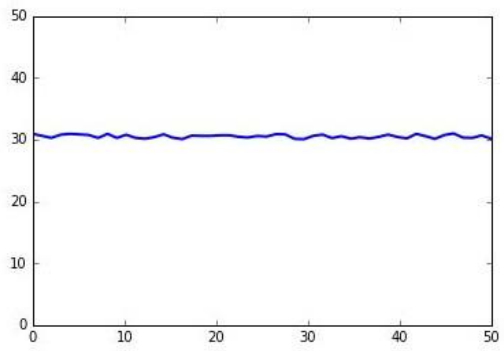

Figura 9. Señal de audio sin normalizar

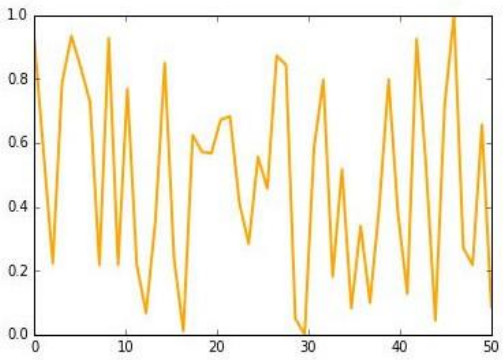

Figura 10. Señal de audio normalizada con MinMax Scaler

- **Z-Score:** Esta normalización se realiza aplicando la siguiente fórmula:

$$X_{normalized} = \frac{X - X_{mean}}{X_{stddev}} \quad (4)$$

Está normalización transforma los datos para que sigan una distribución normal de media 0 y desviación típica 1. En este caso no se consigue agrupar todos los datos en un conjunto de 0 a 1 y además los outliers pueden variar la media de forma considerable y, por ende, provocar la pérdida de información. Las *Figuras 11 y 12* muestran unos datos de ejemplo antes y después de la normalización.

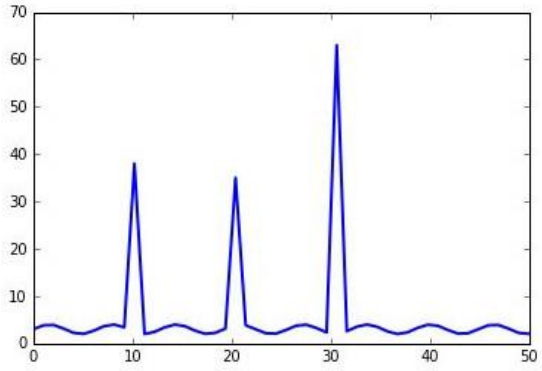

Figura 11. Señal de audio sin normalizar

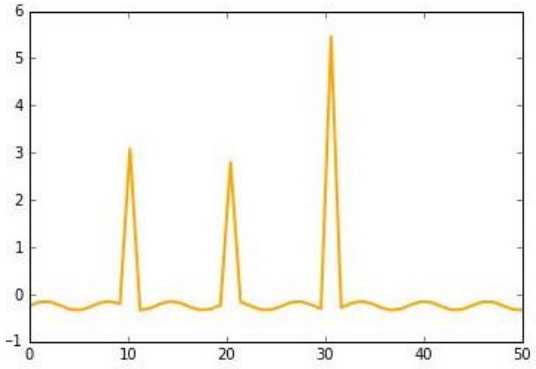

Figura 12. Señal de audio normalizada con Z-Score

- **Escalado robusto (Robust Scaler):** Esta normalización se aplica teniendo en cuenta el rango intercuartílico de forma que sea más robusta a outliers:

$$X_{normalized} = \frac{X - Q_1(X)}{Q_3(X) - Q_1(X)} \qquad (5)$$

Al igual que sucede con la normalización estándar los datos no se limitarán a un rango [0-1] y, además, puesto que utiliza menos información de los datos iniciales, es recomendable utilizar esta normalización cuando existen varios outliers en los datos. Las *Figuras 13 y 14* muestran unos datos de ejemplo antes y después de normalizar.

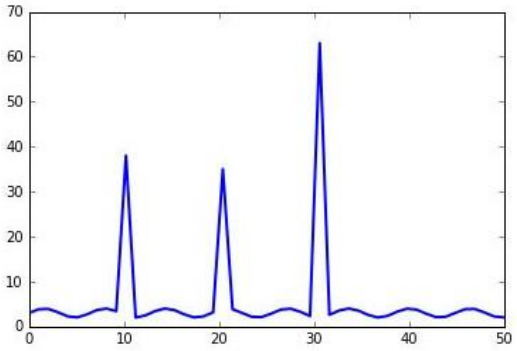

Figura 13. Señal de audio sin normalizar

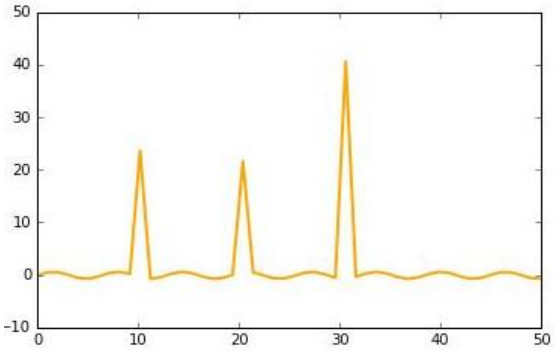

Figura 14. Señal de audio normalizada con Robust Scaler

- **Normalizer:** Para aplicar esta normalización primero se obtiene la norma (l1, l2 o max) de cada variable y posteriormente se normaliza dividiendo entre la norma de la columna obtenida anteriormente:

$$Norm_{L1} = \sum_{i=1}^{n} | X_i | \quad (6)$$

$$Norm_{L2} = \sqrt{\sum_{i=1}^{n} | X_i |^2} \quad (7)$$

$$Norm_{MAX} = max(|x_1|, |x_2|, \ldots, |x_n|) \quad (8)$$

$$X_{Normalized} = \frac{X}{Norm_X} \quad (9)$$

En este caso debemos seleccionar una norma, y cada una de ellas presenta una serie de características. Por un lado, la norma l1 es más robusta mientras que la norma l2 es más estable y tiene un menor coste computacional, dado que l1 no puede resolverse en términos matriciales. La norma max es muy susceptible a posibles valores atípicos. Las *Figuras 15 y 16* muestran ejemplos antes y después de normalizar mediante la norma l2, que es la norma seleccionada para utilizar debido a su estabilidad y coste computacional.

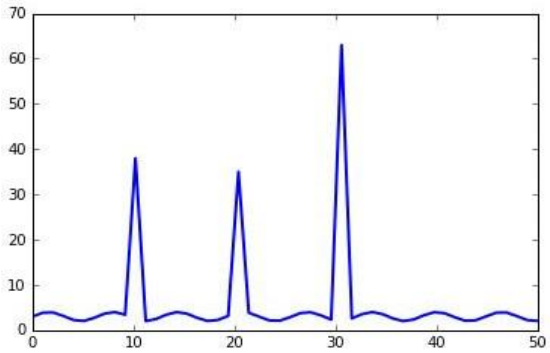

Figura 16. Señal de audio sin normalizar

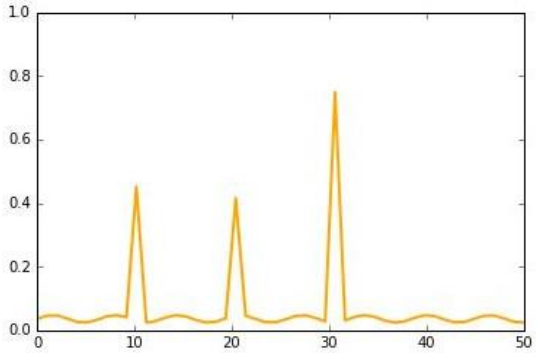

Figura 15. Señal de audio normalizada con Normalizer

Para aplicar estas normalizaciones se hace uso de la librería sklearn.preprocessing y de un método definido por nosotros mismos (Escalado estándar). Primero se ajusta la media y desviación al conjunto de entrenamiento para poder hacer uso de ellas en el momento de testear los algoritmos con los datos de prueba; tras eso se transforman los datos de entrada (Ver *Figura 17*).

Normalización utilizando el método MinMax, que convierte los valores a un intervalo de 0-1

```
scaler = MinMaxScaler()
scalerMezclados = scaler.fit(X_Mezclados)
scalerAleman = scaler.fit(X_Aleman)
scalerEspanol = scaler.fit(X_Espanol)
minMaxScaled_Mezclados = scalerMezclados.transform(X_Mezclados)
minMaxScaled_Aleman = scalerAleman.transform(X_Aleman)
minMaxScaled_Espanol = scalerEspanol.transform(X_Espanol)
```

Normalización ZScore, para transformar los datos a una normal

```
normalizar_Mezclados = normalizar(X_Mezclados,meanMezclados,stdMezclados)
normalizar_Aleman = normalizar(X_Aleman,meanAleman,stdAleman)
normalizar_Espanol = normalizar(X_Espanol,meanEspanol,stdEspanol)
```

Normalización utilizando la norma l1, l2 o max, según se especifique en los argumentos

```
normalized_Mezclados, normMezclados = normalize(X_Mezclados, norm = 'l2',axis=0,return_norm=True)
normalized_Aleman, normAleman = normalize(X_Aleman, norm='l2',axis=0,return_norm=True)
normalized_Espanol, normEspanol = normalize(X_Espanol, norm='l2',axis=0,return_norm=True)
```

Normalización realizada con el rango interquartílico, de forma que sea más robusta frente a valores atípicos

```
robustScaler = RobustScaler()
robustScalerMezclados = robustScaler.fit(X_Mezclados)
robustScalerAleman = robustScaler.fit(X_Aleman)
robustScalerEspanol = robustScaler.fit(X_Espanol)
robustScaled_Mezclados = robustScalerMezclados.transform(X_Mezclados)
robustScaled_Aleman = robustScalerAleman.transform(X_Aleman)
robustScaled_Espanol = robustScalerEspanol.transform(X_Espanol)
```

Figura 17. Aplicación de las normalizaciones en código Python

Además de estas transformaciones, para preparar los datos para el entrenamiento se añade la columna de resultado (variable “Result”), en la que se codifica el valor de salida del audio correspondiente. Está codificación se hace de la siguiente manera: 0 - Alegría, 1 - Tristeza, 2 - Miedo, 3 - Asco, 4 - Ira, 5 - Neutral.

3.2.2. Limpieza y preparación de los datos de actividad física

Primero se realiza la carga de los datos de los 22 pacientes almacenados en el archivo *scores.csv*, después de haber eliminado el único paciente de tipo bipolar I. Mediante la función *read_csv* se procede a la eliminación de los campos que no resultan de interés, en este caso la columna *number* y *days*, y se obtiene una tabla de datos como la que se muestra en la *Figura 18*.

```
datosPacientes = pd.read_csv('depresjon-dataset/data/scores.csv', na_values='--undefined--')
datosPacientes = datosPacientes.drop(columns=['number'])
datosPacientes = datosPacientes.drop(columns=['days'])
datosPacientes.head()
```

	gender	age	afftype	inpatient	edu	marriage	work	madrs1	madrs2
0	2	35-39	2	2	6-10	1	2	19	19
1	2	40-44	1	2	6-10	2	2	24	11
2	1	45-49	2	2	6-10	2	2	24	25
3	2	25-29	2	2	11-15	1	1	20	16
4	2	50-54	2	2	11-15	2	2	26	26

Figura 18. Lectura del archivo CSV y una muestra del contenido

En el siguiente paso se separan los datos en cuatro tablas en función de su clase (unipolar depressive o bipolar II) y su género, para posteriormente realizar un reparto

equilibrado de estos datos con un mismo número de pacientes de una clase y género concretos, y evitar así un posible sobreajuste (*Figura 19*).

```
index_type1_male = datosPacientes[(datosPacientes.afftype==1) & (datosPacientes.gender==2)]
index_type1_female = datosPacientes[(datosPacientes.afftype==1) & (datosPacientes.gender==1)]
index_type2_male = datosPacientes[(datosPacientes.afftype==2) & (datosPacientes.gender==2)]
index_type2_female = datosPacientes[(datosPacientes.afftype==2) & (datosPacientes.gender==1)]

remuve_n = 0

if(len(index_type1_male) <= len(index_type2_male)):
    remove_male = len(index_type2_male) - len(index_type1_male)
    index_type2_male = index_type2_male.sample(len(index_type1_male))
else:
    remove_male = len(index_type1_male) - len(index_type2_male)
    index_type1_male = index_type1_male.sample(len(index_type2_male))
if(len(index_type1_female) <= len(index_type2_female)):
    remove_female = len(index_type2_female) - len(index_type1_female)
    index_type2_female = index_type2_female.sample(len(index_type1_female))
else:
    remove_female = len(index_type1_female) - len(index_type2_female)
    index_type1_female = index_type1_female.sample(len(index_type2_female))

remove_n = remove_male + remove_female
```

Figura 19. Operaciones para la elección de índices en código Python

Una vez hecho esto se concatenan las cuatro tablas para obtener los índices correspondientes a los pacientes que van a ser evaluados.

El siguiente paso es cargar a partir de estos índices los archivos CSV que contiene la carpeta *condition*, que aparecen numerados por pacientes con el siguiente nombre *condition_NumPaciente.csv*. La *Figura 20* muestra un ejemplo de los datos que contienen estos archivos.

	timestamp	date	activity
0	2003-05-07 15:00:00	2003-05-07	1468
1	2003-05-07 15:01:00	2003-05-07	1006
2	2003-05-07 15:02:00	2003-05-07	468
3	2003-05-07 15:03:00	2003-05-07	306
4	2003-05-07 15:04:00	2003-05-07	143

Figura 20. Contenido archivos condition_NumPaciente.csv

Se suprime la columna *date* que no se utiliza y se selecciona el número de horas por la que se agregaran los datos, añadiendo además la media de la actividad (meanActivity), la desviación típica (stdActivity), la varianza (varActivity), el máximo (max) y el mínimo (min) por cada tramo en un día completo. El número de tramos en un día completo dependerá del número de horas por las que hemos agregado la actividad, por lo que si agregamos la actividad, por ejemplo, en periodos de 6 horas, tendremos cuatro tramos de 6 horas en un día. El procedimiento seguido para conseguir esto se muestra en la *Figura 21*.

```
for i in range(0, 22 - remove_n):
    actividadPaciente[i].index = pd.DatetimeIndex(actividadPaciente[i].timestamp)
    aux = actividadPaciente.copy()
    actividadPaciente[i] = round(aux[i].resample(resample_hour).mean(),2)
    actividadPaciente[i]['sumActivity'] = round(aux[i].resample(resample_hour).sum(),2)
    actividadPaciente[i]['stdActivity'] = round(aux[i].resample(resample_hour).std()['activity'],2)
    actividadPaciente[i]['varActivity'] = round(aux[i].resample(resample_hour).var()['activity'],2)
    actividadPaciente[i]['max'] = aux[i].resample(resample_hour).max()['activity']
    actividadPaciente[i]['min'] = aux[i].resample(resample_hour).min()['activity']
    actividadPaciente[i]=actividadPaciente[i].rename(columns = {'activity':'meanActivity'})
```

Figura 21. Obtención de las variables de actividad en código Python

Para evitar incluir en los datos de actividad de cada paciente periodos donde la monitorización había acabado, y hay registros de actividad a 0, se realiza la siguiente operación para conseguir disminuir su porcentaje en los datos:

Mientras que la suma de la actividad media del 20 % del final de los datos sea menor que la suma de la actividad media del 20 % del principio de los datos dividido por 3, se disminuyen los datos un 5 % del final.

A continuación, se eliminan las primeras horas hasta el primer registro que comience a las 00:00:00 y se repite el proceso, eliminando ahora las últimas horas hasta el primer registro anterior a las 00:00:00, para así asegurar que no hay partes de un día en los datos y son días enteros.

El siguiente paso es juntar en un mismo dataframe la tabla de los datos de los pacientes con su correspondiente tabla de datos de actividad y, seguidamente, se normalizan todas las columnas.

Para los datos categóricos se utiliza la función *fit_transform* perteneciente a la clase *LabelEncoder*, que incluye el módulo *preprocessing* importado de *sklearn*, que se encarga de codificar etiquetas con valores entre 0 y n_classes -1.

Para las columnas con valores cuantitativos se aplica la normalización *MinMaxScaler* vista anteriormente (3) de la librería *sklearn*. La *Figura 22* muestra el resultado de la tabla con los datos de entrenamiento sin la clase objetivo.

	meanActivity	sumActivity	stdActivity	varActivity	max	min	hour	gender	age	inpatient	edu	marriage	work	madrs1	madrs2
0	0.051847	0.051849	0.146833	0.021559	0.223583	0.037037	0	2	4	2	2	2	2	0.583333	0.0
1	0.451566	0.451569	0.500885	0.250881	0.551358	0.049383	1	2	4	2	2	2	2	0.583333	0.0
2	0.388030	0.388034	0.394379	0.155537	0.325413	0.049383	2	2	4	2	2	2	2	0.583333	0.0
3	0.127023	0.127023	0.303789	0.092287	0.223583	0.037037	3	2	4	2	2	2	2	0.583333	0.0
4	0.049228	0.049228	0.142920	0.020427	0.197166	0.037037	0	2	4	2	2	2	2	0.583333	0.0

Figura 22. Tabla de los datos de entrenamiento

Las filas que corresponden a un tramo de un mismo día se ponen como características en columnas con el mismo nombre, pero añadiendo un número según el tramo al que corresponda, por lo que cada fila hace referencia a un día completo de un paciente, y se almacena el resultado en el dataframe X_activity_final, mediante las siguientes operaciones que se muestran en la *Figura 23*.

```
columns = X_activity.drop(['hour'], axis=1).columns.values
X_activity_list = []
X_activity_final = X_activity[X_activity.hour==0].drop(['hour'], axis=1).reset_index(drop=True)
y_activity_final = y_activity[X_activity.hour==0].reset_index(drop=True)
X_activity_list.append(X_activity_final)
for i in range(0, n_hour_class - 1):
    X_activity_list.append(X_activity[X_activity.hour==i + 1].reset_index(drop=True))
    X_activity_list[-1] = X_activity_list[-1].drop(['hour'], axis=1)

for j in range(0, n_hour_class - 1):
    X_activity_list[j+1].columns = list(columns + str(j + 1))
    X_activity_final = pd.concat([X_activity_final, X_activity_list[j+1]], axis=1)
```

Figura 23. Operaciones para obtener el dataframe X_activity_final en código Python

Una vez hecho esto se seleccionan las características de los datos del paciente que se desean incorporar para el entrenamiento a la tabla X_activity_final. En la tabla y_activity_final se almacena la clase correspondiente al paciente de ese día concreto.

4. Análisis de los datos

4.1. Datos de audio

En primer lugar, se hace un análisis para conocer cómo se han generado las bases de datos, pues en ambas los audios han sido grabados por actores fingiendo las emociones. De esta forma se determina qué problemas relacionados con el sobreajuste o subajuste podrían presentarse debido al proceso de creación de esas bases de audios.

El sobreajuste se puede producir en la base de datos en alemán al tener solo audios de 10 personas entre 25 y 35 años, lo que dará problemas para clasificar audios de personas en otro rango de edad y de países con un idioma diferente. En la base de datos de audios españoles sucede algo dado que sólo hay 2 personas que graban todos los audios.

Respecto al subajuste, al no tener una gran cantidad de datos y la distribución de ésta no ser proporcional para cada estado de ánimo (problema de desbalanceo de clases), como refleja la *Figura 24*, puede que el modelo no sea capaz de distinguir ciertas emociones.

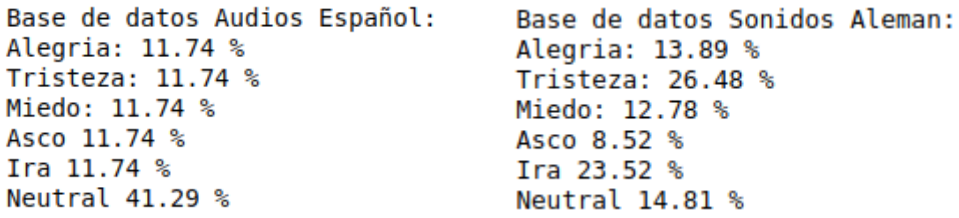

Figura 24. Porcentaje de datos de cada clase en la base de datos en español y alemán

En la *Figura 24* se observa cómo en la base de datos de audios españoles hay una clase (Neutral) que destaca notablemente sobre las demás y es la que tiene más probabilidad de clasificarse correctamente, e incluso de que el algoritmo identifique casi todos los audios como de dicha clase. Por ello será conveniente equilibrar la cantidad de ejemplos antes de entrenar al algoritmo, dado que la diferencia de ejemplos entre la clase neutral y el resto es demasiado grande.

En la base de datos de sonidos en alemán ocurre lo mismo con la clase Tristeza e Ira, que contienen alrededor del 50% de los datos totales.

Los audios constan de un total de 118 variables. Es necesario reducir este conjunto a uno menor, tanto para realizar un análisis más detallado de las mismas, como para filtrar las que son realmente relevantes a la hora de entrenar al sistema de predicción. Esto mejorará el tiempo de cómputo y eliminará redundancias o posibles irregularidades, aunque también es muy probable que el porcentaje de acierto pueda ser reducido.

Está reducción comienza con la supresión de variables que tienen un alto nivel de correlación entre sí, de forma que la cantidad de información perdida sea mínima gracias a que ésta puede estar contenida en otras variables. El nivel de correlación puede ser calculado con el método *corr* de data frame, y de forma gráfica podrá observarse el nivel de correlación gracias a un mapa de calor, que será mostrado por secciones debido al amplio tamaño del conjunto de variables. La correlación se calcula utilizando las siguientes fórmulas.

$$Correlación_{x,y} = \frac{Covarianza_{XY}}{Desviación\ típica_X * Desviación\ típica_Y} \quad (10)$$

$$Covarianza_{x,y} = \frac{\sum_1^n (X_i - \mu_X)(Y_i - \mu_Y)}{n} \quad (11)$$

$$Desviación\ típica_x = \sqrt{\frac{\sum_i^N (X_i - \mu_X)^2}{N}} \quad (12)$$

Se extraerá la correlación que existe en los distintos conjuntos de datos para después analizar qué variables tienen una alta correlación entre sí y, por lo tanto, proceder a su eliminación. Correlaciones por encima de un 0.8, en valor absoluto, serán consideradas como altas y se tendrán en cuenta a la hora de eliminar variables. La *Figura 25* muestra la función utilizada para dicho cometido.

```
correlation = X_Mezclados.corr()
limites = [0,5,10,15,21,23,28,33,38,43,48,53,58,63,68,73,78,83,88,94,100,106,112,118]
for i in range(0,len(limites)-1):
    for j in range(i,len(limites)-1):
        correlation = correlation.iloc[limites[i]:limites[i+1],limites[j]:limites[j+1]]
        if(np.any(correlation<-0.8) or np.any(correlation>0.8)):
            plt.figure()
            sns.heatmap(correlation, xticklabels = correlation.columns.values,
                        yticklabels=correlation.index.values, vmin=-1,vmax=1,annot=True)
            plt.title('Feature correlations')
            plt.tight_layout()
            plt.savefig(correlation.columns.values[0] + correlation.index.values[0])
        correlation = X_Mezclados.corr()
```

Figura 25. Proceso para mostrar las correlaciones mediante mapas de calor en código Python

A parte de conocer la relación que existe entre las variables, será útil saber qué variables tienen una mayor relevancia a la hora de predecir, de forma que éstas se mantengan en el conjunto final de variables y no sean eliminadas. Con ese objetivo se hace uso de un clasificador con estructura de árbol que determine qué variables son más importantes. El código necesario para ejecutar el clasificador se muestra en la *Figura 26*, y uno de los árboles creados se muestra en la *Figura 27*.

```
def moreImportantFeatures(how_many, classifier, inputData, outputData):
    model = classifier.fit(inputData,outputData)
    featureImportance = model.feature_importances_
    orderImportance = np.argsort(featureImportance)
    if(how_many >=0):
        for i in range(1,how_many):
            print(inputData.columns.values[orderImportance[-i]])
    else:
        for i in range(1,-how_many):
            print(inputData.columns.values[orderImportance[i]])
```

```
moreImportantFeatures(-20, ExtraTreesClassifier(n_estimators=90),X_MezcladosNor,y_Mezclados)
```

Figura 26. Implementación de la función moreImportantFeatures

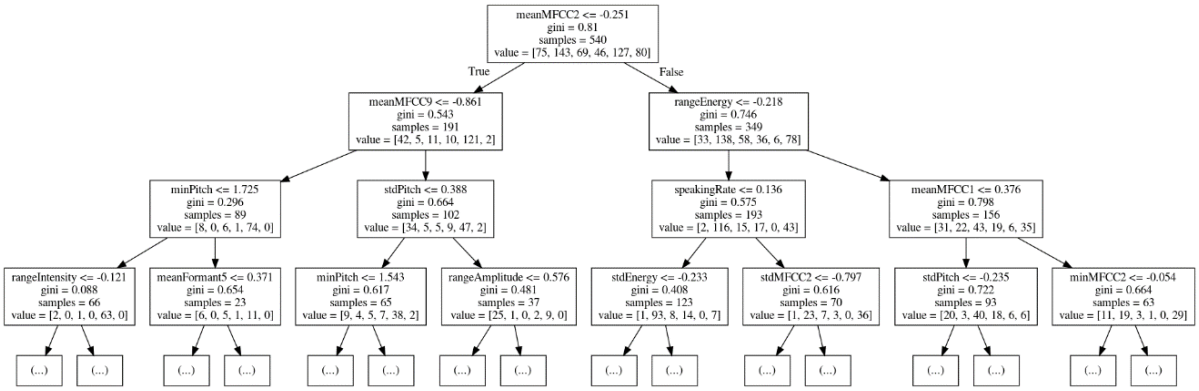

Figura 27. Arbol de decisión generado por la función de la Figura 26

El análisis anterior ha sido ejecutado varias veces dando como variables más importantes: meanMFCC2, minMFCC2, stdMFCC2, meanPitch, meanHarmonicity, meanMFCC1, maxAmplitude, rangeAmplitude, minAmplitude, meanMFCC9, speakingRate, stdHarmonicity, minMFCC9. Por ello al eliminar las variables correlacionadas se procura mantener éstas siempre que sea posible.

Algunos de los mapas de calor de interés son los mostrados en las *Figuras 28, 29 y 30.* En los ejes de los mapas estarán dispuestas las variables a comparar, de forma que en la posición (x,y), se encuentre la correlación entre la variable x y la variable y. La correlación posee valores en el intervalo [-1,1]. Cuando ésta es cercana a 0 se dice que no existe correlación entre las variables (colores rojizos en los mapas), y cuando la correlación es cercana a los extremos es debido a que existe correlación entre las variables; cuando es negativa (colores oscuros) se debe a una relación inversa entre las variables, mientras que cuando es positiva (colores claros) se debe a una relación directa entre las variables.

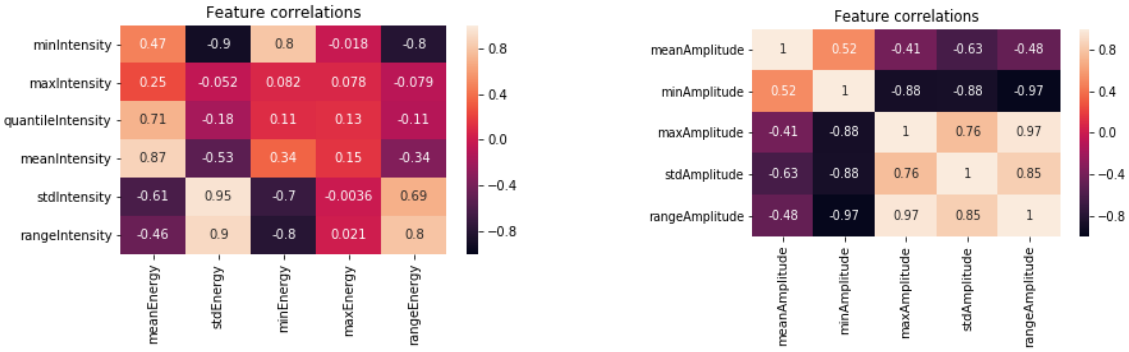

Figura 28. Mapas de calor de las correlaciones entre variables I

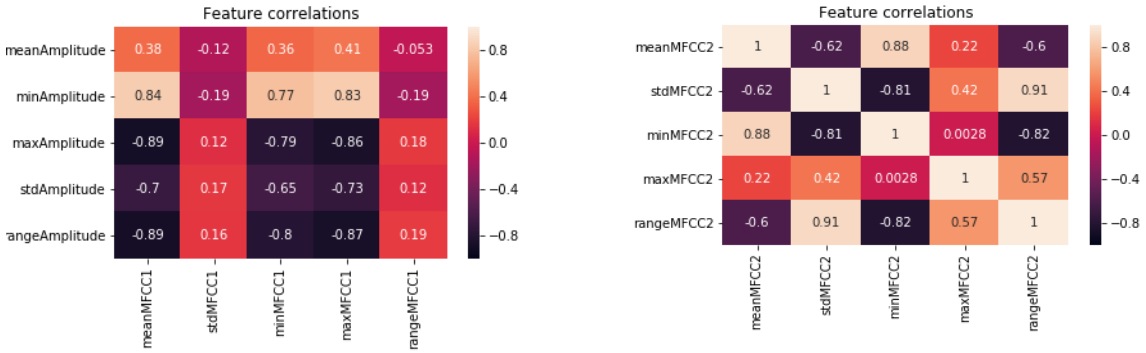

Figura 29. Mapas de calor de las correlaciones entre variables II

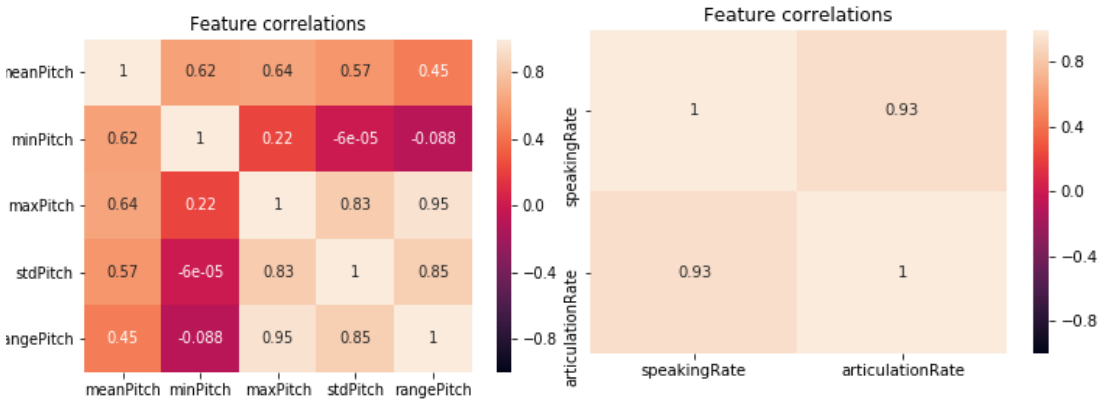

Figura 30. Mapas de calor de las correlaciónes entre variables III

Tras analizar varias gráficas del mismo estilo, se han determinado las variables que poseen una mayor correlación y entre ellas se selecciona un subconjunto de 23 variables que se han eliminado: *meanEnergy*, *minEnergy*, *rangeEnergy*, *stdEnergy*, *rangeAmplitude*, *minMFCC1*, *stdMFCC1*, *rangeMFCC2*, *rangeMFCC4*, *rangeMFCC12*, *maxPitch*, *stdPitch*, *quantileFormant1*, *quantileFormant2*, *rangeFormant2*, *quantileFormant3*, *quantileFormant4*, *rangeFormant4*, *rangeFormant5*, *quantileFormant5*, *stdAmplitude*, *articulationRate*, *meanIntensity*.

Está eliminación de variables resulta en otro conjunto de datos de entrenamiento que, junto a los anteriores, sirven para valorar la importancia de las variables eliminadas; lo que se discute en la parte de resultados. A partir de este punto, en el trabajo se hará referencia a estos datos como conjunto sin variables correlacionadas.

Otro tipo de variables que podrían ser eliminadas son aquellas que tienen muy poca variabilidad en los ejemplos de entrenamiento, ya que no aportan diferencias significativas para extraer algún patrón de ellas. Sin embargo, se deberían establecer unos valores límites para la varianza y la mejor forma de hacerlo sería establecer un porcentaje de igualdad de los datos y calcular la varianza con ese porcentaje y la fórmula adecuada para cada variable. Esta fórmula se obtiene conociendo la distribución que sigue la variable. Dado que hay distintas variables con distintas distribuciones, este método no es el más adecuado para aplicarlo de forma genérica.

Sí es posible eliminar ese tipo de variables utilizando el árbol de decisión que se hizo para determinar qué variables eran las más importantes. Para ello bastará con fijarse en las variables que se consideran menos relevantes y eliminarlas. Esto permite eliminar las variables con poca variabilidad en los ejemplos de entrenamiento ya que el algoritmo las utilizará en los niveles más bajos pues no reducen mucho la entropía de los nodos de éste.

Precisamente al observar las 20 variables menos relevantes, el algoritmo considera que la variable menos relevante es *minFormant1*, la cual, tiene muy poca variabilidad debido a que la mayoría de los valores eran valores NAN y fueron sustituidos por la media.

Tras realizar varios análisis, las variables consideradas menos relevantes por el algoritmo de clasificación son las siguientes: *minFormant1, maxFormant3, meanHarmonicity, maxFormant5, rangeFormant2, rangeFormant1, maxMFCC6, maxFormant2, minFormant2, maxMFCC4, maxFormant4*.

Con esto se obtienen dos nuevos conjuntos; uno de ellos eliminando estas variables menos relevantes, y otro eliminándolas del conjunto sin variables correlacionadas.

4.2. Datos de actividad física

El archivo *scores.csv* contiene 22 pacientes repartidos de la siguiente manera (*Figura 31*):

```
Pacientes género masculino: 12
Pacientes género femenino: 10
Pacientes con depresión: 15
Pacientes bipolares: 7
Pacientes género masculino y depresión: 8
Pacientes género femenino y depresión: 7
Pacientes género masculino y bipolares: 4
Pacientes género femenino y bipolares: 3
```

Figura 31. Número de pacientes en función del tipo y género original

Tras realizar la distribución vista en la obtención de los datos de actividad se cuenta con un total de 14 pacientes para continuar con el proceso, repartidos como aparecen a continuación (*Figura 32*):

```
Pacientes género masculino: 8
Pacientes género femenino: 6
Pacientes con depresión: 7
Pacientes bipolares: 7
Pacientes género masculino y depresión: 4
Pacientes género femenino y depresión: 3
Pacientes género masculino y bipolares: 4
Pacientes género femenino y bipolares: 3
```

Figura 32. Número de pacientes en función del tipo y género tras procesar los datos

Para analizar visualmente las posibles diferencias en la actividad de cada clase de paciente creamos un gráfico de la media total de actividad de cada tipo por horas. La *Figura 33* muestra el resultado obtenido.

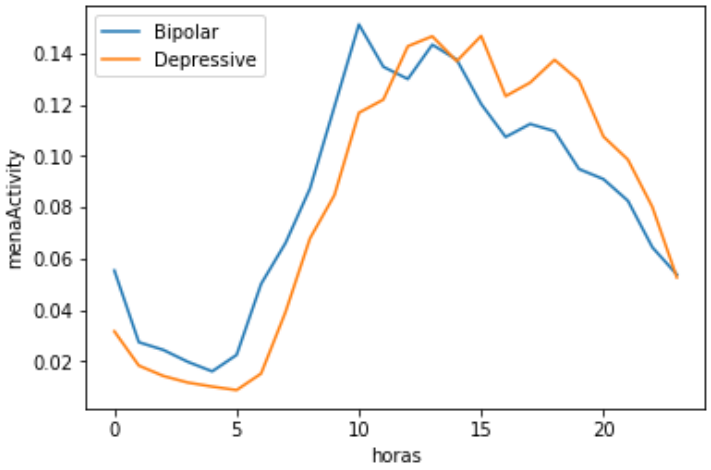

Figura 33. Gráfico de la media de actividad según el tipo de paciente por horas

Como se puede observar en la *Figura 32*, hay mayor actividad de los pacientes bipolares en horario de madrugada, hasta aproximadamente las 11:00 de la mañana, a partir de la cual cambia la tendencia y se produce una mayor actividad de los pacientes con depresión. Los picos de mayor actividad en pacientes bipolares respecto a los pacientes con depresión se producen alrededor de las 6:00-9:00 de la mañana y en pacientes con depresión alrededor de las 14:00-19:00 de la tarde.

Tras realizar este análisis general, se observa el comportamiento individual de pacientes de cada clase. A continuación se crean dos gráficos que muestran un comparativo de la media de la actividad agregada por cada hora en un día completo de cada tipo de paciente para ver qué diferencias pueden observarse y si se cumple lo anteriormente analizado (*Figuras 34 y 35*).

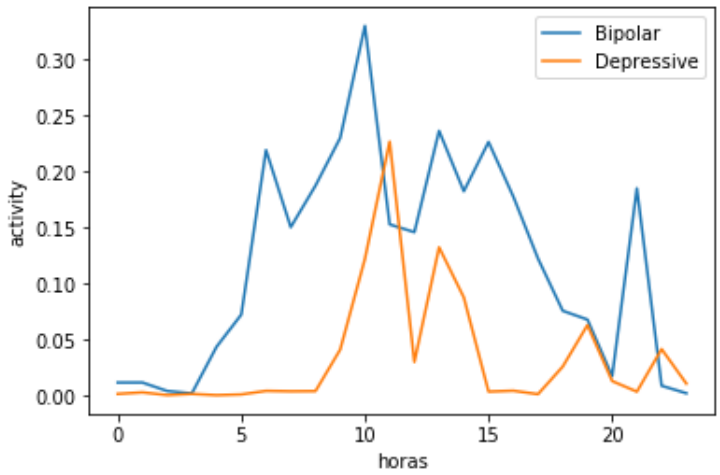

Figura 34. Gráfica de la media de actividad de pacientes con depresión y bipolares por horas I

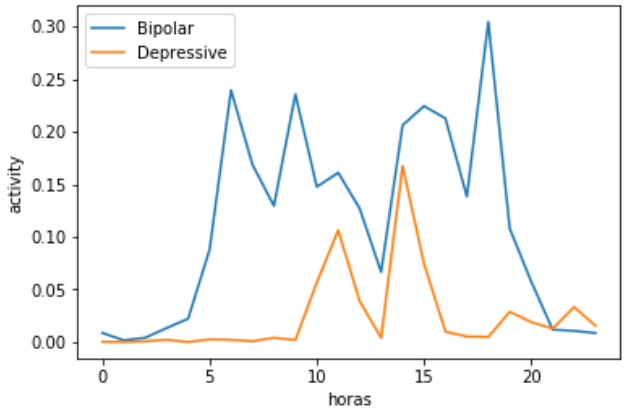

Figura 35. Gráfica de la media de actividad de pacientes con depresión y bipolares por horas II

Como se puede observar en la *Figura 34*, las primeras horas de la madrugada la actividad es más reducida que en la de los pacientes bipolares, y la tendencia cambia a las horas finales del día, donde son los pacientes con depresión los que tienen registros de actividad mayores.

A la vista de estos resultados, se cree conveniente que la actividad sea agregada en cuatro tramos de 6 horas (00:00-06:00, 06:00-12:00, 12:00-18:00, 18:00-00:00), ya que en la actividad de cada tipo de paciente se pueden observar cambios de mayor contraste en periodos grandes de tiempo. Aun así, se procede a realizar el entrenamiento con los datos agregados cada 6 y cada 2 horas para ver qué modelos ofrecen mejor comportamiento según el tipo de algoritmo utilizado.

El número total de días medidos va a depender de los pacientes incorporados en la muestra; al ser un proceso aleatorio no puede determinarse con exactitud. El valor está en torno a los 200 días.

Respecto a las características de los pacientes, se decide incluir solo el género ya que las demás características no están equilibradas. Por lo tanto, al tener pocos datos podría producirse sobreajuste de las clases predominantes. La *Figura 36* muestra una comparativa de la actividad media entre hombres y mujeres por horas.

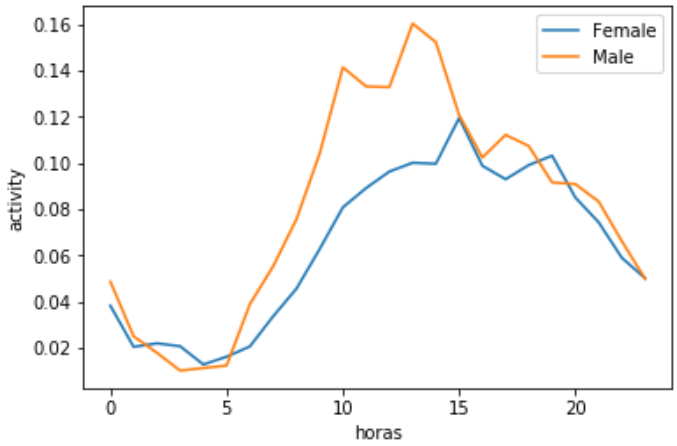

Figura 36. Gráfico de la media de actividad según el género del paciente por horas

Como se puede observar, los pacientes de género femenino tienen mayor actividad a primera hora de la madrugada, de 02:00 a 04:00, y cambia la tendencia hasta aproximadamente las 15:00, donde la actividad de los hombres es muy superior. A partir de aquí la actividad entre ambos se iguala. Se puede ver claramente como en general la actividad de los pacientes de género masculino es superior de media durante el transcurso del día por lo que este puede ser una característica diferencial para incluir en el proceso de predicción.

Las columnas que contienen el mínimo de actividad denominadas *min* son excluidas del conjunto de entrenamiento al ser siempre 0 el valor mínimo de actividad, como es lógico.

En la *Figura 37* se observa un ejemplo del resultado final del conjunto de entrenamiento con la actividad agregada cada 6 horas.

meanActivity	sumActivity	stdActivity	varActivity	max	...	stdActivity3	varActivity3	max3	gender
0.056345	0.056347	0.175595	0.030832	0.189375	...	0.363296	0.131984	0.189375	2
0.053499	0.053499	0.170915	0.029213	0.167000	...	0.397378	0.157908	0.259000	2
0.037130	0.037131	0.121434	0.014745	0.107375	...	0.245509	0.060278	0.134125	2
0.043865	0.043860	0.138403	0.019156	0.110875	...	0.165621	0.027430	0.100750	2
0.033688	0.033689	0.118415	0.014021	0.121875	...	0.143671	0.020643	0.091625	2

Figura 37. Tabla del conjunto de entrenamiento con los datos de actividad agregados cada 6 horas.

Dependiendo del tipo de agregación utilizado para la actividad, cada 6 o cada 2 horas, el número de columnas variará su tamaño de 21 en el primer caso a 61 en el segundo. Éste es otro aspecto importante que considerar ya que puede influir en el rendimiento del algoritmo.

5. Aplicación de los algoritmos

Dentro del aprendizaje automático se puede distinguir entre aprendizaje supervisado y no supervisado.

El aprendizaje supervisado requiere tener disponibles un conjunto de pares de variables de entrada que se asocian a una variable de salida, que será el conjunto de entrenamiento. Con la ayuda de un algoritmo el objetivo es predecir la salida para valores nuevos de las variables de entrada.

El aprendizaje supervisado se usa principalmente para resolver problemas de clasificación, donde la variable de salida es categórica, y problemas de regresión, donde la variable de salida es un valor real.

En el aprendizaje no supervisado sólamente se conocen las variables de entrada. Su objetivo es crear un modelo que permita obtener información de los datos basándose en su estructura o distribución interna.

El aprendizaje no supervisado se usa principalmente para resolver problemas de agrupamiento (clustering), donde se quiere descubrir agrupaciones que tienen relaciones entre sí, y problemas de asociación, donde se desea descubrir reglas que describen grandes porciones de los datos.

Se han probado distintos algoritmos de clasificación, todos ellos supervisados, sobre los conjuntos de datos de audio y actividad, con el objetivo de identificar el algoritmo más adecuado y que proporcione la mayor precisión a la hora de determinar el estado de ánimo de un paciente.

Los algoritmos usados en esta parte son: Random Forest [13], Gradient Boosting [21], Support Vector Machine [6], Redes Neuronales (MLP) [23] y Naïve Bayesian [16].

A la hora de probar el funcionamiento de los algoritmos es necesario separar los datos en dos conjuntos, uno de entrenamiento y otro de validación. De esta manera, el algoritmo será entrenado con los datos del conjunto de entrenamiento y se prueba su eficacia prediciendo el resultado de los datos del conjunto de validación, comprobando la tasa de acierto y fallo, con lo que se obtiene la tasa de éxito. Esta labor la facilita la función *train_test_split* de la librería de Scikit-learn que separa los dos conjuntos en base a un tamaño dado para el conjunto de validación.

Para que el funcionamiento del algoritmo sea mejor, se utiliza un método denominado validación cruzada (cross validation), que consiste en separar el conjunto de datos, como anteriormente, en dos conjuntos, uno de entrenamiento y otro de validación, pero esta vez seleccionamos ‘x’ conjuntos de entrenamiento y validación de un tamaño dado y se prueba cada vez con una parte distinta de los datos. Finalmente se obtiene la precisión con la media de las ‘x’ pruebas realizadas en el conjunto. La librería de Scikit-

learn también proporciona la función *cross_validate* que realiza automáticamente lo descrito y permite seleccionar el número de particiones a realizar en el conjunto de datos. A continuación, se muestra un ejemplo del funcionamiento (*Figura 38*).

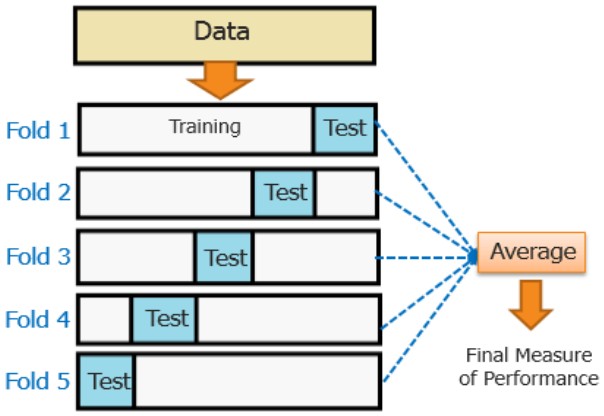

Figura 38. Funcionamiento Cross Validation

Uno de los problemas que se puede presentar a la hora de aplicar los algoritmos es el sobreajuste y el subajuste.

El sobreajuste se produce cuando los datos de entrenamiento para una clase específica son muy parecidos entre sí y no hay mucha variedad dentro de los posibles datos en los que queremos que se aplique el algoritmo, o cuando se utiliza un modelo tan complejo que "memoriza" los datos de entrenamiento.

El subajuste se puede producir cuando tenemos pocos datos de entrenamiento o los datos tienen muy poca similitud entre sí para cada clase específica. De esta manera la posibilidad de fallo es mayor, debido a que el modelo no está suficientemente entrenado. En la *Figura 39* se ejemplifican las diferentes posibilidades comentadas.

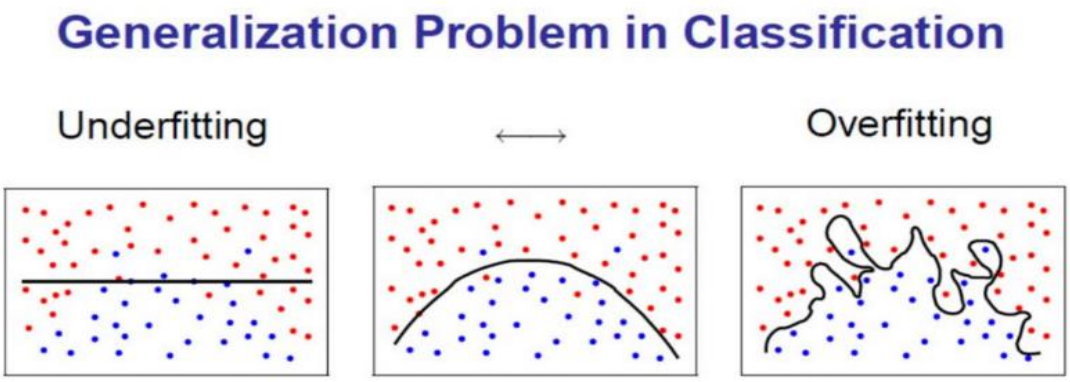

Figura 39. Problemas de clasificación

Tras este breve análisis y una vez realizados todos los procedimientos comentados previamente, comienza la fase de modelado, y para ello se utilizan los conjuntos de datos que se han obtenido tras esos procedimientos.

Para los audios se obtienen varios conjuntos preparados para el entrenamiento. En concreto, se plantean distintas posibilidades, que surgen de variar la normalización, el conjunto de datos y el conjunto de variables:

- Conjunto de datos:
    1. Datos mezclados
        1. Todos los ejemplos
        2. Sin outliers
    2. Datos alemán
        1. Todos los ejemplos
        2. Sin outliers
    3. Datos español
        1. Todos los ejemplos
        2. Sin outliers
- Normalización:
    1. MinMax
    2. Z-Score
    3. Norma l2
    4. RobustScaler
- Conjunto de variables:
    1. Todas las variables
    2. Eliminadas variables correlacionadas
    3. Eliminadas variables poco significativas
    4. Eliminadas variables correlacionadas y poco significativas

Realizando el producto cartesiano de esas posibilidades se obtienen 96 posibles conjuntos con los que realizar pruebas. En cada una de esas 96 pruebas se aplican todos los modelos comentados en líneas anteriores, de forma que también sea posible determinar qué modelo es el más adecuado para entrenar a los datos, junto con los valores de los parámetros.

Para la actividad física se realizan pruebas con los datos de actividad agregados cada dos y cada 6 horas. Para cada conjunto de datos se realizan 3 pruebas para determinar la eficacia del modelo aplicando todos los algoritmos mencionados anteriormente.

5.1. Random Forest

Un árbol de decisión es un diagrama de flujo con estructura de árbol binario, donde cada nodo interno denota una prueba para un atributo, cada rama representa un resultado de la prueba, y cada nodo hoja tiene una etiqueta de clase.

El algoritmo Random Forest es una combinación de árboles de decisión donde cada árbol depende de los valores de un vector aleatorio probado independientemente y con la misma distribución para todos los árboles en el bosque. Cada árbol de decisión realiza una predicción y una vez conocidos los resultados de todos los árboles se decide la clase resultante por votación popular, es decir, la clase que más veces ha sido predicha.

En este proyecto está implementado mediante la función *RandomForestClassifier* de la librería de Scikit-learn. En la obtención de la precisión para este algoritmo, se prueban distintos valores del parámetro *n_estimators,* que indica el número de árboles en el bosque (Cuánto mayor sea el número de árboles más fiable es el resultado final, en cambio se degrada el rendimiento).

Se muestra un ejemplo de su funcionamiento en la *Figura 40*:

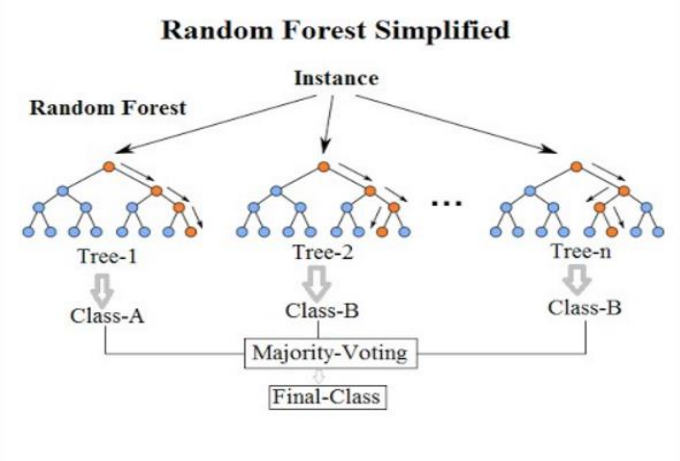

Figura 40. Funcionamiento Random Forest Algorithm

5.1.1. Datos de audio

Para analizar los resultados se analizan los resultados obtenido utilizando los diferentes conjuntos de datos:

Datos Mezclados:
Para los casos en los que se utilizaban todos los ejemplos disponibles en el conjunto de datos se consigue un 77-79% de acierto, siendo habitual que los casos en los que no se eliminan variables sean un poco mejores e impliquen algún segundo más para el entrenamiento, requiriendo éste entre 4 y 5 segundos.

Sin embargo, utilizando el conjunto de datos resultante de eliminar los outliers, el porcentaje de acierto se incrementa al 85% y el tiempo se reduce en torno a los 2 segundos. Esto es debido a que en el conjunto inicial de datos se encuentran ejemplos de diferentes idiomas, lo que provocaría una mayor confusión en la clasificación. Al eliminar los casos más extremos esta clasificación se facilita. Al eliminar el conjunto de variables que tenían una alta correlación se consigue reducir el tiempo empleado sin sacrificar el porcentaje de acierto, obteniendo un 85,52% en 1,89 segundos. Este resultado se obtuvo entrenando el algoritmo con el conjunto de datos resultante de eliminar outliers, eliminar las variables correlacionadas mencionadas en el apartado 4 y normalizando con la norma *l2,* aunque utilizar otras normalizaciones no modifica apenas el resultado. El número de estimadores para el modelo era de 31.

Datos Alemán:
Utilizando los datos sin eliminar outliers los resultados se encuentran en torno al 73% de acierto y 1 segundo de tiempo de entrenamiento. Los resultados son muy similares para todos los conjuntos de variables probados, por lo que un conjunto óptimo sería uno con

las variables correlacionadas o menos importantes eliminadas, ya que al eliminar ambas sí se produce un mayor descenso de la tasa de acierto.

Con estos datos, al eliminar outliers, el porcentaje de acierto se ve reducido al 70%, debido a que la cantidad de ejemplos disponibles ya era muy escasa. En cuanto al tiempo estimado para el entrenamiento es muy similar al tiempo que requería el uso de los datos sin eliminar outliers.

Se puede concluir que el mejor resultado obtenido ha sido de un 74,26% de aciertos en 1,3 segundos, utilizando los datos sin eliminar outliers, eliminando las variables correlacionadas y normalizando con Z-Score, aunque eliminar las variables menos importantes y normalizar de otra forma produce resultados similares. El número de estimadores para el modelo era de 61.

Datos Español:
Utilizando los datos sin eliminar outliers se observa que los datos obtenidos son muy similares para todos los conjuntos de variables y normalizaciones posibles, alcanzando siempre un porcentaje de acierto del 92% en un tiempo de entrenamiento de 1 segundo, y alcanzando el 93% en algún caso, con un tiempo de 3 segundos y haciendo uso de todas las variables. En vista a estos resultados es más razonable el primer resultado a pesar de que el acierto sea algo menor, ya que el tiempo se ve reducido.

Con los datos obtenidos tras eliminar los outliers, el porcentaje de acierto se incrementa ligeramente, alcanzando en casi todas las pruebas un acierto en torno al 93%. En este caso eliminar las variables menos importantes sí que otorga un buen resultado consiguiendo incrementar el acierto en un tiempo muy similar.

Se puede concluir que la configuración que parece producir el mejor resultado es utilizar los datos sin outliers y eliminar las variables menos importantes, lo que consigue un 94% en 2 segundos con la normalización MinMax. Es importante comentar nuevamente que el uso de otra normalización llevaría a la obtención de un resultado muy similar. Además, el modelo hacía uso de 71 estimadores.

5.1.2. Datos de actividad

Datos agregados cada 6 horas
La precisión máxima obtenida con los datos de actividad es del 76,36 % en un tiempo total de ejecución de 1,09 segundos, conseguido con un valor de 91 en el parámetro *n_estimators.*

Datos agregados cada 2 horas
La precisión máxima obtenida es del 70,18 % en un tiempo total de ejecución de 0,65 segundos, conseguido con un valor de 41 en el parámetro *n_estimators.*

5.2. Gradient Boosting

Gradient Boosting utiliza un algoritmo de aprendizaje débil, árboles de decisión en este caso, para hacer las predicciones. Los árboles de decisión se agregan de uno en uno, y los árboles existentes no cambian. Para seleccionar los parámetros de cada uno de los árboles de decisión que se agregan al modelo se utiliza el gradiente descendiente que minimiza la función de pérdida y mejora la predicción. La principal desventaja de este modelo es que requiere mucho tiempo de entrenamiento y hacer una cuidadosa selección de los parámetros.

En este proyecto está implementado mediante la función *GradientBoostingClassifier* de la librería de Scikit-learn. Los parámetros más relevantes son *n_estimators*, que indica el número de árboles incluir en el modelo, y la tasa de aprendizaje, *learning_rate*, que controla el grado en que cada árbol puede corregir los errores en los árboles anteriores.

El único parámetro que se modifica es el número de estimadores y se prueba con valores desde el 1 al 91 en intervalos de 10. Con cada valor se realiza un entrenamiento a través de la validación cruzada y finalmente se selecciona aquel valor para el cual la precisión es mayor y el tiempo no es excesivo.

5.2.1. Datos de audio

Datos Mezclados:
Los resultados obtenidos tras aplicar este algoritmo presentan una diferencia significativa en función de la presencia de outliers en el conjunto de datos. Al hacer uso de todos los ejemplos disponibles los porcentajes obtenidos son muy similares para las diferentes pruebas aplicadas, aunque el tiempo empleado para entrenar el modelo sí que presenta mayores variaciones. Mientras que el porcentaje de acierto está alrededor del 78%, los tiempos empleados se encuentran en un intervalo entre 20 y 30 segundos, siendo menores cuando el conjunto de variables es menor; es decir, eliminando las variables correlacionadas y las menos significativas se consigue la mejor relación entre el porcentaje de aciertos y el tiempo estimado.

Sin embargo, eliminando aquellos ejemplos que contienen valores atípicos, el porcentaje de acierto se incrementa hasta el 85% en los conjuntos de datos que poseen todas las variables, y como mínimo el 83% para los conjuntos en los que se eliminan algunas de ellas. En cuanto al tiempo, también presenta una mejora, pues ahora el entrenamiento se realiza en unos 15 - 20 segundos.

Por lo tanto, la configuración que proporciona mejor resultado, consiguiendo un 85% de aciertos con un tiempo de 13 segundos, es el conjunto de datos sin outliers, con todas las variables y normalizado con Z-Score; pero el uso de otra normalización no presenta grandes diferencias en los resultados, ya que solo incrementa un par de segundos el tiempo de entrenamiento. El modelo hacía uso de 21 estimadores.

Datos Alemán:
Al utilizar los conjuntos que poseen todos los ejemplos los resultados obtenidos tienen un porcentaje de acierto entre el 72% y el 75%, con un tiempo de entrenamiento entre los 5 y 10 segundos. En este caso cuantas más variables posee el conjunto de datos mayor acierto tiene, aunque con un mayor tiempo de entrenamiento. Cómo las diferencias no son muy significativas se considera que sería adecuado utilizar todas las variables para conseguir un mayor porcentaje de aciertos, a pesar de que eso incremente el tiempo empleado.

Al utilizar los datos sin outliers no se reduce el porcentaje de aciertos, como sucedía en el RFC, sino que se mantiene entre el 72% y 75%, siendo algo inferior al 72% en casos en los que se eliminan variables menos importantes y variables correlacionadas; y en cuanto al tiempo sí que hay una pequeña mejora de unos 2-3 segundos, siendo más clara cuando se utiliza el conjunto de datos que no posee las variables menos significativas.

Con esto se determina que el mejor resultado, sería de un 75,45% de aciertos en unos 7 segundos, que se obtiene con los datos sin outliers, eliminando las variables menos significativas y utilizando la normalización MinMax. El uso de otra normalización con este algoritmo no provoca grandes variaciones en el resultado. El número de estimadores utilizados por el modelo es de 41.

Datos Español:
Utilizando todos los ejemplos disponibles se consigue un porcentaje de acierto en torno al 88%-90%, con un tiempo empleado de unos 10 segundos. Los mejores porcentajes se obtienen con el conjunto de variables al completo, o bien, eliminando sólo las variables menos significativas, aunque esta vez el tiempo empleado al utilizar distintos conjuntos de variables es muy similar, mejorando escasamente cuando se utiliza el conjunto que no incluye las variables menos relevantes.

Si, por otro lado, se utilizan los ejemplos restantes al eliminar los outliers, los resultados obtenidos son bastante similares, aunque sí es cierto que en algunos casos el porcentaje de aciertos se ve ligeramente incrementado y en la mayoría de las pruebas el tiempo empleado es algo menor. Nuevamente los mejores resultados se obtienen, o bien con el conjunto de variables al completo, o bien eliminando aquellas que se consideran menos significativas.

Como mejor resultado se ha obtenido un 91,56% de aciertos en 7,34 segundos con los datos sin outliers, y eliminando el conjunto de variables menos significativas. Además, el modelo utilizaba 21 estimadores.

5.2.2. Datos de actividad

Datos agregados cada 6 horas

La precisión máxima obtenida con los datos de actividad es del 73,90 % en un tiempo total de ejecución de 0,74 segundos, conseguido con un valor de 91 en el parámetro *n_estimators.*

Datos agregados cada 2 horas

La precisión máxima obtenida es del 69,72 % en un tiempo total de ejecución de 0,89 segundos, conseguido con un valor de 71 en el parámetro *n_estimators.*

5.3. Support Vector Machine (SVM)

Las SVM transforman los datos a un espacio de dimensión alta a través de una función que se utiliza como kernel. Tras esto el algoritmo encuentra el hiperplano que separa las clases minimizando el error (en caso de existir más de dos clases, como es este caso, se realizan varias divisiones, pero siempre con SVM binarios). Hay que configurar el kernel, que será la función utilizada para separar los datos, y los parámetros asociados a éste. En este caso se utiliza un kernel RBF (Radial Basis Function) por lo que hay que ajustar gamma y el coste. El coste indica el peso que se le asigna a cada observación, es decir, a mayor peso más restrictivo será el SVM; y el gamma actúa de forma que al aumentarlo crecerá el sobreajuste.

Este kernel utilizado está definido de la siguiente manera:

$$K(x, x') = exp(\frac{||\, x - x' \,||^2}{2\sigma^2}) \qquad (13)$$

Para determinar el valor más adecuado de los parámetros se hicieron varias pruebas con distintos C y σ, y se seleccionó los que daban un mejor porcentaje de aciertos.

5.3.1. Datos de audio

Datos Mezclados:

A diferencia de lo que sucedía con los algoritmos anteriores, esta vez el resultado es más dependiente de la normalización utilizada. La normalización MinMax y Z-Score se comportan de forma muy similar; sin embargo, al utilizar la norma l2 o el Robust Scaler los resultados obtenidos empeoran.

Utilizando los datos con todos los ejemplos disponibles, las normalizaciones MinMax y Z-Score arrojan un porcentaje de acierto que está alrededor del 85%, con un tiempo de unos 2-3 segundos; con las otras normalizaciones se obtiene entre un 75% y un 80% de aciertos, en tiempos similares, aunque en ocasiones algo mayores. En cuanto al conjunto de variables más adecuado, todos presentan unos resultados muy parecidos, por lo que lo óptimo sería eliminar las variables menos importantes y las variables correlacionadas, ya que el tiempo de entrenamiento es algo menor.

Utilizando los datos sin outliers los resultados mejoran para todas las normalizaciones posibles, y además las diferencias entre ellas se ven reducidas; con MinMax y Z-Score el porcentaje de acierto está en torno al 90%, mientras que con la norma l2 y el Robust Scaler se ve reducido al 86%-88%. El tiempo de entrenamiento está entre 1 y 2 segundos para todos los casos y los conjuntos de variables se comportan de la misma forma que con los datos sin eliminar outliers, por lo que de nuevo sería más conveniente eliminar las variables menos significativas y las correlacionadas para reducir lo máximo posible el tiempo de computación.

La configuración que proporciona mejores resultados es utilizar los datos sin outliers, con el conjunto de variables sin las menos significativas y las correlacionadas, y utilizando la normalización MinMax (o Z-Score), con lo que se obtiene un 90,14% de aciertos en 1,67 segundos. Los valores para los parámetros del modelo son 1000 para C y 0,1 para σ.

Datos Alemán:
Nuevamente las normalizaciones presentan diferencias significativas en los resultados, aunque esta vez la norma l2 se comparta algo mejor, pero el Robust Scaler sigue dando los peores porcentajes de aciertos.

Utilizando los datos sin eliminar outliers, los porcentajes de acierto se encuentran entre el 81% y 83%, con tiempo muy pequeños, menores al medio segundo; estos porcentajes son menores al eliminar las variables correlacionadas, o cuando se utiliza la norma l2. Los mejores resultados se obtienen con el conjunto que no contiene las variables menos significativas.

Utilizando los datos sin outliers los porcentajes de acierto se reducen mínimamente, quedando en torno al 78%- 80%, aunque cuando se utiliza el conjunto de variables que no contiene a las variables menos significativas este porcentaje alcanza el 83%, al igual que sucedía al utilizar todos los ejemplos. Los tiempos, sin embargo, siguen siendo muy reducidos y apenas se aprecian diferencias.

Para resumir, en este caso parece que lo más importante es seleccionar el conjunto de variables que no contiene a las menos significativas, además de aplicar la normalización MinMax o Z-Score. La elección del conjunto de datos con o sin outliers no modifica mucho el mejor resultado obtenido, que es un 83% de acierto en 0,3 segundos. El modelo utiliza un valor 100 para C y 0,01 para σ.

Datos Español:
Esta vez la normalización Robust Scaler es la que presenta unos resultados peores, y aunque el resto se comporte de forma similar, la normalización MinMax y Z-Score siguen proporcionando los mejores resultados.

Utilizando los datos sin eliminar outliers se consigue un porcentaje de acierto en torno al 95%, con tiempos de menos de 1 segundo, aunque esta tasa de exito se reduce al

85% cuando utilizamos el Robust Scaler. El conjunto de variables que mejor rendimiento da es el que no contiene ni las variables correlacionadas ni las menos significativas.

Utilizando los datos sin outliers se incrementa mínimamente el porcentaje de acierto al 96%, y el tiempo se reduce en muchos casos por debajo del medio segundo. Los distintos conjuntos de variables no presentan muchas diferencias en los resultados, aunque si se eliminan las variables menos significativas y las correlacionadas, el tiempo empleado se ve reducido mínimamente y para volúmenes de datos mayores esa diferencia puede verse incrementada.

Por lo tanto, lo óptimo sería utilizar los datos sin outliers, eliminando las variables menos significativas y las correlacionadas, haciendo uso de la normalización MinMax o Z-Score; con todo eso se consigue un porcentaje de aciertos del 96,44% en 0,33 segundos. Los valores para los parámetros son 100 para C y 0,1 para σ.

5.3.2. Datos de actividad

Datos agregados cada 6 horas

La precisión máxima obtenida con los datos de actividad es del 75,32 % en un tiempo total de ejecución de 0,05 segundos, conseguido con unos valores en los parámetros *C* y σ de 1000 y 0.01, respectivamente.

Datos agregados cada 2 horas

La precisión máxima obtenida es del 80,36 % en un tiempo total de ejecución de 0,16 segundos, conseguido con unos valores en los parámetros *C* y σ de 1000 y 0.1 respectivamente.

5.4. Redes neuronales

Una red neuronal de tipo feedforward (MLP) es un conjunto de unidades de procesamiento (neuronas) organizadas en capas, de entrada, salida u ocultas. La capa de entrada representa las variables de entrada y la capa de salida el resultado. Cada unidad de procesamiento tiene una serie de entradas que se corresponden a cada una de las neuronas de la capa anterior con las que está conectada; cada una de esas entradas posee un peso por el que se multiplica la entrada.

Para el aprendizaje la red utiliza varios ejemplos y calcula los errores comparando la salida obtenida con la salida correcta para dicho ejemplo. Tras esto ejecuta un algoritmo que actualiza los pesos de entrada de cada unidad de procesamiento. Así los pesos se ajustan hasta que el error se estabiliza y no puede reducirse (algoritmo de aprendizaje de retropropagación).

Para encontrar la estructura adecuada de la red se realizan varias pruebas hasta que los resultados obtenidos son óptimos. Para determinar el número de capas se realizaron pruebas con 1, 2 y 3 capas ocultas, y los resultados obtenidos fueron muy

similares en cuanto a la precisión del modelo, por lo que se decidió probar con una red de una sola capa oculta.

Tras esto los parámetros que se modifican, son el número de neuronas en la capa oculta y el valor de regularización. Para la tasa de aprendizaje se utiliza una velocidad adaptativa que toma un valor aleatorio como inicial; el solver utilizado es "lbfgs", ya que de acuerdo a la documentación encontrada en la página de Scikit-learn, éste es el más adecuado cuando la cantidad de ejemplos es reducida.

La modificación de los pesos $W_{i,j,k}$ de la entrada j en la iteración k tras el ejemplo i se realiza mediante la expresión:

$$W_{i,j,k} = W_{i,j,k-1} + \alpha * e_{i,k} * x_{i,j}$$
$$W_{i,0,k} = W_{i,0,k-1} + \alpha * e_{i,k} \quad (14)$$

Donde el error para el ejemplo i en la iteración k, con una predicción $\hat{y}_{i,k}$:

$$e_{i,k} = y_i - \hat{y}_{i,k} \quad (15)$$

Siendo la tasa de aprendizaje α un valor del intervalo [0, 1] y, $x_{i,j}$ e y_i los datos de entrada y la salida del ejemplo i.

5.4.1. Datos de audio

Datos Mezclados:

Al igual que sucedía con las SVM, la normalización aplicada genera diferencias significativas en los resultados, siendo mejores con la normalización MinMax y Z-Score.

Sin eliminar outliers el porcentaje de acierto obtenido se encuentra entre el 81% y el 84%, con un tiempo de entrenamiento entre 2 y 3 segundos. Esto sucede con las normalizaciones que dan buenos resultados; con las otras dos el porcentaje de aciertos se reduce, consiguiendo a lo sumo un 76%. Y en cuanto al conjunto de variables óptimo, al eliminar las variables menos significativas y las correlacionadas se obtiene un porcentaje de acierto muy alto en un tiempo muy similar al resto, por lo que sería el conjunto más adecuado.

Con los datos sin outliers el porcentaje de aciertos se ve mejorado significativamente, alcanzando un 90% con las mejores normalizaciones, y en torno a un 85% en el resto. Los tiempos obtenidos son muy similares, de nuevo en torno a los 2-3 segundos, y el conjunto de variables que mejores resultados proporciona vuelve a ser el que no contiene las variables menos significativas ni las correlacionadas.

Es decir, el mejor caso da un porcentaje de aciertos del 90,28% en un tiempo de 2 segundos, con los datos sin outliers, con las variables menos significativas y correlacionadas eliminadas, y normalizados con Z-Score (aunque al normalizar con MinMax se obtiene un resultado muy similar). Los valores para los parámetros del modelo son de 60 neuronas para la capa oculta, y 0,1 para α.

Datos Alemán:
En este caso la única normalización que parece presentar peores resultados es la que se realiza mediante el Robust Scaler, y aunque con el resto son muy similares, la normalización Z-Score es la que mejores porcentajes proporciona.

Utilizando todos los ejemplos, los resultados obtenidos son unos porcentajes de acierto entre el 80% y el 84%, con tiempos de 1-2 segundos en todos los casos. El porcentaje de acierto es mayor cuando se utiliza la normalización adecuada (Z-Score) y además cuando el conjunto de variables menos significativas ha sido eliminado. El uso de cualquier otra normalización, sin tener en cuenta el Robust Scaler, reduce el porcentaje de acierto como mucho al 78%, empleando también alrededor de 1 segundo.

Si se utilizan sólo aquellos ejemplos que no contienen outliers, los porcentajes empeoran un poco, siendo un 78%-82%, aunque los tiempos se mantienen en torno a los 1-2 segundos. Al igual que sucedía al utilizar todos los ejemplos, el conjunto que mejor se comporta es el que no posee las variables menos significativas. Por lo tanto, lo más idóneo sería utilizar este conjunto con la normalización Z-Score, ya que al utilizar otras los porcentajes de acierto se reducen al 76%, de forma similar a lo que sucede al utilizar todos los ejemplos disponibles.

En resumen, el mejor resultado que da un 84,85% de acierto en 1,5 segundos, se consigue utilizando todos los ejemplos, la normalización Z-Score y el conjunto de variables que no contiene a las menos significativas. Los valores para los parámetros del modelo son de 20 neuronas para la capa oculta y 1 para α.

Datos Español:
En este caso sólamente al utilizar el Robust Scaler los porcentajes de acierto se ven reducidos, y al igual que con los datos mezclados y datos alemán, la normalización Z-Score funciona algo mejor que el resto.

Utilizando los datos que contienen todos los ejemplos, los porcentajes de acierto se encuentran en torno al 92%-95%, con tiempos de entrenamiento entre 1 y 2 segundos. En cuanto a los conjuntos de variables más adecuados, comentar que con cualquiera de ellos los resultados obtenidos se acercan al 94%-95% de acierto. Sin embargo, si se eliminan las variables menos significativas y las correlacionadas, el tiempo se reduce algo y se consigue un 95% de aciertos.

Con los datos sin outliers, la normalización aplicada vuelve a ser indiferente pues los resultados son muy similares. En este caso el porcentaje de acierto se incrementa hasta el 96%, con tiempo de entre 1-2 segundos, existiendo casos en los que baja del segundo. En cuanto al conjunto de variables no presentan diferencias muy grandes, por lo que se opta por eliminar tanto las variables menos significativas como las variables correlacionadas, de forma que se reduzca el tiempo de entrenamiento.

En resumen, el mejor resultado obtenido es de 96,56% en 0,43 segundos, conseguido con los datos sin outliers, eliminando variables menos significativas y

correlacionadas, y con la normalización Z-Score. Los valores para los parámetros del modelo son de 20 neuronas para la capa oculta y 0,1 para α.

5.4.2. Datos de actividad

Datos agregados cada 6 horas

La precisión máxima obtenida con los datos de actividad es del 76,82 % en un tiempo total de ejecución de 0,70 segundos, conseguido con unos valores en los parámetros *hidden_layer_sizes, solver* y *alpha* de 20, *'lbfgs'* y 0.1 respectivamente.

Datos agregados cada 2 horas

La precisión máxima obtenida es del 74,95 % en un tiempo total de ejecución de 1,96 segundos, conseguido con unos valores en los parámetros *hidden_layer_sizes, solver* y *alpha* de 80, *'lbfgs'* y 0.01 respectivamente.

5.5. Naïve Bayes

Es un método de aprendizaje supervisado que se basa en aplicar el teorema de Bayes. Como particularidad de este predictor es que asume la independencia condicional entre cada par de variables dado el valor de la predicción, de ahí el término "Naïve" (ingenuo). Para este proyecto se ha utilizado Gaussian Naïve Bayes, definido como:

$$P(X_i|\,Y) \;=\; \frac{1}{\sqrt{2\pi{\sigma_y}^2}}\; exp(-\frac{(X_i - \;\mu_y)^2}{2{\sigma_y}^2}) \qquad (16)$$

Donde X_i es el valor de la variable i, $\mu_y\; y\; \sigma_y$ son la media y desviación de la variable X asociada a la clase Y.

El teorema de Bayes:

$$P(Y|X_1,\ldots,X_n) \;=\; \frac{P(Y)\;P(X_1,\ldots,X_n|Y)}{P(X_1,\ldots X_n)} \qquad (17)$$

Donde $P(Y|X_1,\ldots,X_n)$ es la probabilidad de que Y sea cierto sabiendo que son ciertos $X_1,\ldots,X_n$, $P(Y)$ la probabilidad de Y, $P(X_1,\ldots X_n)$ la probabilidad de $X_1\; y\; X_2\; \ldots y\; X_n$. Y es la clase en la que se intenta clasificar un ejemplo y X_i el valor que posee la variable i.

5.5.1. Datos de audio

Con los datos de audio la probabilidad de cada clase es la misma, es decir, 0.16666, pero se fija para 4 clases a 0.17 y para las otras dos a 0.16, de forma que la probabilidad de todas sea muy similar.

En este caso la normalización sí que es completamente indiferente ya que los porcentajes conseguidos con todas las pruebas son idénticos.

Datos Mezclados:

Utilizando los datos con todos los ejemplos los resultados son muy pobres en cuanto a tasa de acierto, alrededor del 45%, con tiempos por debajo de las décimas de segundo. El uso de un conjunto de variables más reducido beneficia al resultado, siendo mejor cuando se eliminan tanto las variables menos significativas como las variables correlacionadas, llegando al 45,73% de acierto.

Eliminando los outliers, el porcentaje mejora llegando al 74%, nuevamente con tiempos muy bajos. Esta vez, a diferencia de lo que sucedía con los datos sin eliminar outliers, cuando se eliminan las variables menos significativas el resultado empeora un 4%, mientras que eliminar sólo las variables correlacionadas no modifica el porcentaje de acierto.

Con todo esto se puede decir que el mejor resultado sería de un 74,4% de aciertos, conseguido con los datos sin outliers, y eliminando las variables correlacionadas.

Datos Alemán:
Utilizando los datos sin eliminar outliers los resultados nuevamente no son muy buenos, pero sí que mejoran un poco a los porcentajes obtenidos utilizando los datos mezclados. En este caso el acierto ronda el 60-63%, siendo mejor cuando se eliminan las variables menos significativas, y empeorando cuando se eliminan las correlacionadas. En cuanto al tiempo de entrenamiento, nuevamente es ínfimo y siempre se encuentra por debajo de las décimas de segundo.

Con los datos sin outliers hay un ligero incremento del porcentaje de acierto, alcanzando un 65%. El comportamiento con los diferentes conjuntos de variables es idéntico a lo que sucede al utilizar todos los ejemplos, y por ello lo más conveniente sería eliminar las variables menos significativas.

En resumen, el mejor resultado se consigue eliminando los outliers y las variables menos significativas, para alcanzar así un 65,38% de acierto.

Datos Español:
Utilizando los datos con todos los ejemplos el porcentaje de acierto es bastante mejor que con los datos mezclados y en alemán, llegando al 83%. Este resultado se consigue eliminando las variables correlacionadas, ya que al eliminar las menos significativas el porcentaje se reduce al 81%. El tiempo de entrenamiento se encuentra por debajo de las décimas de segundo en todos los casos.

Utilizando los datos tras eliminar los outliers, los resultados continúan mejorando, alcanzando un 89% de acierto. Para conseguir dicho porcentaje es necesario, al igual que sucede al utilizar todos los ejemplos, eliminar las variables correlacionadas ya que al eliminar las variables menos significativas el porcentaje de acierto se reduce al 85%.

Por lo tanto el mejor resultado es de un 89,86% de acierto, el cuál es obtenido al eliminar los outliers y las variables correlacionadas.

5.5.2. Datos de actividad

Datos agregados cada 6 horas

La precisión obtenida con los datos de actividad es del 67,72 %, siendo claramente la peor tasa de acierto conseguida.

Datos agregados cada 2 horas

La precisión obtenida es del 61,63 %.

6. Resultados

6.1. Datos de audio

Analizados todos los algoritmos y los resultados, se procede a compararlos entre sí, tratando de justificar los valores obtenidos.

Acierto y tiempo	**RFC**	**GBC**	**SVM**	**MLP**	**NB**
Mezclados	85,52% 1,89 segundos	85% 13 segundos	90,14% 1,67 segundos	90,28% 2 segundos	74,40% <0,1 segundos
Alemán	74,26% 1,3 segundos	75,45% 7 segundos	83% 0,3 segundos	84,85% 1,5 segundos	65,38% <0,1 segundos
Español	94% 2 segundos	91,56% 7,34 segundos	96,44% 0,33 segundos	96,56% 0,43 segundos	89,86% <0,1 segundos

Figura 41. Tabla de porcentajes de acierto y tiempos de ejecución de cada algoritmo con los datos de audio.

En los resultados de la *Figura 41* se aprecia que el conjunto de datos que mejor porcentajes de éxito obtiene es el que contiene la información de los audios en español, seguido del que contiene la información de los audios en alemán y español, y en último lugar los datos de los audios alemanes. Si se tiene en cuenta los tamaños y heterogeneidad de los datos, estos resultados son sencillos de explicar ya que la cantidad de audios en español es significativamente mayor a la de los audios alemanes.Por ello este modelo es capaz de aprender mejor los patrones que puedan encontrarse en los datos. Además, los audios han sido grabados únicamente por dos personas, por lo que las diferencias de voz entre distintas personas son menos apreciables. En los datos mezclados, a pesar de tener una mayor cantidad de ejemplos, existen audios de diversas personas lo que provoca que existan diferencias en las variables tanto por los idiomas como por las características de la voz; además los ejemplos en español y alemán se encuentran desbalanceados. En cuanto a los datos en alemán, aparte de tener una cantidad de ejemplos muy escasa, existe mucha variabilidad en ellos, dado que los audios han sido grabados por 10 personas; estos dos factores provocan, por lo tanto, ese decremento del porcentaje de acierto.

Los distintos algoritmos utilizados presentan diferencias en el porcentaje de acierto y en el tiempo empleado para el entrenamiento, proporcionando mejores resultados SVM y MLP, con porcentajes muy similares, aunque con tiempos algo inferiores por parte de las SVM, lo que convierte a estos modelos en los más idóneos para la tarea de clasificar audios por estado de ánimo. Seguidamente se encuentran el RFC y el GBC, cuyos porcentajes de acierto son algo más reducidos pero similares entre sí. Sin embargo en el tiempo de entrenamiento sí que se encuentran grandes diferencias entre estos algoritmos, pues el RFC requiere entre 1-2 segundos mientras que el GBC es el más

lento de todos, empleando como mínimo unos 7 segundos. Por último el algoritmo que peores porcentajes de acierto consigue sería Naïve Bayes, a pesar de ser el más rápido.

Normalización	**RFC**	**GBC**	**SVM**	**MLP**	**NB**
Mezclados	Norma l2	Z-Score	MinMax	Z-Score	Indiferente
Alemán	Z-Score	MinMax	MinMax	Z-Score	Indiferente
Español	MinMax	Norma l2	MinMax	Z-Score	Indiferente

Figura 42. Tabla con las normalizaciones utilizadas por algoritmo en los datos de audio

De las cuatro normalizaciones probadas, se puede observar en la *Figura 42* que las mejores son Z-Score y MinMax. Por lo tanto, lo más adecuado es normalizar con MinMax cuando se utilice SVM y con Z-Score cuando se utilice el MLP

Conjunto de variables	**RFC**	**GBC**	**SVM**	**MLP**	**NB**
Mezclados	Sin variables correlacionadas	Todas las variables	Sin variables correlacionadas ni poco significativas	Sin variables correlacionadas ni poco significativas	Sin variables correlacionadas
Alemán	Sin variables correlacionadas	Sin variables poco significativas	Sin variables poco significativas	Sin variables poco significativas	Sin variables poco significativas
Español	Sin variables poco significativas	Sin variables poco significativas	Sin variables correlacionadas ni poco significativas	Sin variables correlacionadas ni poco significativas	Sin variables correlacionadas

Figura 43. Tabla del conjunto de variables óptimo de cada algoritmo en los datos de audio

Por último, en cuanto al conjunto de variables óptimo, se observa en la *Figura 43* que los mejores son aquellos en los que se suprimen las variables menos significativas, y los que, además de éstas, eliminan las variables correlacionadas. Se detecta que en los casos en los que no se eliminan variables no significativas ni correlacionadas, son en los audios alemanes. Esto puede deberse al hecho de poseer una menor cantidad de datos, ya que la redundancia presente en variables correlacionadas ayudaba al modelo a predecir con mayor exactitud. Sin embargo, en los datos de español y mezclados, con los modelos cuyas predicciones son más correctas, lo que es óptimo es eliminar tanto variables poco significativas como variables correlacionadas. Con esto se puede afirmar que lo más adecuado sería eliminar tanto las variables no significativas y las correlacionadas ya que, si se contara con una mayor cantidad de datos de los audios alemanes, esas redundancia en los datos no sería necesaria y, por lo tanto, utilizará los mismos conjuntos de variables que utilizan el resto de los datos.

Comparados los resultados obtenidos por los diferentes algoritmos, se procede a comparar con otros trabajos que han hecho uso de las mismas bases de datos para comprobar así la validez y posible mejora de los modelos. De los trabajos comentados en el capítulo 2, tres de ellos utilizan la base de datos *Berlín Emotional Database* y uno de esos tres utiliza también la *Emotional Speech Synthesis Database*.

El trabajo *Speech Emotion Recognition: Methods and Cases Study* es el único que hace uso de la base de audios en español, consiguiendo un 90,05% de aciertos utilizando una Recurrent Neuronal Networks (RNN), mientras que en este estudio se consigue aumentar ese porcentaje de acierto usando Support Vector Machine (SVM) o MultiLayer Perceptron (MLP), hasta un 96%.

Este estudio junto con los trabajos *Statistical Evaluation of Speech Features for Emotions Recognition y Speech Emotion Classification Using SVM and MLP on prosodic and voice quality features* utilizan la *Berlín Emotional Database*, consiguiendo unos porcentajes de acierto de 75,90% con Multivariable Linear Regression (MLR) para el primer trabajo; 83,17% (clasificación en 7 clases) y 95% (clasificación en 2 clases) con MultiLayer Perceptron (MLP) para el segundo, y 76,82% con Support Vector Machine (SVM) y 78,60% con MultiLayer Perceptron (MLP) para el tercero. Con estos resultados se comprueba, al igual que sucedía en este estudio, que los algoritmos SVM y MLP son los que mejores resultados presentan para la clasificación de emociones en el habla. Además, vemos que los mejores porcentajes obtenidos para los datos de audios en alemán son de un 83% para SVM y de un 84,85% para MLP por lo que, en general, mejoran los resultados posteriores; aunque se observa que al agregar los datos en dos clases el porcentaje de acierto crece significativamente.

Como se ha podido observar, los resultados obtenidos al clasificar el estado de ánimo con audios son bastante satisfactorios. Sin embargo, el objetivo de este estudio es determinar si un paciente va a sufrir un episodio de manía o depresión. Con esta finalidad se podría realizar, al igual que se procede en otros trabajos, una agrupación de emociones que reflejen si el paciente se encuentra en un estado de euforia o manía, en un estado neutral, o en un estado de depresión. El estado de euforia se corresponde con la alegría, la ira y el miedo, mientras que la depresión se corresponde con la tristeza y el asco (clasificación similar a la utilizada en trabajos como *Context-Independent Multilingual Emotion Recognition from Speech Signals*).

Para comparar los resultados de esa agrupación se escogen los modelos SVM y MLP con la base de audios españoles ya que proporcionan los mejores porcentajes, y se comparan las matrices de confusión obtenidas para ambas posibilidades. En estas matrices se situan en el eje x las clases predichas por el algoritmo, mientras que en el eje y se encuentra la clase real a la que pertenece el ejemplo; así cuando un valor se encuentra en la diagonal es debido a que se produce un acierto. Por lo tanto el objetivo es que los todos los valores distintos de 0 se encuentren la diagonal de la matriz. Cuanto mayor sea el número de ejemplos en una casilla más azul será esta. Dado que el número de ejemplos de cada clase es diferente se muestra también la matriz de porcentajes, de forma que las puede verse que clase se clasifica mejor con solo mirar la diagonal principal. En la *Figura*

44 se muestran los resultados de las SVM, en los que se observa que las clases que más confunde son Alegría con Ira y Miedo, mientras que Neutral y Tristeza es capaz de diferenciarlas con claridad.

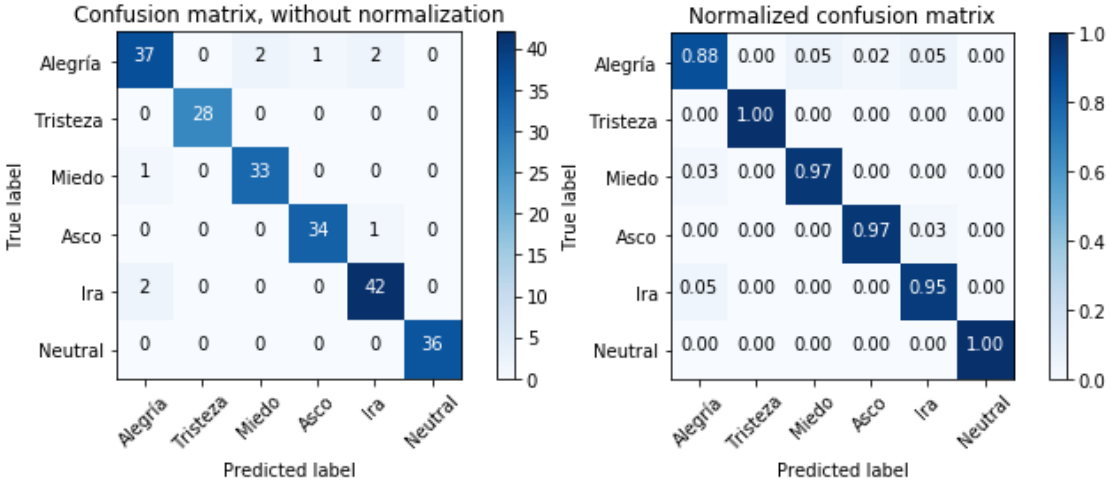

Figura 44. Matriz de confusión de los datos de audio con SVM

Los resultados que se muestran en la *Figura 45*, relativos a la clasificación con MLP, son realmente similares a los de SVM y nuevamente la clase con la que se encuentra más problemas es con Alegría.

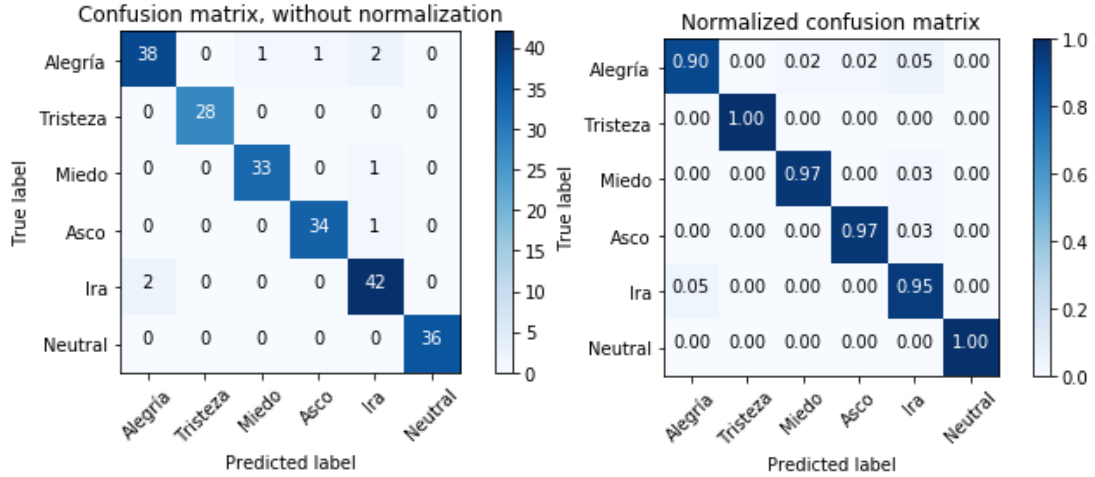

Figura 45. Matriz de confusión de los datos de audio con MLP

Si ahora se compara con los resultados de agregar las clases, *Figura 46*, los porcentajes de acierto no mejoran significativamente ya que eran muy altos, pero sí muestra que podría ser una aproximación para detectar episodios en personas con trastorno bipolar.

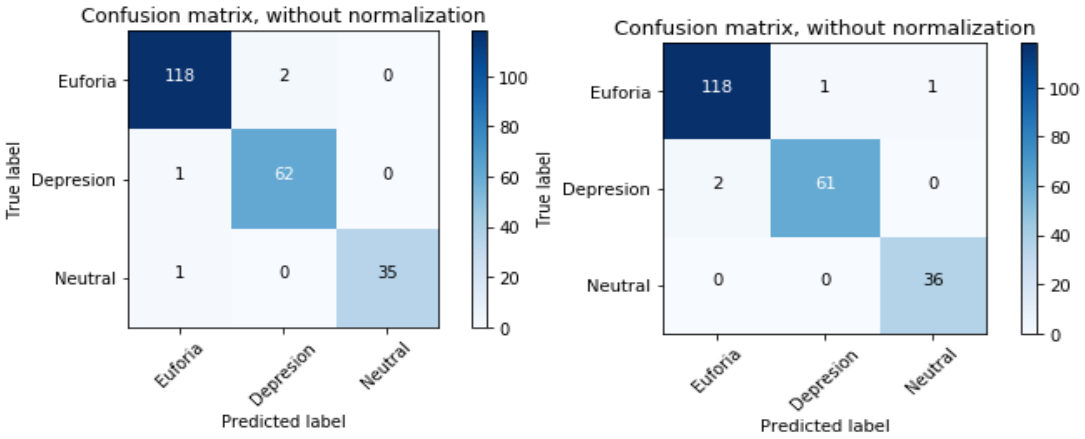

Figura 46. Matrices de confusión de los datos de audio con SVM (izquierda) y MLP (derecha) con datos agregados

6.2. Datos de actividad física

Una vez se han aplicado a los datos de actividad todos los algoritmos y se han obtenido los mejores resultados con cada uno, además de sus respectivos tiempos, se procede a realizar una comparación entre todos ellos para ver cuál es el óptimo. La *Figuras 47 y 48* muestran una comparativa de los porcentajes de acierto y de tiempo de ejecución de cada algoritmo.

	RFC	**GBC**	**SVM**	**MLP**	**NB**	**MEDIA**
Agregación 6 horas	76,36 %	73,90 %	75,32 %	76,82 %	67,72 %	74,02 %
Agregación 2 horas	70,18 %	69,72 %	80,36 %	74,95 %	61,63 %	70,82 %
MEDIA	73,14 %	71,75 %	77,76 %	75,88 %	64,53 %	

Figura 47. Tabla de porcentajes de acierto de cada algoritmo con los datos de actividad física.

	RFC	**GBC**	**SVM**	**MLP**	**MEDIA**
Agregación 6 horas	1,09	0,74	0,05	0,7	0,17
Agregación 2 horas	0,65	0,89	0,16	1,96	0,42
MEDIA	0,81	0,81	0,08	1,03	

Figura 48.Tabla de tiempos de ejecución de cada algoritmo con los datos de actividad física

Como se esperaba, las pruebas de entrenamiento con el conjunto de datos donde la actividad está agregada en periodos de dos horas tiene en general peor tasa de acierto que con los datos donde la actividad está agregada en periodos de seis horas, con un porcentaje de acierto medio del 70,82 % en el primer caso y del 74,41 % en el segundo.

Poniendo el foco en los datos agregados en periodos de 6 horas, se observa, que el máximo porcentaje de acierto ha sido obtenido mediante el algoritmo MLP con un 76,82 %, seguido del RFC con un 76,36 % y SVM con un 75,32 %. Aunque el algoritmo MLP pueda parecer a priori el mejor, tras realizar varias pruebas se ha podido llegar a la conclusión que para realizar esta clasificación funcionan mejor los algoritmos de SVM y RFC, donde los resultados son más estables.

La *Figura 49* muestra la matriz de confusión resultado de realizar la predicción con el modelo de RFC, el cual se ha entrenado con los parámetros óptimos obtenidos anteriormente.

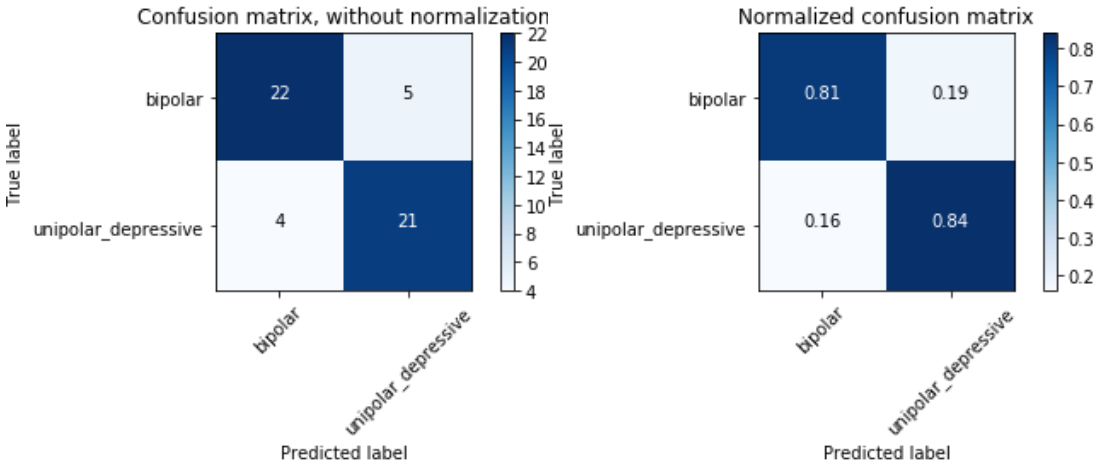

Figura 49. Matriz de confusión de los datos de actividad física con RFC

Se llega a conseguir unos porcentajes de aciertos bastante altos, en torno al 80 %. También hay que tener en cuenta que los datos de actividad totales para realizar el entrenamiento no son muchos, alrededor de 200 días entre los 22 pacientes. Teniendo una muestra mayor se podría llegar a optimizar el proceso de aprendizaje de los modelos.

Comparando los algoritmos SVM y RFC con los que se obtienen resultados similares, es el segundo el que se muestra ligeramente superior en la predicción final. Pero como se observa en la tabla de tiempos, SVM es mucho más rápido que RFC, con un tiempo 0.08 segundos de media contra un 0.81 segundos de media en el RFC. Esto puede llegar a ser relevante cuando se manejan una cantidad de datos mucho mayor, ya que puede determinar qué modelo es óptimo en función de lo que se necesite, máxima precisión o buena eficiencia, pues RFC puede llegar a ralentizar mucho el proceso de entrenamiento.

7. Conclusiones y trabajos futuros

7.1. Conclusiones

El objetivo de este trabajo ha sido el análisis de una serie de variables objetivas, obtenidas mediante dispositivos cotidianos (Smartphone y pulseras de monitorización de actividad), con la finalidad de determinar si un paciente con trastorno bipolar sufrirá un episodio en un futuro cercano. El estudio se ha centrado en la voz y en la actividad física, y se puede afirmar que existen variables que pueden ayudar a ese objetivo, permitiendo el desarrollo de un sistema que ayude a los médicos especialistas a detectar estos episodios.

En el análisis de la voz no se ha detectado una característica clara que determine el estado de ánimo de una persona, pero sí se han observado variables importantes, como el tono de la voz, velocidad del habla, la intensidad o los MFCC. Por lo tanto, agregando algunos estados de ánimo como se comenta en la sección anterior, estas variables podrían utilizarse para determinar si un paciente sufrirá o no un episodio.

Otro aspecto importante que se obtiene del trabajo es que el idioma influye a la hora de distinguir el estado de ánimo, por lo que es importante utilizar un modelo entrenado con audios del mismo idioma para mejorar la precisión del modelo; incluso, si es posible, entrenar con audios de la misma persona puesto que las características de la propia voz también pueden influir en el modelo.

En cuanto a la actividad física, se ha visto que ésta adquiere gran importancia para diferenciar si un paciente sufre de trastorno bipolar o depresión, pues los ritmos de actividad de cada tipo de paciente se diferencian con claridad a lo largo del día. Por ello se ha llegado a considerar una característica relevante a tener en cuenta para analizar trastornos de este tipo.

7.2. Trabajo futuro

En futuros trabajos la tarea principal será la de obtener un conjunto de datos cuyo origen sean personas a las que se les haya diagnosticado trastorno bipolar. Por cuestiones relativas a la protección de datos, en esta ocasión no ha sido posible disponer de los necesarios para obtener un buen modelo. También procurar que estos datos de voz y actividad física procedan de un mismo paciente para poder llegar a mejorar la eficiencia y calidad de la predicción.

Otro camino por explorar sería el análisis del sueño de los pacientes, ya que se ha visto que tiene especial relevancia a la hora de diferenciar depresión y trastorno bipolar.

Por último, sería interesante analizar los datos con un modelo de deep learning, ya que la mayoría de los algoritmos de Machine Learning utilizados en este trabajo son menos potentes y flexibles, y podría ayudar a detectar cuál es la estructura más idónea para los datos, sobre todo en lo relacionado a la voz.

7’. Conclusions and future works

7.1’. Conclusion

The objective of this work has been the analysis of a series of objective variables, obtained through daily devices (Smartphone and Wearables), to determine if a patient with bipolar disorder will suffer an episode soon. The study has focused on voice and physical activity, and it can be said that there are variables that can help that goal, allowing the development of a system that helps specialists to detect these episodes.

In the analysis of the voice a clear feature that determines the mood of a person has not been detected but important variables such as tone of voice, speed of speech, intensity or MFCC have been observed. Therefore, by aggregating some moods as discussed in the previous section, these variables could be used to determine whether a patient will suffer an episode.

Another important aspect that is obtained from the work is that the difference in the language affects the time of distinguishing the mood, so it is important to use a model trained with audios of the same language to improve the precision of the model; even, if it is possible, to train with audios of the same person since the characteristics of the voice itself can also influence the model.

As for the physical activity, it has been seen that this acquires a great importance when differentiating if a patient suffers from bipolar disorder or depression, because the rhythms of activity of each type of patient differ clearly throughout the day. Therefore, it has come to consider a relevant feature to consider for analyzing such disorders.

7.2’. Future work

In future work the main task will be to obtain a set of data whose origin are people who have been diagnosed bipolar disorder. For questions related to data protection, on this occasion it has not been possible to dispose of what is needed to get a good model. Also make sure that these data of voice and physical activity come from the same patient to be able to improve the efficiency and quality of the prediction.

Another way to explore would be the analysis of patients' sleep, since it has been seen to have special relevance when it comes to differentiate depression and bipolar disorder.

Finally, it would be interesting to analyze the data with a model of deep learning, since most of the Machine learning algorithms used in this work are less powerful and flexible and could help to detect what is the the most suitable structure for the data, especially in relation to the voice.

8. Bibliografía

[1] Berlín Emotional Database, (http://emodb.bilderbar.info/start.html).

[2] Beynon, Suzanne & Soares-Weiser, Karla & Woolacott, Nerys & Duffy, Steven & Geddes, John. (2008). Psychosocial interventions for the prevention of relapse in bipolar disorder: Systematic review of controlled trials. The British journal of psychiatry : the journal of mental science. 192. 5-11. 10.1192/bjp.bp.107.037887.

[3] De Jong, N.H. & Wempe, T. (2009). Praat script to detect syllable nuclei and measure speech rate automatically. *Behavior research methods*, 41 (2), 385 – 390

[4] Emotional speech Synthesis Database, (http://catalog.elra.info/en-us/repository/browse/ELRA-S0329/).

[5] Hozjan, Vladimir & Kacic, Zdravko. (2003). Context-Independent Multilingual Emotion Recognition from Speech Signals. International Journal of Speech Technology. 6. 311-320. 10.1023/A:1023426522496.

[6] Hsu, C.-W & Chang, C.-C & Lin, C.-J. (2003). A Practical Guide to Support Vector Classification. 101. 1396-1400.

[7] Idris, Inshirah & Salam, Md Sah & Sunar, Mohd Shahrizal. (2015). Speech emotion classification using SVM and MLP on prosodic and voice quality features. Jurnal Teknologi. 78. 10.11113/jt.v78.6925.

[8] Iliou, Theodoros & Anagnostopoulos, Christos-Nikolaos. (2009). Statistical Evaluation of Speech Features for Emotion Recognition. 2010 Fifth International Conference on Digital Telecommunications. 121-126. 10.1109/ICDT.2009.30.

[9] Jing, Shaoling & Mao, Xia & Chen, Lijiang. (2017). Prominence features: Effective emotional features for speech emotion recognition. Digital Signal Processing. 72. 10.1016/j.dsp.2017.10.016.

[10] Jing, Zhang & Zhongde, Pan & Chao, Gui & Ting, Xue & Yezhe, Lin & Jie, Zhu & Donghong, Cui. (2017). Analysis on speech signal features of manic patients.

[11] Junestrand, Axel. (2018). Application of machine learning algorithms for bipolar disorder crisis prediction, 2018.

[12] Kerkeni, Leila & Serrestou, Youssef & Mbarki, Mohamed & Raoof, Kosai & Mahjoub, Mohamed. (2018). Speech Emotion Recognition: Methods and Cases Study. 175-182. 10.5220/0006611601750182.

[13] Liaw, Andy & Wiener, Matthew. (2001). Classification and Regression by RandomForest. Forest. 23.

[14] Oudeyer, Pierre-Yves. (2003). The Production and Recognition of Emotions in Speech: Features and Algorithms. International Journal of Human-Computer Studies. 59. 157-183. 10.1016/S1071-5819(02)00141-6.

[15] Petrushin, Valery. (2000). Emotion in Speech: Recognition and Application to Call Centers. Proceedings of Artificial Neural Networks in Engineering.

[16] Patil, T.R. & Sherekar, Swati. (2013). Performance Analysis of Naive Bayes and J48 Classification Algorithm for Data Classification. Int. J. Comput. Sci. Appl.. 6. 256-261.

[17] Praat: doing phonetics by computer, (http://www.fon.hum.uva.nl/praat/).

[18] Project Jupyter, (https://jupyter.org/).

[19] Python Software Foundation, (http://www.python.org).

[20] Rashidul Hasan, Md & Jamil, Mustafa & Rabbani, Golam & Rahman, Md. Saifur. (2004). Speaker Identification Using Mel Frequency Cepstral Coefficients. Proceedings of the 3rd International Conference on Electrical and Computer Engineering (ICECE 2004).

[21] Ridgeway, Greg. (2001). The State of Boosting. Comp Sci Stat. 31.

[22] Rincón, Carmen. (2007). Diseño, implementación y evaluación de técnicas de identificación de emociones a través de la voz.

[23] Sathyanarayana, Shashi. (2014). A Gentle Introduction to Backpropagation. Numeric Insight, Inc Whitepaper.

[24] Scikit-learn, (https://scikit-learn.org/stable/).

[25] The Depresjon Dataset, (https://datasets.simula.no/depresjon/).

Anexo I: Aplicaciones para monitorización del sueño

A continuación se han analizado una serie de aplicaciones móviles para monitorizar el sueño ya que para futuros trabajos se considera interesante trabajar con ello, pues es una actividad muy relevante para detectar posibles crisis:

- **Sleep as Android:** Aplicación de pago, solo disponible en dispositivos Android, con un coste de 5,99€. En cuanto a funcionalidades al monitorizar el sueño la aplicación mide, el tiempo que el usuario permanece d4espierto, en sueño ligero, en fase REM y en sueño profundo; el porcentaje de sueño profundo; la deficiencia total, referente al tiempo para llegar a las ocho horas (puede tener valores negativos si no se llegan a 8 horas y positivos si se sobrepasan); los ronquidos y hora a la que se quedó dormido. Además, la aplicación permite valorar el sueño, desde un punto de vista subjetivo, en un sistema de puntaje basado en estrellas.

Esos datos se proporcionan mediante distintas estadísticas de forma gráfica.

También se pueden exportar los datos en csv directamente (Respaldo-> Exportar datos). En este csv se incluyen las siguientes variables:

1. Id - Identificador único de grabación
2. Tz - Timezone
3. From - Fecha y hora de inicio de la grabación
4. To - Fecha y hora de fin de la grabación
5. Sched - Siguiente alarma de fin de grabación programada.
6. Hours - Duración de la grabación
7. Rating - Valoración del usuario (0.0 - 5.0, de 0.25)
8. Comment - Comentarios del usuario y tags
9. Snore - Valor del ronquido (-1 si la opción estaba deshabilitada)
10. Noise - Media del ruido captado por la grabación
11. Cycles - Ciclos de sueño medidos (-1 si la grabación se ha insertado manualmente)
12. DeepSleep - Valor agregado del sueño profundo
13. LenAdjust - Ajusta la duración de la grabación en caso de algún periodo de no grabación (despierto, pause)
14. Geo - Valor hash de la geolocalización
15. Times - (dos líneas), la primera, contiene datos del acelerómetro agregados de los distintos periodos, la segunda, el ruido medido a lo largo de la noche.

Para ajustar la medición al gusto y condiciones del usuario, la aplicación pone a disposición de estos distintos parámetros configurables (monitoreo de sueño, la grabación de sonidos, las canciones de cuna y técnicas para sueños lúcidos y prevención del jet lag). Otras funcionalidades interesantes son la sincronización con otros dispositivos como

relojes y bombillas inteligentes, y aplicaciones como Google Fit o servicios en la nube. Es interesante remarcar que la integración con la aplicación Google Fit permitirá la obtención de los datos de sueño a través de ella.

Profundizando en la configuración del monitoreo de sueño existen algunas opciones referidas a la medición de las distintas variables, como la activación de un retraso en el monitoreo de sueño para evitar el pico inicial, o el modo de detección de las distintas fases del sueño, que puede ser configurado para realizarlo mediante el acelerómetro del teléfono o mediante sonar. El sonar se basa en el uso de los altavoces y el micrófono del teléfono, para no ser necesario tener un dispositivo (smartphone o wearable) en la cama a la hora de dormir, además de que también proporcionará un seguimiento de la respiración durante el sueño.

Continuando con el tema de los ajustes del sueño, la aplicación contempla el caso en el que el usuario duerme acompañado de otra persona. Para ello implementa una opción que, mediante la sincronización con otro dispositivo que mida desde el otro lado de la cama, permite filtrar las interferencias causadas por la otra persona.

Como se ha comentado antes la aplicación ofrece la posibilidad de sincronizarla con un reloj inteligente; esto brinda algunas posibilidades extras como el monitoreo de la frecuencia cardiaca y el pulso oximétrico (uso del oxímetro para ver que tu cuerpo mantiene una respiración y niveles de oxígeno saludables durante el sueño).

- **Sleep Better:** Aplicación gratuita disponible en Android y Iphone, aunque se puede ampliar a la versión premium por 1,99€, que ofrece algunas funcionalidades extras. Con la versión gratuita la aplicación permite detectar fases en las que el usuario permanece despierto, en sueño ligero o profundo; y añadir una valoración subjetiva del sueño, mediante un sistema de puntaje basado en emoticonos que van desde muy bien hasta muy mal. Una serie de opciones permiten determinar el contexto del sueño, es decir, actividad realizada en el día, si ha sido un día estresante, si la cama no es la del usuario…

A diferencia de la aplicación anterior, ésta no permite sincronizarla con otros dispositivos para obtener mejores resultados.

- **Sleep Tracker:** Aplicación gratuita, con una versión de pago con valor de 0,99€ para quitar los anuncios. Se ha utilizado la aplicación para monitorizar algunas noches, pero el resultado no es correcto y, además, tiene varios errores que provocan el cierre de esta.

- **Prime Nap:** Aplicación gratuita disponible en Android, con una versión de pago de 2,99€. Para monitorizar el sueño, el smartphone debe colocarse en la cama para detectar el movimiento y de esta forma determinar en qué fase del sueño se encuentra el usuario. Está medición proporciona distintas variables como el tiempo total de la grabación, el tiempo despierto y dormido (separado además en fase REM, ligero y profundo), los ciclos de sueño que tiene el usuario y, opcionalmente, se pueden añadir una descripción de

varias variables de los diversos estados de sueño que haya tenido el usuario durante el tiempo que ha estado dormido.

Esos datos se disponen de forma gráfica en un apartado de la aplicación, pero ésta permite exportar los datos en un archivo csv que contiene las siguientes variables:

1. Date - Fecha del sueño
2. Activities - Actividades realizadas en el día
3. Duration - Duración total de la monitorización
4. Duration(sec) - Duración total de la monitorización en segundos
5. Naps - Siestas
6. Times - Hora de inicio y fin de la grabación
7. Comments - Comentarios acerca de la grabación
8. Cycles - Ciclos de sueño durante la monitorización
9. Dreams - Sueños
10. REM - Tiempo en fase REM
11. REM(sec] - Tiempo en fase REM en segundos
12. Light - Tiempo en sueño ligero
13. Light(sec) - Tiempo en sueño ligero en segundos
14. Deep - Tiempo en sueño profundo
15. Deep(sec) - Tiempo en sueño profundo en segundos
16. Dream Reports - Reportes sobre los sueños

En cuanto a opciones de configuración, la aplicación no permite modificar casi nada, aunque en la pantalla inicial proporciona 5 opciones interesantes: vigilar la batería durante la monitorización, establecer la relación de que teléfono apagado es igual a dormir, monitorizar automáticamente, grabar ruidos y poner música para dormir.

Comentar que al realizar diversas pruebas con la aplicación se ha comprobado que las mediciones no eran muy precisas, y en varias ocasiones la aplicación se queda colgada. Con estas consideraciones se determina que está aplicación no es adecuada para tomar los datos de sueño que se utilizarán en el estudio.

- **SleepTime:** Aplicación gratuita que goza de una versión premium, que puede contratarse por 9,99€ al mes o 29,99€ al año. Esta versión premium incluye informes de sueño para imprimir e informes sobre el ritmo cardiaco durante el sueño.

La aplicación no permite obtener los informes de sueño desde la versión gratuita, pero de forma gráfica muestra: la duración de la monitorización, hora de acostarse y levantarse, eficiencia del sueño como un porcentaje, porcentaje de tiempo del sueño que se está despierto, porcentaje del sueño que es ligero y porcentaje del tiempo que es profundo.

Para monitorizar el sueño la aplicación informa de que es necesario colocar el móvil en la cama junto a la almohada, pero la medición sería correcta situando el dispositivo en una mesilla de altura similar a la de la cama.

Anexo II: Código de Praat

Script para crear el csv de los audios alemanes en Praat, **ScriptCrearCsv.praat**:

```
procedure extraerAudios: nameOfPath$
	if nameOfPath$ = "Alegria"
		etiqueta = 0
	elsif nameOfPath$ = "Tristeza"
		etiqueta = 1
	elsif nameOfPath$ = "Miedo"
		etiqueta = 2
	elsif nameOfPath$ = "Asco"
		etiqueta = 3
	elsif nameOfPath$ = "Ira"
		etiqueta = 4
	elsif nameOfPath$ = "Neutral"
		etiqueta = 5
	endif

	name$ = audiosPath$ + "/" + nameOfPath$ + "/" + "*.wav"
	# name$ = audiosPath$ + "/" + nameOfPath$ + "/" + "Espanol*.wav"
	Create Strings as file list: "list", name$
	numberOfSongs = Get number of strings
	for counter from 1 to numberOfSongs
		select Strings list
		name$ = Get string... counter
		name$ = audiosPath$ + "/" + nameOfPath$ + "/" + name$
		sound=Read from file: (name$)
	include ObtenerVariables.praat
	appendFileLine:nameOfResultFile$,meanAmplitude,",",minAmplitude,",",maxAmplitude,",",stdAmplitude,",",
	rangeAmplitude,",",meanPitch,",",minPitch,",",maxPitch,",",stdPitch,",",rangePitch,",",meanHarmonicity,","
	,minHarmonicity,",",maxHarmonicity,",",stdHarmonicity,",",rangeHarmonicity,",",minIntensity,",",maxIntens
	ity,",",quantileIntensity,",",meanIntensity,",",stdIntensity,",",rangeIntensity,",",speakingrate,",",articulationra
	te,",",meanEnergy,",",stdEnergy,",",minEnergy,",",maxEnergy,",",rangeEnergy,",",meanMFCC1,",",stdMFC
	C1,",",minMFCC1,",",maxMFCC1,",",rangeMFCC1,",",meanMFCC2,",",stdMFCC2,",",minMFCC2,",",max
	MFCC2,",",rangeMFCC2,",",meanMFCC3,",",stdMFCC3,",",minMFCC3,",",maxMFCC3,",",rangeMFCC3,
	",",meanMFCC4,",",stdMFCC4,",",minMFCC4,",",maxMFCC4,",",rangeMFCC4,",",meanMFCC5,",",stdMF
	CC5,",",minMFCC5,",",maxMFCC5,",",rangeMFCC5,",",meanMFCC6,",",stdMFCC6,",",minMFCC6,",",m
	axMFCC6,",",rangeMFCC6,",",meanMFCC7,",",stdMFCC7,",",minMFCC7,",",maxMFCC7,",",rangeMFCC
	7,",",meanMFCC8,",",stdMFCC8,",",minMFCC8,",",maxMFCC8,",",rangeMFCC8,",",meanMFCC9,",",std
	MFCC9,",",minMFCC9,",",maxMFCC9,",",rangeMFCC9,",",meanMFCC10,",",stdMFCC10,",",minMFCC1
	0,",",maxMFCC10,",",rangeMFCC10,",",meanMFCC11,",",stdMFCC11,",",minMFCC11,",",maxMFCC11,",
	",rangeMFCC11,",",meanMFCC12,",",stdMFCC12,",",minMFCC12,",",maxMFCC12,",",rangeMFCC12,",",
	minFormant1,",",maxFormant1,",",quantileFormant1,",",meanFormant1,",",stdFormant1,",",rangeFormant1
	,",",minFormant2,",",maxFormant2,",",quantileFormant2,",",meanFormant2,",",stdFormant2,",",rangeForm
	ant2,",",minFormant3,",",maxFormant3,",",quantileFormant3,",",meanFormant3,",",stdFormant3,",",rangeF
	ormant3,",",minFormant4,",",maxFormant4,",",quantileFormant4,",",meanFormant4,",",stdFormant4,",",ran
	geFormant4,",",minFormant5,",",maxFormant5,",",quantileFormant5,",",meanFormant5,",",stdFormant5,","
	,rangeFormant5,",",etiqueta
	endfor
	select all
	Remove
endproc

deleteFile: nameOfResultFile$
appendFileLine:nameOfResultFile$,"meanAmplitude,minAmplitude,maxAmplitude,stdAmplitude,rangeAmplitude,meanP
itch,minPitch,maxPitch,stdPitch,rangePitch,meanHarmonicity,minHarmonicity,maxHarmonicity,stdHarmonicity,rangeH
armonicity,minIntensity,maxIntensity,quantileIntensity,meanIntensity,stdIntensity,rangeIntensity,speakingRate,articulati
onRate,meanEnergy,stdEnergy,minEnergy,maxEnergy,rangeEnergy,meanMFCC1,stdMFCC1,minMFCC1,maxMFCC1,r
angeMFCC1,meanMFCC2,stdMFCC2,minMFCC2,maxMFCC2,rangeMFCC2,meanMFCC3,stdMFCC3,minMFCC3,ma
xMFCC3,rangeMFCC3,meanMFCC4,stdMFCC4,minMFCC4,maxMFCC4,rangeMFCC4,meanMFCC5,stdMFCC5,min
MFCC5,maxMFCC5,rangeMFCC5,meanMFCC6,stdMFCC6,minMFCC6,maxMFCC6,rangeMFCC6,meanMFCC7,std
MFCC7,minMFCC7,maxMFCC7,rangeMFCC7,meanMFCC8,stdMFCC8,minMFCC8,maxMFCC8,rangeMFCC8,mean
MFCC9,stdMFCC9,minMFCC9,maxMFCC9,rangeMFCC9,meanMFCC10,stdMFCC10,minMFCC10,maxMFCC10,ran
geMFCC10,meanMFCC11,stdMFCC11,minMFCC11,maxMFCC11,rangeMFCC11,meanMFCC12,stdMFCC12,minMF
CC12,maxMFCC12,rangeMFCC12,minFormant1,maxFormant1,quantileFormant1,meanFormant1,stdFormant1,rangeF
ormant1,minFormant2,maxFormant2,quantileFormant2,meanFormant2,stdFormant2,rangeFormant2,minFormant3,ma
xFormant3,quantileFormant3,meanFormant3,stdFormant3,rangeFormant3,minFormant4,maxFormant4,quantileForma
nt4,meanFormant4,stdFormant4,rangeFormant4,minFormant5,maxFormant5,quantileFormant5,meanFormant5,stdFor
mant5,rangeFormant5,Result"
@extraerAudios: "Miedo"
@extraerAudios: "Asco"
@extraerAudios: "Ira"
```

```
@extraerAudios: "Neutral"
@extraerAudios: "Alegria"
@extraerAudios: "Tristeza"
```

Script para mover los audios españoles a los directorios adecuados en Praat, **TrasnformarAudiosEspañol.praat**:

```
form Filename and number of songs
sentence audiosSourcePath ../../Universidad/TFG/S0329/INTER1SP_01
sentence audiosDestPath ./Audios
endform

procedure nombreArchivos: direct$
	if direct$ = "sessa001" || direct$ = "sessa002"
		.nombre$ = "EspanolIra"
		.destiny$ = "Ira"
	elsif direct$ = "sessf001" || direct$ = "sessf002"
		.nombre$ = "EspanolMiedo"
		.destiny$ = "Miedo"
	elsif direct$ = "sessd001" || direct$ = "sessd002"
		.nombre$ = "EspanolAsco"
		.destiny$ = "Asco"
	elsif direct$ = "sesss001" || direct$ = "sesss002"
		.nombre$ = "EspanolTristeza"
		.destiny$ = "Tristeza"
	elsif direct$ = "sessj001" || direct$ = "sessj002"
		.nombre$ = "EspanolAlegria"
		.destiny$ = "Alegria"
	elsif direct$ = "sessn001" || direct$ = "sessn002"
		.nombre$ = "EspanolNeutral"
		.destiny$ = "Neutral"
	elsif direct$ = "sessh001"
		.nombre$ = "EspanolNeutralAlto"
		.destiny$ = "Neutral"
	elsif direct$ = "sessl001"
		.nombre$ = "EspanolNeutralBajo"
		.destiny$ = "Neutral"
	elsif direct$ = "sessw001"
		.nombre$ = "EspanolNeutralLento"
		.destiny$ = "Neutral"
	elsif direct$ = "sessz001"
		.nombre$ = "EspanolNeutralRapido"
		.destiny$ = "Neutral"
	endif
endproc


procedure transformarAudios: nameOfPath$
	directory$ = audiosSourcePath$ + "/" + nameOfPath$ + "/*"
	directoryList = Create Strings as directory list: "directoryList", directory$
	numberOfDirectories = Get number of strings
	for i from 1 to numberOfDirectories
		select directoryList
		nameOfDirectory$ = Get string... i
		@nombreArchivos: nameOfDirectory$
		nameOfFiles$ = nombreArchivos.nombre$
		nameOfDestPath$ = nombreArchivos.destiny$
		nameOfDirectory$ = audiosSourcePath$ + "/" + nameOfPath$ + "/" + nameOfDirectory$
		name$ = nameOfDirectory$ + "/" + "*.l16"
		Create Strings as file list: "list", name$
		numberOfSongs = Get number of strings
		for counter from 1 to numberOfSongs
			select Strings list
			name$ = Get string... counter
			name$ = nameOfDirectory$ + "/" + name$
			sound=Read Sound from raw 16-bit Little Endian file: (name$)
			destPath$ = audiosDestPath$ + "/" + nameOfDestPath$ + "/" + nameOfFiles$ +
string$ (counter) + ".wav"
			select sound
			Save as WAV file: (destPath$)
		endfor
	endfor
endproc

@transformarAudios: "f"
@transformarAudios: "m"
```

Script para obtener las variables necesarias de los audios, **ObtenerVariables.praat**:

```
meanAmplitude = Get mean: 0, 0.0, 0.0
minAmplitude = Get minimum: 0.0, 0.0, "Sinc70"
maxAmplitude = Get maximum: 0.0, 0.0, "Sinc70"
stdAmplitude = Get standard deviation: 0, 0.0, 0.0
rangeAmplitude = maxAmplitude - minAmplitude
select sound
To Pitch: 0.0, 75.0, 600.0
meanPitch = Get mean: 0.0, 0.0, "Hertz"
minPitch = Get minimum: 0.0, 0.0, "Hertz", "Parabolic"
maxPitch = Get maximum: 0.0, 0.0, "Hertz", "Parabolic"
stdPitch = Get standard deviation: 0.0, 0.0, "Hertz"
rangePitch = maxPitch - minPitch
select sound
To Harmonicity (ac): 0.01,75.0,0.1,4.5
meanHarmonicity = Get mean: 0.0, 0.0
minHarmonicity = Get minimum: 0.0, 0.0, "Parabolic"
maxHarmonicity = Get maximum: 0.0, 0.0, "Parabolic"
stdHarmonicity = Get standard deviation: 0.0, 0.0
rangeHarmonicity = maxHarmonicity - minHarmonicity
select sound
To Intensity: 100.0, 0.0, 1
minIntensity = Get minimum: 0.0, 0.0, "Parabolic"
maxIntensity = Get maximum: 0.0, 0.0, "Parabolic"
quantileIntensity = Get quantile: 0.0, 0.0, 0.5
meanIntensity = Get mean: 0.0, 0.0, "dB"
stdIntensity = Get standard deviation: 0.0, 0.0
rangeIntensity = maxIntensity - minIntensity
select sound
include CalculoSpeechRate.praat
select sound
To MFCC: 12, 0.015, 0.005, 100.0, 100.0, 0.0
To TableOfReal: 1
meanEnergy = Get column mean (label): "c0"
stdEnergy = Get column stdev (label): "c0"
meanMFCC1 = Get column mean (label): "c1"
stdMFCC1 = Get column stdev (label): "c1"
meanMFCC2 = Get column mean (label): "c2"
stdMFCC2 = Get column stdev (label): "c2"
meanMFCC3 = Get column mean (label): "c3"
stdMFCC3 = Get column stdev (label): "c3"
meanMFCC4 = Get column mean (label): "c4"
stdMFCC4 = Get column stdev (label): "c4"
meanMFCC5 = Get column mean (label): "c5"
stdMFCC5 = Get column stdev (label): "c5"
meanMFCC6 = Get column mean (label): "c6"
stdMFCC6 = Get column stdev (label): "c6"
meanMFCC7 = Get column mean (label): "c7"
stdMFCC7 = Get column stdev (label): "c7"
meanMFCC8 = Get column mean (label): "c8"
stdMFCC8 = Get column stdev (label): "c8"
meanMFCC9 = Get column mean (label): "c9"
stdMFCC9 = Get column stdev (label): "c9"
meanMFCC10 = Get column mean (label): "c10"
stdMFCC10 = Get column stdev (label): "c10"
meanMFCC11 = Get column mean (label): "c11"
stdMFCC11 = Get column stdev (label): "c11"
meanMFCC12 = Get column mean (label): "c12"
stdMFCC12 = Get column stdev (label): "c12"
To Table: "rowLabel"
maxEnergy = Get maximum: "c0"
minEnergy = Get minimum: "c0"
rangeEnergy = maxEnergy - minEnergy
maxMFCC1 = Get maximum: "c1"
minMFCC1 = Get minimum: "c1"
rangeMFCC1 = maxMFCC1 - minMFCC1
maxMFCC2 = Get maximum: "c2"
minMFCC2 = Get minimum: "c2"
rangeMFCC2 = maxMFCC2 - minMFCC2
maxMFCC3 = Get maximum: "c3"
minMFCC3 = Get minimum: "c3"
rangeMFCC3 = maxMFCC3 - minMFCC3
maxMFCC4 = Get maximum: "c4"
minMFCC4 = Get minimum: "c4"
rangeMFCC4 = maxMFCC4 - minMFCC4
maxMFCC5 = Get maximum: "c5"
minMFCC5 = Get minimum: "c5"
```

rangeMFCC5 = maxMFCC5 - minMFCC5
maxMFCC6 = Get maximum: "c6"
minMFCC6 = Get minimum: "c6"
rangeMFCC6 = maxMFCC6 - minMFCC6
maxMFCC7 = Get maximum: "c7"
minMFCC7 = Get minimum: "c7"
rangeMFCC7 = maxMFCC7 - minMFCC7
maxMFCC8 = Get maximum: "c8"
minMFCC8 = Get minimum: "c8"
rangeMFCC8 = maxMFCC8 - minMFCC8
maxMFCC9 = Get maximum: "c9"
minMFCC9 = Get minimum: "c9"
rangeMFCC9 = maxMFCC9 - minMFCC9
maxMFCC10 = Get maximum: "c10"
minMFCC10 = Get minimum: "c10"
rangeMFCC10 = maxMFCC10 - minMFCC10
maxMFCC11 = Get maximum: "c11"
minMFCC11 = Get minimum: "c11"
rangeMFCC11 = maxMFCC11 - minMFCC11
maxMFCC12 = Get maximum: "c12"
minMFCC12 = Get minimum: "c12"
rangeMFCC12 = maxMFCC12 - minMFCC12
select sound
To Formant (burg): 0.0, 5.0, 5500.0, 0.025, 50.0
minFormant1 = Get minimum: 1, 0.0, 0.0, "hertz", "Parabolic"
maxFormant1 = Get maximum: 1, 0.0, 0.0, "hertz", "Parabolic"
quantileFormant1 = Get quantile: 1, 0.0, 0.0, "hertz", 0.5
meanFormant1 = Get mean: 1, 0.0, 0.0, "hertz"
rangeFormant1 = maxFormant1 - minFormant1
stdFormant1 = Get standard deviation: 1, 0.0, 0.0, "hertz"
minFormant2 = Get minimum: 2, 0.0, 0.0, "hertz", "Parabolic"
maxFormant2 = Get maximum: 2, 0.0, 0.0, "hertz", "Parabolic"
quantileFormant2 = Get quantile: 2, 0.0, 0.0, "hertz", 0.5
meanFormant2 = Get mean: 2, 0.0, 0.0, "hertz"
stdFormant2 = Get standard deviation: 2, 0.0, 0.0, "hertz"
rangeFormant2 = maxFormant2 - minFormant2
minFormant3 = Get minimum: 3, 0.0, 0.0, "hertz", "Parabolic"
maxFormant3 = Get maximum: 3, 0.0, 0.0, "hertz", "Parabolic"
quantileFormant3 = Get quantile: 3, 0.0, 0.0, "hertz", 0.5
meanFormant3 = Get mean: 3, 0.0, 0.0, "hertz"
stdFormant3 = Get standard deviation: 3, 0.0, 0.0, "hertz"
rangeFormant3 = maxFormant3 - minFormant3
minFormant4 = Get minimum: 4, 0.0, 0.0, "hertz", "Parabolic"
maxFormant4 = Get maximum: 4, 0.0, 0.0, "hertz", "Parabolic"
quantileFormant4 = Get quantile: 4, 0.0, 0.0, "hertz", 0.5
meanFormant4 = Get mean: 4, 0.0, 0.0, "hertz"
stdFormant4 = Get standard deviation: 4, 0.0, 0.0, "hertz"
rangeFormant4 = maxFormant4 - minFormant4
minFormant5 = Get minimum: 5, 0.0, 0.0, "hertz", "Parabolic"
maxFormant5 = Get maximum: 5, 0.0, 0.0, "hertz", "Parabolic"
quantileFormant5 = Get quantile: 5, 0.0, 0.0, "hertz", 0.5
meanFormant5 = Get mean: 5, 0.0, 0.0, "hertz"
stdFormant5 = Get standard deviation: 5, 0.0, 0.0, "hertz"
rangeFormant5 = maxFormant5 - minFormant5

Script para transformar audios sin etiquetar, **TransformarAudio.praat**:

```
form Filename and number of songs
sentence nameOfPath ./Ejemplos/
sentence nameOfResultFile sonido.csv
endform
deleteFile: nameOfResultFile$

appendFileLine:nameOfResultFile$,"name,meanAmplitude,minAmplitude,maxAmplitude,stdAmplitude,rangeAmplitude,
meanPitch,minPitch,maxPitch,stdPitch,rangePitch,meanHarmonicity,minHarmonicity,maxHarmonicity,stdHarmonicity,
rangeHarmonicity,minIntensity,maxIntensity,quantileIntensity,meanIntensity,stdIntensity,rangeIntensity,speakingRate,ar
ticulationRate,meanEnergy,stdEnergy,minEnergy,maxEnergy,rangeEnergy,meanMFCC1,stdMFCC1,minMFCC1,maxM
FCC1,rangeMFCC1,meanMFCC2,stdMFCC2,minMFCC2,maxMFCC2,rangeMFCC2,meanMFCC3,stdMFCC3,minMF
CC3,maxMFCC3,rangeMFCC3,meanMFCC4,stdMFCC4,minMFCC4,maxMFCC4,rangeMFCC4,meanMFCC5,stdMFC
C5,minMFCC5,maxMFCC5,rangeMFCC5,meanMFCC6,stdMFCC6,minMFCC6,maxMFCC6,rangeMFCC6,meanMFC
C7,stdMFCC7,minMFCC7,maxMFCC7,rangeMFCC7,meanMFCC8,stdMFCC8,minMFCC8,maxMFCC8,rangeMFCC8
,meanMFCC9,stdMFCC9,minMFCC9,maxMFCC9,rangeMFCC9,meanMFCC10,stdMFCC10,minMFCC10,maxMFCC1
0,rangeMFCC10,meanMFCC11,stdMFCC11,minMFCC11,maxMFCC11,rangeMFCC11,meanMFCC12,stdMFCC12,mi
nMFCC12,maxMFCC12,rangeMFCC12,minFormant1,maxFormant1,quantileFormant1,meanFormant1,stdFormant1,ra
ngeFormant1,minFormant2,maxFormant2,quantileFormant2,meanFormant2,stdFormant2,rangeFormant2,minFormant
3,maxFormant3,quantileFormant3,meanFormant3,stdFormant3,rangeFormant3,minFormant4,maxFormant4,quantileF
ormant4,meanFormant4,stdFormant4,rangeFormant4,minFormant5,maxFormant5,quantileFormant5,meanFormant5,st
dFormant5,rangeFormant5"

Create Strings as file list... list 'nameOfPath$'/*.wav
numberOfFiles = Get number of strings
for ifile to numberOfFiles
  select Strings list
  fileName$ = Get string... ifile
  path$ = nameOfPath$ + "/"+ fileName$
  sound = Read from file: (path$)
include ObtenerVariables.praat

appendFileLine:nameOfResultFile$,fileName$,",",meanAmplitude,",",minAmplitude,",",maxAmplitude,",",stdAmplitude,
",",rangeAmplitude,",",meanPitch,",",minPitch,",",maxPitch,",",stdPitch,",",rangePitch,",",meanHarmonicity,",",minHar
monicity,",",maxHarmonicity,",",stdHarmonicity,",",rangeHarmonicity,",",minIntensity,",",maxIntensity,",",quantileInte
nsity,",",meanIntensity,",",stdIntensity,",",rangeIntensity,",",speakingrate,",",articulationrate,",",meanEnergy,",",stdEne
rgy,",",minEnergy,",",maxEnergy,",",rangeEnergy,",",meanMFCC1,",",stdMFCC1,",",minMFCC1,",",maxMFCC1,",",r
angeMFCC1,",",meanMFCC2,",",stdMFCC2,",",minMFCC2,",",maxMFCC2,",",rangeMFCC2,",",meanMFCC3,",",std
MFCC3,",",minMFCC3,",",maxMFCC3,",",rangeMFCC3,",",meanMFCC4,",",stdMFCC4,",",minMFCC4,",",maxMFC
C4,",",rangeMFCC4,",",meanMFCC5,",",stdMFCC5,",",minMFCC5,",",maxMFCC5,",",rangeMFCC5,",",meanMFCC6
,",",stdMFCC6,",",minMFCC6,",",maxMFCC6,",",rangeMFCC6,",",meanMFCC7,",",stdMFCC7,",",minMFCC7,",",ma
xMFCC7,",",rangeMFCC7,",",meanMFCC8,",",stdMFCC8,",",minMFCC8,",",maxMFCC8,",",rangeMFCC8,",",mean
MFCC9,",",stdMFCC9,",",minMFCC9,",",maxMFCC9,",",rangeMFCC9,",",meanMFCC10,",",stdMFCC10,",",minMF
CC10,",",maxMFCC10,",",rangeMFCC10,",",meanMFCC11,",",stdMFCC11,",",minMFCC11,",",maxMFCC11,",",rang
eMFCC11,",",meanMFCC12,",",stdMFCC12,",",minMFCC12,",",maxMFCC12,",",rangeMFCC12,",",minFormant1,",",
maxFormant1,",",quantileFormant1,",",meanFormant1,",",stdFormant1,",",rangeFormant1,",",minFormant2,",",maxFo
rmant2,",",quantileFormant2,",",meanFormant2,",",stdFormant2,",",rangeFormant2,",",minFormant3,",",maxFormant3
,",",quantileFormant3,",",meanFormant3,",",stdFormant3,",",rangeFormant3,",",minFormant4,",",maxFormant4,",",qu
antileFormant4,",",meanFormant4,",",stdFormant4,",",rangeFormant4,",",minFormant5,",",maxFormant5,",",quantileF
ormant5,",",meanFormant5,",",stdFormant5,",",rangeFormant5
endfor
```

Y el último script que calcula el speech rate, y cuya creación corresponde a Nivja de Jong and Ton Wempe [3]:

```
# shorten variables
silencedb = -25
#'silence_threshold'
mindip = 2
#'minimum_dip_between_peaks'
showtext = 1
#'keep_Soundfiles_and_Textgrids'
minpause = 0.3
#'minimum_pause_duration'

# print a single header line with column names and units
#printline soundname, nsyll, npause, dur (s), phonationtime (s), speechrate (nsyll/dur), articulation rate (nsyll /
phonationtime), ASD (speakingtime/nsyll)

# read files
#Create Strings as file list... list 'directory$'/*.wav
#numberOfFiles = Get number of strings
#for ifile to numberOfFiles
#   select Strings list
#   fileName$ = Get string... ifile
#   Read from file... 'directory$'/'fileName$'

# use object ID
  soundname$ = selected$("Sound")
  soundid = selected("Sound")

  originaldur = Get total duration
  # allow non-zero starting time
  bt = Get starting time

  # Use intensity to get threshold
  To Intensity... 50 0 yes
  intid = selected("Intensity")
  start = Get time from frame number... 1
  nframes = Get number of frames
  end = Get time from frame number... 'nframes'

  # estimate noise floor
  minint = Get minimum... 0 0 Parabolic
  # estimate noise max
  maxint = Get maximum... 0 0 Parabolic
  #get .99 quantile to get maximum (without influence of non-speech sound bursts)
  max99int = Get quantile... 0 0 0.99

  # estimate Intensity threshold
  threshold = max99int + silencedb
  threshold2 = maxint - max99int
  threshold3 = silencedb - threshold2
  if threshold < minint
     threshold = minint
  endif

 # get pauses (silences) and speakingtime
  To TextGrid (silences)... threshold3 minpause 0.1 silent sounding
  textgridid = selected("TextGrid")
  silencetierid = Extract tier... 1
  silencetableid = Down to TableOfReal... sounding
  nsounding = Get number of rows
  npauses = 'nsounding'
  speakingtot = 0
  for ipause from 1 to npauses
     beginsound = Get value... 'ipause' 1
     endsound = Get value... 'ipause' 2
     speakingdur = 'endsound' - 'beginsound'
     speakingtot = 'speakingdur' + 'speakingtot'
 endfor

  select 'intid'
  Down to Matrix
  matid = selected("Matrix")
  # Convert intensity to sound
  To Sound (slice)... 1
  sndintid = selected("Sound")
```

```
# use total duration, not end time, to find out duration of intdur
# in order to allow nonzero starting times.
intdur = Get total duration
intmax = Get maximum... 0 0 Parabolic

# estimate peak positions (all peaks)
To PointProcess (extrema)... Left yes no Sinc70
ppid = selected("PointProcess")

numpeaks = Get number of points

# fill array with time points
for i from 1 to numpeaks
   t'i' = Get time from index... 'i'
endfor


# fill array with intensity values
select 'sndintid'
peakcount = 0
for i from 1 to numpeaks
   value = Get value at time... t'i' Cubic
   if value > threshold
        peakcount += 1
        int'peakcount' = value
        timepeaks'peakcount' = t'i'
   endif
endfor


# fill array with valid peaks: only intensity values if preceding
# dip in intensity is greater than mindip
select 'intid'
validpeakcount = 0
currenttime = timepeaks1
currentint = int1

for p to peakcount-1
  following = p + 1
  followingtime = timepeaks'following'
  dip = Get minimum... 'currenttime' 'followingtime' None
  diffint = abs(currentint - dip)

  if diffint > mindip
    validpeakcount += 1
    validtime'validpeakcount' = timepeaks'p'
  endif
    currenttime = timepeaks'following'
    currentint = Get value at time... timepeaks'following' Cubic
endfor


# Look for only voiced parts
select 'soundid'
To Pitch (ac)... 0.02 30 4 no 0.03 0.25 0.01 0.35 0.25 450
# keep track of id of Pitch
pitchid = selected("Pitch")

voicedcount = 0
for i from 1 to validpeakcount
  querytime = validtime'i'

  select 'textgridid'
  whichinterval = Get interval at time... 1 'querytime'
  whichlabel$ = Get label of interval... 1 'whichinterval'

  select 'pitchid'
  value = Get value at time... 'querytime' Hertz Linear

  if value <> undefined
    if whichlabel$ = "sounding"
        voicedcount = voicedcount + 1
        voicedpeak'voicedcount' = validtime'i'
    endif
  endif
endfor


# calculate time correction due to shift in time for Sound object versus
```

```
  # intensity object
  timecorrection = originaldur/intdur

  # Insert voiced peaks in TextGrid
  if showtext > 0
    select 'textgridid'
    Insert point tier... 1 syllables

    for i from 1 to voicedcount
       position = voicedpeak'i' * timecorrection
       Insert point... 1 position 'i'
    endfor
  endif

  # clean up before next sound file is opened
   select 'intid'
   plus 'matid'
   plus 'sndintid'
   plus 'ppid'
   plus 'pitchid'
   plus 'silencetierid'
   plus 'silencetableid'

   Remove
   if showtext < 1
     select 'soundid'
     plus 'textgridid'
     Remove
   endif

# summarize results in Info window
  speakingrate = 'voicedcount'/'originaldur'
  articulationrate = 'voicedcount'/'speakingtot'
  npause = 'npauses'-1
  asd = 'speakingtot'/'voicedcount'
```

Anexo III: Contribuciones de los miembros

Las contribuciones realizadas se comentan a continuación siguiendo el esquema de esta memoria.

Jose Manuel Pinto Lozano:

Realización del resumen inicial a partir de lo hablado con las tutoras y lo comentado por el alumno que realizó un proyecto similar el año anterior.

De la introducción del trabajo, redacción del contexto de la investigación tomando como ejemplo de nuevo el proyecto del año anterior, explicación de los detalles básicos de algunas de las asignaturas cursadas durante el grado por parte del alumno, planificación conjunta de una guía para la realización del proyecto, introducción a las tecnologías utilizadas, creación de un repositorio donde se han publicado a lo largo del curso los avances del proyecto y estructuración conjunta de la memoria del proyecto.

El apartado de datos fue rellenado al final una vez que se supieron que datos serían utilizados para el proyecto. Estos datos fueron obtenidos tras una profunda búsqueda de bases de datos públicas a través de la web o de otros trabajos.

Del estado del arte, indagó en busca de varios proyectos similares para conocer las técnicas más utilizadas, modelos que otorgaban malos resultados, etc. También se encargó del análisis de múltiples aplicaciones para la monitorización de la actividad física, del sueño y de la voz.

A partir de este punto, en el cual se comienzan a analizar datos, el trabajo se ha repartido de forma que uno de los miembros trata con los datos de la voz y el otro con los datos de la actividad física. Jose Manuel se encarga de la parte relativa a los audios de voz, y el primer paso para tratar con estos es extraer las variables necesarias.

Las tutoras comentaron una herramienta para el análisis de audios denominada Praat, por lo que el alumno se encargó de instalar dicha herramienta y de iniciar un proceso de aprendizaje para manejarla y poder tratar así los audios del proyecto. Para este aprendizaje se utilizó el manual de ayuda de Praat y algunos trabajos que ya habían tratado con la herramienta. Tras aprender lo necesario para ser capaz de realizar un script, el siguiente paso fue analizar que variables podían extraerse y como debía hacerse. Así finalmente se desarrollaron 4 scripts dos de ellos procesaban una jerarquía de carpetas para cargar los audios necesarios y así llamar a otro de los scripts que se encargaba de la extracción de variables. El otro script creado servía para extraer las variables de un audio sin etiquetarlo con la emoción correspondiente. Además de la creación de scripts, en alguno de los trabajos leídos por el alumno se mencionaban variables para las cuales eran necesario un procesamiento mas complejo de la señal para obtenerlas, y el alumno se encargó de buscar un script que realizase dicho procesamiento. El resultado de este trabajo fueron los csv que contenían los datos que se utilizarían a continuación.

Tras obtener las variables, realizó una pequeña exploración de los datos para ser consciente de las transformaciones que requería el conjunto de datos para más adelante aplicarlo a un modelo. Con esto, vio que serían necesario aplicar operaciones como normalizar y eliminar outliers, además de sustituir algunos valores nulos.

Implementó por tanto la función encargada de sustituir los valores nulos por la media de la variable, la función que calculaba el rango intercuartílico con la finalidad de detectar y eliminar los outliers presentes y, por último, incorporó distintos tipos de normalizaciones (4, comentadas en la memoria), para así poder comparar los resultados que arrojaban cada una. Tras este preprocesamiento realizado, se encargó de detallar todo el proceso en la memoria en la sección de preprocesamiento de datos, en los apartados de obtención de datos de audio y limpieza y preparación de los datos de audio.

Tras esto el alumno ha continuado con el análisis en profundidad de las variables, realizando una función que mostraba la correlación entre las variables, de forma que pudiese determinar cuales estaban correlacionadas. Con los mapas de calor obtenidos el alumno destacó un conjunto de variables que podrían ser suprimidas. Además de esto ha realizado un pequeño análisis de los audios para determinar que problemas relacionados con el sobreajuste o subajuste podrían encontrarse. Y, por último, en este apartado, creó un clasificador de forma que pudiese determinar que variables son las más y menos importantes para predecir.

Realizados todos estos pasos, el alumno procedió a realizar todas las pruebas posibles con los 5 algoritmos, para compararlos entre sí utilizando el porcentaje de acierto y el tiempo empleado. Probó las 96 posibilidades y analizó los resultados obtenidos separando las explicaciones en 3 en función del origen de los datos (alemán, español o mezclados) y determinó cuales eran los resultados que mejor rendimiento otorgaban para cada algoritmo.

Aunó esos resultados y comparó los diferentes algoritmos, normalizaciones y conjuntos de variables para determinar así cual de esas posibilidades era la óptima para la clasificación de los audios. Además, comparó esos resultados con otros trabajos que utilizaban las mismas bases de datos, para así validar los porcentajes obtenidos.

Para realizar algo más cercano a la predicción de episodios, el alumno realizó una agregación de los estados de ánimo, previamente debatida entre los miembros y tutoras, para así obtener un clasificador con 3 clases de salida (manía, neutral o depresión).

Otra de sus últimas aportaciones al trabajo, fue la redacción de parte de las conclusiones y trabajos futuros.

Y su última aportación fue la revisión de la segunda mitad del trabajo, para comprobar algunas erratas cometidas durante la elaboración de la memoria.

Ismael Setti Alonso:

En primer lugar, realizó un análisis comparativo de los smartwatches y smartbands existentes en el panorama actual, que podrían serles útiles para realizar una monitorización tanto de la actividad física, el sueño y otras muchas variables para poder determinar y predecir el estado de un paciente. Este análisis se centró en comparar diferentes modelos con diferentes softwares y diferenciar lo que ofrece cada dispositivo.

Para entrar en contexto sobre lo ya realizado en ésta matería, procedió a realizar una búsqueda de trabajos de investigación relacionados con la detección de emociones mediante datos de audio, sueño y actividad física. Lo más destacado fue la obtención de varios proyectos similares que trataban con datos de audio dónde pudo comprobar que tecnicas se están empleando, los algoritmos que mejor están funcionando y sobre todo cuales son las variables en los archivos de audio más determinantes a la hora de predecir estados de animo.

Posteriormente empezó a realizar una exhaustiva búsqueda de datos, tanto de audios, actividad física y sueño que pudieran apostar información valiosa de cara a la utilización de machine learning. Durante este proceso pudo encontrar dos de las bases de datos de audio usadas en el proyecto para el reconocimiento de emociones. Entre ellas una base de datos con audios en alemán (*Berlín database of emotional speech*) y otra con audios en español grabados por actores (*Emotional speech synthesis database*), esta última conseguida mediante la petición para fines académicos a la asociación francesa ELRA y con la ayuda de nuestras directoras. También consiguió obtener los datos relativos a la actividad física de pacientes con depresión y trastorno bipolar mediante la página web mencionada en el proyecto.

Tras realizar una búsqueda intensa de datos de sueño adecuados a nuestro trabajo no consiguió encontrar nada interesante y que los pudiera proporcionar los resultados que esperan con este proyecto, por lo que se decidió dejar este apartado de lado y seguir avanzando con los datos de actividad física y audio.

El apartado de datos fue rellenado al final una vez que se supieron que datos serían utilizados para el proyecto. Estos datos fueron obtenidos tras una profunda búsqueda de bases de datos públicas a través de la web o de otros trabajos.

El siguiente paso tras analizar las variables que obtuvieron en el proceso de obtención de datos, tanto de audio como actividad física, fue recopilar información sobre los algoritmos actualmente utilizados en trabajos de predicción y clasificación con tecnologías de machine learning para posteriormente poder aplicarlos a nuestros datos de la manera más efectiva. Gracias a haber cursado la asignatura de Minería de datos y Big Data supo identificar más rápidamente los modelos que mejor podrían funcionar y los parámetros a modificar para estos modelos que podrían beneficiarlos en el proceso de aprendizaje. También realizó una breve descripción del funcionamiento básico de cada algoritmo.

Tras esto cada componente del proyecto se dividió las fases de limpieza, preprocesamiento y aplicación de algoritmos, realizando Ismael la parte correspondiente con los datos de actividad física.

Para conseguir tener los datos de actividad física de la forma deseada se procedió a cargarlos y realizar diversas transformaciones para conseguir unir los datos de los pacientes con su actividad física correspondiente que se encontraba agregada en intervalos de 1 minuto en archivos csv distintos. También hubo que limpiar columnas indeseadas y partes de los archivos de actividad donde fallaba la monitorización, equilibrar los datos, obtener las variables de actividad física que pudieran influir positivamente en los resultados como la media, desviación típica, varianza, máximo, etc.

Una vez obtenidas todas las columnas deseadas probó a utilizar diferentes formas de normalización, quedándose con la que obtuvo posteriormente mejores resultados a la hora de aplicar los algoritmos.

Tras conseguir obtener un dataframe con los datos de la forma deseada procedió a realizar un análisis de estos datos. Aquí pudo observar que los datos si no estaban equilibrados por tipo de paciente y género se podía producir sobreajuste y alterar la fiabilidad de los resultados finales. También pudo comprobar mediante la realización de gráficas que la actividad de un tipo de paciente frente a otro difería de manera clara en el transcurso de un día completo. Esto diferenciación también pudo verla clara entre la actividad según el género del paciente. Todo esto contribuyo para que tomara unas decisiones u otras en la realización del proyecto.

Una vez analizados los datos y realizado los cambios oportunos procedió a aplicar cada uno de los algoritmos como se explica en la memoria del proyecto y realizar un análisis comparativo de los resultados, viendo cuales son los resultados obtenidos en cuanto a precisión y tiempo.

Por último realiza una comparación de los resultados obtenidos con cada modelo con respecto al porcentaje de acierto y el tiempo de ejecución de cada uno, y procedió a un análisis de estos.

Para finalizar, procedió a la redacción de la memoria junto a Jose que repartieron en función de los datos a estudiar de cada uno, aportando también contenido a la conclusión y la bibliografía. A parte, también realizó una posterior revisión de la primera mitad del trabajo para comprobar si existían errores en la elaboración y redacción del documento.