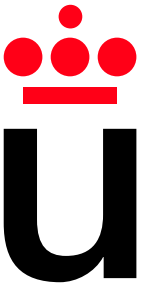


Universidad
Rey Juan Carlos

Escuela Técnica Superior
de Ingeniería Informática

Grado en Ingeniería del Software

Curso 2024-2025

Trabajo Fin de Grado

ESTUDIO DE NUEVAS CARACTERÍSTICAS SOBRE LA PARTE COMPLEJA DE LA SEÑAL PARA LA DETECCIÓN DE VOCES FALSAS

Autor: Sara González Terroba
Tutores: Antonio González Pardo, Clara Simón De Blas

Agradecimientos

Con este trabajo fin de grado se cierra esta etapa de mi vida y, aunque posiblemente en un futuro continúe mis estudios, es innegable que el primer período universitario durante el comienzo de la segunda década de nuestra vida es algo irrepetible. Me enorgullece y me llena de alegría haber vivido todo lo que me han traído mis grados, pero es algo que se debe completamente a la gente que me ha acompañado en este camino.

Mirando en retrospectiva, no me aventuro al afirmar que lo más valioso que saca uno tras dichos años no son los conocimientos técnicos, teóricos o las llamadas hard skills, si no todas aquellas aparentes nimiedades que vamos guardando en la mochila. Es aquel momento en el que profundizas tu relación con tu compañero y descubres las motivaciones vitales de alguien que decidió estudiar lo mismo que tú, brindándote una visión completamente novedosa de las vidas que hay más allá del entorno en el que te criaste. Es el momento en el que tienes que encontrar cómo gestionar problemas que no se te habían presentado antes, que implican a otros y no tienen relevancia sólo en tu vida sino en la de gente que tanto te cae bien como no tan bien. Es el momento de gestionar tu propias expectativas respecto a tus limitaciones y tus fortalezas. Y es, desde luego, el momento de trabajar de distintas formas, de adaptarte a las personas y de darte a entender en nuevos círculos, es el momento de encontrar puntos medios y, en definitiva, de expandir los límites de todo lo experimentado hasta entonces.
Cada uno de esos pequeños momentos que te marcan de por vida y construyen tu ser actual es algo que no habría podido vivir sin mis compañeros del doble grado, que desde luego ya no son compañeros, sino que me los llevo para siempre en calidad de amigos. Tampoco podría haberlo hecho sin el resto de mis compañeros, tanto del grado en matemáticas con del de software. Y mucho menos podría haber sido sin cada uno de los profesores que me han formado en estos años.

Ser profesor no es llegar a una clase e impartir un material, o lo es, pero no únicamente. La cantidad de matices que tiene dicha profesión hace que, a mi parecer, sea muy complicado conseguir ejercerla de forma destacable y por ello tengo la completa certeza de la suerte que he tenido al poder aprender de los profesores de esta universidad, pues sí son maravillosos profesores.
En especial quiero mencionar y agradecer a los que me han acompañado hasta el último momento de esta etapa, siendo mis tutores en este proyecto. Antonio

y Clara, dos personas que llevan sus labores como profesores al extremo de la excelencia, implicándose con sus alumnos y propiciando la base para que todo aquel que pasa por sus manos pueda ser en un futuro un gran profesional como ellos mismos. Por el apoyo, la motivación y el ejemplo que me han dado a lo largo de estos años, gracias.

Quiero agradecer a mi padre César, mi madre Pilar y mi hermana Alejandra, que forman parte del núcleo de mi ser. Les agradezco por proporcionarme las herramientas y los medios para estar donde estoy hoy, por el esfuerzo inigualable para acompañarme mientras voy creciendo, por llevarme a muchos sitios, por descubrirme mundo e interesarse por mis intereses, por guardarse los miedos y permitirme hacer lo que fuera necesario para conseguir un buen futuro. El entorno estable y de cariño que me han dado los tres ha sido la mejor tierra que alguien podría tener para brotar y desarrollarse. Siempre me dicen que están orgullosos de mí, pero tan sólo espero seguir creciendo y llegar a florecer y dar buenos frutos para que eso jamás deje de ser así.

Finalmente dar las gracias a mis abuelos, Pepe y Pili, por ser fuente inagotable de cariño y por siempre hacerme sentir especial.

Dedico este proyecto a mi abuelo Eloy. Si algo me entristece de cerrar esta etapa, es que él no esté a mi lado para verlo.

Resumen

Este documento busca, mediante el uso de una serie de características de la señal de la voz, estudiar el rendimiento de la parte compleja de la señal a la hora de clasificar las voces en reales o generadas. Un primer documento correspondiente al Trabajo Fin de Grado de Matemáticas recogía los fundamentos teórico-matemáticos tras el proyecto y la investigación presentes, mientras que con este, se presenta el enfoque técnico que permite la realización del mismo, analizando el proceso software que se ha llevado a cabo y profundizando en la investigación al añadir el estudio de nuevas características y el uso de un nuevo modelo.

Palabras clave:

- Python
- Ciberseguridad
- Deepfakes de voz
- Características de la voz

Abstract

This paper aims, by using a number of speech signal features, to study the performance of the complex part of the speech signal in classifying voices into real or generated voices. A first document corresponding to the Final Degree Project of Mathematics collected the theoretical-mathematical foundations behind the project and the research present, while this one presents the technical approach that allows the realisation of the same, analysing the software process that has been carried out and deepening the research by adding the study of new features and the use of a new model.

Palabras clave:

- Python
- Cybersecurity
- Deepfakes
- Voice features

Índice de contenidos

Índice de tablas

Índice de figuras

Índice de códigos

1 Introducción

1.1. Contexto y alcance

Cuando se habla de deepfakes se habla de un grupo de algoritmos que permiten generar nuevo contenido multimedia con el fin de alterar aquellos contenidos reales y verídicos. Tanto la generación de videos como la generación de voces puede resultar atractiva para fines de entretenimiento. Hace unos años se hizo completamente viral un anuncio de la compañía Cruzcampo, en el cual revivían por unos segundos a la famosa Lola Flores. Es muy llamativo e interesante tener al alcance de la mano el generar algo así y son una realidad presente en las vidas de toda la población desde hace años, tanto los llamados éticos que son generados por aquellas personas que desean hacer un beneficio para la sociedad, como desde luego aquellos dañinos desarrollados por malefactores que buscan estafar y conseguir beneficios al engañar a otras personas.

Esto genera graves problemas a muchos niveles: primero, desconfianza en la información que recibimos, pues antes una imagen, un vídeo o un audio eran medios fiables de transmisión de la información, pero bajo estas nuevas circunstancias se ve alterada la integridad de la información y no aportan una certeza completa de que lo que se ve, o se oye, realmente haya tenido lugar. También surgen problemas éticos e incluso legales, pues se pueden utilizar estos medios para atacar a las víctimas situándolas en unos hechos en los que no estuvieron o haciéndoles hacer cosas que no hicieron. ¿Cómo va a utilizar la policía un vídeo de internet para identificar a un sospechoso cuando estos medios dejan de ser fiables?

Hasta ahora, en cualquier situación social, los seres humanos utilizan la voz como método para reconocer a los demás de manera incuestionable. El reconocimiento de voz también se utiliza para la autenticación biométrica, por lo que la conversión de voz podría utilizarse de forma poco ética para violar la privacidad y la seguridad. En este caso, se habilitan las posibilidades de tergiversación y usurpación de identidad, lo que exige soluciones inmediatas desde la literatura científica.

1.2. Estado del arte

El primer ejemplo famoso de contenido multimedia manipulado data del año 1860 y consistió en una imagen de un político a quien le pusieron el rostro de Abraham Lincoln [1], realizado evidentemente mediante técnicas manuales. Esto demuestra que los deepfakes no son una consecuencia intrínseca de la evolución de la tecnología y que la creatividad y las ideas del ser humano trascienden las generaciones. Con la evolución de las tecnologías, la creación de los teléfonos móviles y portátiles y la presencia continua de estos, el entorno se vuelve mucho más digital con un crecimiento exponencial de contenidos multimedia digitales al acceso de todos. Sumado a esto, se presenta un desarrollo del campo de Machine Learning y los algoritmos que pueden modificar, o generar, contenidos multimedia de forma que al ser humano se le presentan nuevos entornos y medios para poner en práctica sus ideas independientemente de que tengan un fin nocivo o beneficioso, pues la tecnología es simplemente una herramienta más.

Ya en 1997 se presentó un proyecto académico llamado Video Rewrite Program [2] orientado a trabajar con el doblaje de las películas para reanimar los movimientos faciales de un vídeo existente al cambiarle la pista de audio a una nueva, de forma que automáticamente se consiguiera un mayor realismo al visualizar la película en los diferentes idiomas.

Poco después, en 2001 [3], se pudo ver una de las primeras veces que tomó presencia el clonado audiovisual, pues se presenta esta técnica con el fin aparentemente inocuo de replicar a un actor que había fallecido y poder terminar así la producción de la película, igual que se podría utilizar en situaciones en los que el actor no se encuentre en condiciones de terminarla por otros motivos. En cualquier caso, con esto se empezaban a plantear los primeros dilemas morales en relación a la inmortalidad digital, la esencia de una persona y el consentimiento o voluntad póstuma de la misma [4]. De alguna, forma esto permite "revivir" a la gente, al menos y hasta ahora, de forma digital y por un período concreto, pero se hace bajo unas circunstancias en las que la persona fallecida no tiene evidentemente control sobre su imagen pues se impone su rostro, su voz y gestos sobre un guión o un vídeo determinado por los creadores de dicho vídeo. Han pasado más de dos décadas y este dilema fundamental es algo que no sólo no se

ha resuelto, sino que se ha llegado a incrementar con el auge de las aplicaciones de Inteligencia Artificial que convierten las fotografías a vídeos y traen de nuevo a la vida a familiares queridos [5].

Cuándo fue el punto clave en el que se hicieron famosos los deepfakes es algo que varía dependiendo de las fuentes que se consulten, pero algo que hay en común es que la fecha ronda los años comprendidos en el segundo lustro de 2010. Evidentemente, este desacuerdo en las fechas se debe a que depende de qué se considere como fama o viralidad, ¿a qué grupos debe alcanzar o qué personas deben conocerlo?

El primer deepfake, como se conoce en la actualidad, se encontró en un foro de Reddit en 2017 y consistió en una serie de vídeos de contenido pornográfico a los que se les había impuesto los rostros de actrices famosas [6]. Poco después y en línea con este suceso, publicaron la aplicación deepNude que facilitaba la generación de desnudos falsos a raíz de una foto [7]. Estos acontecimientos ya hicieron bastante viral todo el mundo de los deepfakes, pero evidentemente era algo que principalmente conocían aquellos usuarios de foros y de contenido explícito, por lo que aún eran técnicas que pasaban desapercibidas para el público general.

En abril de 2018, se hizo famosa la aplicación FakeApp debido a que Buzzfeed la utilizó para sustituir en un número humorístico la cara y la voz del cómico Jordan Peele por la del presidente estadounidense Barack Obama[8], lo cual levantó revuelo debido al realismo conseguido con la aplicación habiendo sólo hecho uso de unos minutos de discurso original para extraer las características de estilo. Ahora se habían utilizado los deepfakes en la televisión, por lo que ahora casi cualquiera podía conocer de la existencia de los mismos y tal vez deba considerarse este como el punto clave en la expansión de los deepfakes.

Desde entonces los deepfakes no han dejado de aparecer, haciéndose especialmente virales cuando implicaban a políticos. En mayo de 2019, se editó selectivamente un vídeo de la presidenta de Estados Unidos, Nancy Pelosi, para que pareciera que arrastraba las palabras y que estaba borracha o confusa [9]. El vídeo se compartió en Facebook y recibió más de 2,2 millones de visitas en 48 horas. En la actualidad, las aplicaciones y el software que genera deepfakes es completamente accesible incluso para personas sin formación alguna en estos campos, consiguiendo generar este contenido falso en apenas segundos. A medida que la tecnología se vuelve más accesible, los problemas que ya estaban previamente presentes pero de forma aislada, se incrementan a una granularidad incluso menor, llegando a afectar de formas horribles la vida de las personas, siendo cada vez más comunes los casos de pornografía no consentida tanto a famosos, como a ciudadanos cualesquiera e incluso a niños [10].

Surge entonces y de forma inmediata la necesidad de legislar para, de alguna forma, intentar que el impacto de los perjuicios de esta tecnología sean los menores posibles. Las primeras legislaciones específicas están, sin embargo, en-

focadas principalmente al campo político y se encuentran sobre el año 2019 en China, algunos estados de Estados Unidos [11] y, a partir de 2020, en Europa. Sin embargo, y dada la urgencia que se presentaba, muchos delitos de deepfakes se regulan mediante legislación previa que abarca ciberdelitos más genéricos sobre contenidos en línea nocivos e ilegales, sin necesidad de esperar a que surja nueva legislación específica [12].

Es destacable que coloquialmente tengan una mayor presencia o viralidad aquellos deepfakes que implican un medio visual, ya sean imágenes o vídeos. Sin embargo, eso no quiere decir que a un nivel de investigación haya una exclusiva prioridad de los mismos, pues tal y como en este proyecto se tratan, los deepfakes de audio también son muy relevantes y tienen presencia tanto en la investigación científica beneficiosa como en aquellos avances que hacen agentes malintencionados.

En dicho ámbito, en el año 2017, Google presentó Tacotron [13], un modelo de síntesis del habla de tecnología Text-To-Speech (TTS) que fue evolucionando con las investigaciones, consiguiendo que al poco se introdujeron en el modelo la prosodia (para que el modelo aprendiera una correcta pronunciación de los discursos y palabras), la entonación, el control del estilo e incluso la sintetización de discursos con varios interlocutores. Desde entonces los modelos no han dejado de perfeccionarse y por ello también se investiga en el ámbito de la detección de estos deepfakes.

Cinco años después, en 2022, Lim, Chae y Lee [14] propusieron la detección de los deepfakes de audio mediante modelos convolucionales y temporales. Detectaron que los espectrogramas resultantes del deepfake solían tener menos irregularidades y eran algo más planos y menos aleatorios que uno extraído de una voz real, aunque se pensó que esto se podía deber a una falta de variedad de acentos en la base de datos que se utilizó para entrenar el modelo y no tanto en la capacidad del modelo para generar las voces. En cualquier caso, con este tipo de modelos de redes convolucionales (CNNs) y los temporales (LSTM) se consiguió una fiabilidad de entre un 97-99 % a la hora de detectar los deepfakes de voz, a costa del gran coste computacional que tienen estos modelos y considerando que es una detección que no se hace a tiempo real.

En el reto ASVspoof 2019 [15], las redes neuronales convolucionales consiguieron el menor error, un 4.04 %, que luego incluso se redujo a un 1.26 %. El problema se ha estado analizando de formas similares, Mcuba et al. [16] trabajaron con imágenes y con sus correspondientes espectrogramas, cronogramas, espectrogramas MEL y los coeficientes MEL (MFCC), de forma que obtuvo un modelo de una red convolucional VGG-16 con una precisión de detección del 85.91 %. La problemática actual existente es que los nuevos estudios comienzan a tener tasas de acierto mucho menores, rondando el 40-60 % de precisión, de forma que se presupone que debido al auge de los deepfakes y a la rápida evolución positiva que está sufriendo todo el campo de la generación, cada vez es más

compleja su detección.

Partiendo de esto y con la nueva dificultad que se ha presentado para la detección de voces falsas, en este proyecto se replantea la forma de trabajarlo, con la esperanza de que un nuevo enfoque sobre el problema aporte alguna mejora. Siempre se han trabajado con la totalidad de la señal, pero sabiendo que lo que hay tras toda esta tecnología son matemáticas y que al inicio del flujo de transformación de la señal se encuentra en el espacio complejo, surge la posibilidad de separar ambas partes real y compleja de la señal y estudiarlas por separado. Quizá el modelo sea capaz de generar voces que en su totalidad se asemejan completamente a las reales, pero al separar sus partes en el espacio complejo, no se sabe con certeza si se comportará igual o surgirán diferencias respecto a una voz real.

Es por ello que en este proyecto se trabajará con esta separación de las partes real y complejas de la señal de audio en distintos aspectos.

1.3. Estructura del documento

El documento se compone de las siguientes partes:

- **Capítulo de objetivos**
  Permite presentar los objetivos del documento y el proyecto.

- **Dataset de voces**
  Esta sección se encarga de recoger la información referente a HABLA, el dataset de grabaciones de voz que sirve como conjunto de datos para trabajar en este proyecto.

- **Metodología**
  A lo largo de esta sección se presenta cómo se ha estructurado el trabajo y aquellos requisitos que han sido definidos, junto con un diagrama de casos para aportar una mayor comprensión del funcionamiento del código.

- **Desarrollo del proyecto**
  A lo largo de esta sección se revisa el desarrollo del trabajo, cómo se han usado y aplicado las herramientas, las características que se han implementado y los modelos entrenados.

- **Resultados**
  La parte del documento que recapitula los resultados obtenidos tras el estudio y el entrenamiento de los modelos, poniendo énfasis en aquellas características investigadas.

- **Conclusiones**
  Tras observar los resultados, en esta sección se extraen las conclusiones.

- **Trabajos futuros**
  Finalmente, en este apartado se presentan algunos posibles trabajos futuros que pueden resultar de interés.

2

Objetivos

Este trabajo presenta la parte técnica del proyecto para el estudio e investigación de nuevas características de la voz tras la separación de la partes compleja y real de la señal, con el fin de saber si dichas nuevas características son útiles al detectar deepfakes de voz.

2.1. Selección y cálculo de características sobre las partes compleja y real de la señal.

Se estudiarán las posibles características que extraer tras separar la parte compleja de la parte real de la señal del audio para seleccionar aquellas que tengan más potencial. Una vez seleccionadas, se deberán calcular dichas características mediante la implementación del software y el correcto uso de las herramientas disponibles como distintas librerías de python.

2.2. Extracción de conclusiones sobre las nuevas características empleadas.

Haciendo uso de dos modelos entrenados con las características calculadas en este proyecto, se analizarán los resultados para considerar la relevancia de las mismas. Se hará uso de modelos discriminativos que hayan demostrado un buen

rendimiento y sean eficaces en las predicciones, además de ser necesario que haya forma de interpretarlos y extraer una explicación de aquellas características que han utilizado en mayor medida a la hora de clasificar.

Se definen dos subobjetivos:

1. Implementación de los modelos que se van a entrenar.
2. Contraste y análisis de los resultados obtenidos gracias a los modelos entrenados con las características.

2.3. Uso de un proceso software para el correcto desarrollo del proyecto.

Se deberá hacer uso de los procedimientos que definen un proceso software para asegurar la correcta estructuración del proyecto, el cumplimiento de los requisitos y necesidades del mismo así como favorecer la eficiencia y eficacia en el desarrollo.

Con ese fin se plantean los siguientes subobjetivos:

1. Elección de la metodología software más adecuada para el desarrollo del proyecto.
2. Redacción de los requisitos funcionales y no funcionales.
3. Validación del proceso al finalizar el desarrollo del proyecto.

3

Dataset de voces. HABLA

En este proyecto se hace uso del dataset HABLA debido a que es un conjunto de voces generado para el entrenamiento de modelos de detección de voces falsas y ello aporta una serie de ventajas [17]. Otro motivo para la selección de este dataset es el idioma, ya que globalmente existe una predisposición a que la investigación se realice en un ámbito angloparlante, debido también a la mayor cantidad de recursos disponibles, pero eso no es indicativo de que luego sea menos necesario la detección de deepfakes de voz en otros idiomas. De hecho, los deepfakes son más peligrosos para colectivos vulnerables de la sociedad (personas de edad avanzada, personas en la infancia o con desconocimiento tecnológico) que si no son de países angloparlantes, posiblemente no sepan hablar inglés y se encuentren en ataques realizados en su lengua nativa.

Este dataset está compuesto de grabaciones de voz en español latinoamericano y, ya que su generación estaba pensada específicamente para este uso, se encuentra normalizado y con los nombres establecidos de forma que permiten saber la naturaleza del audio; en caso de ser generado, la técnica con la que se generó y además la demografía del hablante en términos de acento y género.

Una gran ventaja de este dataset es que la generación de las voces falsas no se ha realizado únicamente mediante una técnica, sino que se han empleado cuatro distintas, aportando una mayor variedad y robustez a los datos. Un modelo que trabaje con este dataset para la detección de deepfakes estará expuesto a un mayor número de matices que puedan surgir dependiendo de la técnica generativa que haya creado la voz, consiguiendo así que el modelo discriminativo tenga unos mejores resultados al clasificar las voces falsas que si sólo se utilizaran voces

generadas por una técnica para el entrenamiento y luego se le presentaran otras técnicas.

Se considera de interés profundizar brevemente en las tecnologías que generaron los audios, con el fin de comprenderlas y entender por qué supone una ventaja que exista una mayor variedad de técnicas generativas a la hora de entrenar un modelo discriminativo. Algo a considerar es que las tecnologías que se han utilizado son generativas de imágenes y que, gracias a los espectrogramas, se hace uso de ellas para generar audios posteriormente mediante una transformación.

16.000 grabaciones generadas mediante CycleGAN. La tecnología CycleGAN es un tipo concreto de GAN (Generative Adversarial Network). Una GAN estándar se compone de dos submodelos: uno generativo que crea imágenes para engañar al modelo discriminativo y el discriminativo que pretende conseguir diferenciar imágenes reales de las que ha generado el generativo; de forma que tras una serie de iteraciones el modelo generativo es lo suficientemente bueno como para engañarle. Partiendo de esto, la CycleGAN tiene un funcionamiento análogo, con la diferencia de que se encarga de generar imágenes con ciertas características cuando no existen pares de imágenes de los que aprender, ya que no genera las imágenes a partir de ruido, sino de otras imágenes. En concreto, este dataset se utiliza para que mediante los espectrogramas se puedan generar voces con características especiales, de forma que se asemejen más a lo deseado y sirvan de entrenamiento para el modelo [18].

16.000 grabaciones generadas mediante un modelo de Difusión. Los modelos de difusión son modelos que nacen de la física estadística y pretenden comprender cómo se difunden las cosas con el tiempo, en este contexto, los píxeles de la imagen. Gracias a observar cómo la imagen clara del espectrograma se vuelve ruidosa, aprende a hacer lo contrario: empezar con ruido puro gaussiano y aclararlo gradualmente mediante un procedimiento llamado *denoising* para revelar la imagen final o solución deseada [19].

16.000 grabaciones generadas mediante una StarGan. Al igual que la CycleGAN, la StarGAN es un subtipo de GAN con la especialidad que permite generar las voces variando cualidades de ellas, como podría ser en el caso de este dataset, el acento, pero también puede modificar características como el estilo o el timbre de la voz. Cada una de estas cualidades se denomina dominio y por eso esta tecnología, al generar nuevas voces con estos cambios, se dice que es una GAN multidominio. Al hacer uso de un modelo discriminativo se asegura que el resultado parece auténtico, obteniendo muy buenos resultados, al igual que sucede con los de la CycleGAN [20].

10.000 grabaciones generadas mediante modelos TTS. Los modelos Text-To-Speech parten del un texto como input y le ponen voz mimetizando las

características como el estilo y el timbre de un locutor sobre el que han sido previamente entrenados. Dentro de las generadas en esta categoría: 5,000 fueron generadas por el modelo base TTS, mientras que 2,500 son de un modelo Diff-TTS, que combina la generación de voz a partir de texto con la metodología iterativa y el *denoising* de los modelos de difusión; y otras 2,500 mediante TTS-StarGAN, generando las voces a partir del texto y permitiendo el alterar las características de la voz albergando distintos locutores en un mismo modelo [21].

La Figura 3.1 muestra la clasificación de las diferentes muestras del dataset HABLA en base a con qué tecnología se han generado. El tamaño de cada sección es proporcional al número de audios de esa clase respecto al total del dataset.

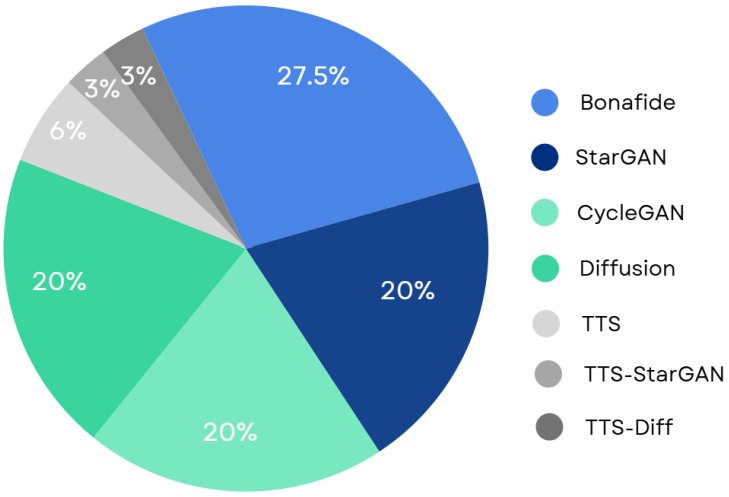


Figura 3.1: Diagrama de la distribución de muestras de audio en función de la técnica de generación en HABLA.

4

Metodología

4.1. Metodología de desarrollo

Se comienza esta sección con el detalle de la metodología software que ha sido aplicada en el proyecto. Por la naturaleza novedosa del tema se optó por una metodología ágil, ya que así se conseguía una mayor flexibilidad y adaptación a medida que se iba investigando, permitiendo una refinación de los requisitos en las distintas iteraciones.

En las primeras etapas del proyecto se realizaron sprints bisemanales principalmente enfocados a la búsqueda de información general y estudio de la misma y contextualización del problema, estableciendo así unos requisitos iniciales y teniendo reuniones con los tutores al final de cada sprint para discutir los distintos avances y descubrimientos que tuvieron lugar en ese período de tiempo.

A medida que el proyecto fue avanzando, en las últimas etapas los sprints presentaban una duración de una única semana, al necesitar una mayor revisión y agilidad por parte de los tutores, pues al enfocarse más en la parte aplicada aumentaba el número de avances en menor tiempo. Además, se tomaban nuevas decisiones y se modificaban los requisitos iniciales al ir observando los resultados obtenidos. Para el final de cada sprint siempre se revisaba el trabajo realizado, poniendo énfasis en analizar con retrospectiva aquel trabajo que obtuvo buenos resultados frente a aquello que no funcionó, considerando posibles ajustes tras la experiencia aprendida y permitiendo la adaptación a las nuevas necesidades que hubieran surgido.

4.2. Requisitos

A continuación se especifican los requisitos definidos. En este proyecto no se presentan requisitos de interfaz al no contar con un desarrollo de la misma, y por tanto, únicamente se tienen en cuenta aquellos requisitos funcionales y no funcionales.

4.2.1. Requisitos funcionales

Código	**RF_1**
Nombre	Carga de los ficheros de audio del dataset
Descripción	El código debe acceder a los ficheros almacenados en Google Drive para poder usarlos en el resto de funcionalidades, trabajar con ellos y extraer sus características.
Nivel de importancia	Muy alta.

Código	**RF_2**
Nombre	Reproducción de los audios
Descripción	Se debe poder reproducir cualquier audio almacenado en el dataset desde una función en el código para poder hacer comprobaciones y tener sensibilidad de los datos.
Nivel de importancia	Media.

Código	**RF_3**
Nombre	Representación gráfica de las ondas sonoras
Descripción	Se necesita proporcionar una representación gráfica de las ondas sonoras de cualquier audio de voz para comprender mejor la estructura de los mismos, como por ejemplo, si tienen silencios al inicio o al final.
Nivel de importancia	Media.

Código	**RF_4**
Nombre	Creación de espectrogramas
Descripción	Se requiere la generación de espectrogramas de cada audio para separar la parte real y la parte compleja de la señal.
Nivel de importancia	Muy alta.

Código	**RF_5**
Nombre	Visualización de espectrogramas
Descripción	La representación de los espectrogramas calculados permite comprobar de forma visual si existen diferencias entre la parte compleja y la parte real de la señal de la voz, lo que proporciona una primera aproximación de si el tema planteado puede tener resultados positivos.
Nivel de importancia	Alta.

Código	**RF_6**
Nombre	Estudio de la distribución de los valores de la señal
Descripción	Se debe estudiar si los valores de la parte compleja y la parte real de la señal de la voz tras haberlas separado siguen la distribución normal.
Nivel de importancia	Baja.

Código	**RF_7**
Nombre	Cálculo de estadísticos
Descripción	El software ha de calcular los siguientes estadísticos como parte de las características de la voz: media, desviación estándar, percentiles 5, 10, 50, 90, 95.
Nivel de importancia	Alta.

Código	**RF_8**
Nombre	Cálculo de características de slope
Descripción	El software ha de calcular el slope para segmentos con voz y sin voz entre los 0 y 500 Hz para ambas partes de la señal.
Nivel de importancia	Alta.

Código	**RF_9**
Nombre	Cálculo de características de loudness
Descripción	El software ha de calcular los percentiles de loudness para segmentos con voz y sin voz entre los 0 y 500 Hz para ambas partes de la señal.
Nivel de importancia	Alta.

Código	**RF_10**
Nombre	Cálculo de características del f0
Descripción	La frecuencia fundamental debe calcularse para ambas partes de la señal considerando los semitonos desde la frecuencia 27,5 Hz.
Nivel de importancia	Alta.

Código	**RF_11**
Nombre	Cálculo de características del shimmer
Descripción	Tanto en la parte real como en la compleja, el software ha de calcular el shimmer local.
Nivel de importancia	Alta.

Código	**RF_12**
Nombre	Cálculo de características del jitter
Descripción	El jitter local debe calcularse para ambas partes de la señal.
Nivel de importancia	Alta.

Código	**RF_13**
Nombre	Extracción de características de control
Descripción	Debe extraerse un grupo de características que por estudios esté demostrado de su rendimiento en este ámbito, para que sirvan de control y contraste a las nuevas características.
Nivel de importancia	Muy alta.

Código	**RF_14**
Nombre	Almacenamiento persistente de las características
Descripción	El código debe permitir el almacenamiento de las características de forma persistente para evitar la pérdida de datos. Se hará uso de OpenSMILE para este fin.
Nivel de importancia	Muy alta.

Código	**RF_15**
Nombre	Selección de datos por cross-validation
Descripción	Debe implementarse una parte del código que realice correctamente la selección de los conjuntos de entrenamiento y test para cada experimento, con el fin de asegurar que no se produzca overfitting.
Nivel de importancia	Muy alta.

Código	**RF_16**
Nombre	Visualización de las distribuciones de las características
Descripción	Se han de generar diagramas de densidad e histogramas para que se pueda observar cómo se distribuyen los valores de las características calculadas sobre la parte real y la parte compleja, con el fin de que esta información apoye a los resultados para extraer conclusiones.
Nivel de importancia	Media.

Código	**RF_17**
Nombre	Entrenamiento de modelos
Descripción	Es necesario entrenar dos modelos diferentes con las características calculadas para observar los resultados que se obtienen al usarlas.
Nivel de importancia	Muy alta.

Código	**RF_18**
Nombre	Explicación de los modelos
Descripción	Se debe implementar código que permita interpretar las predicciones de los modelos, indicando aquellas características que han sido de mayor utilidad.
Nivel de importancia	Muy alta.

Código	**RF_19**
Nombre	Evaluación de los modelos
Descripción	Es necesario el uso de métricas como la precisión o el recall que indiquen el rendimiento de los modelos de forma objetiva, contrastando si dependiendo del modelo varía la importancia de cada característica al hacer las predicciones.
Nivel de importancia	Muy alta.

4.2.2. Requisitos no funcionales

Código	**RNF_1**
Nombre	Uso de python para la implementación del código
Descripción	Se utilizará Python para todo el código referente al proyecto debido a la familiaridad con el campo de Machine Learning y las librerías existentes tanto de ML como de tratamiento de audio.
Nivel de importancia	Muy alta.

Código	**RNF_2**
Nombre	Uso de Google Colab como entorno de desarrollo
Descripción	El código que se implemente se hará en un notebook de Google Colab debido a la comodidad de colaboración con los tutores, la capacidad de computación del entorno y la fácil conexión con el dataset almacenado en Google Drive.
Nivel de importancia	Muy alta.

Código	**RNF_3**
Nombre	Compatibilidad con distintos dispositivos
Descripción	Aunque de cara a la investigación no es completamente imprescindible, se considera un requisito valioso que el proyecto se pueda trabajar cómodamente sin necesidad de una configuración compleja del entorno, y que por ejemplo, se pueda ejecutar el código o redactar la documentación independientemente de si se usa un dispositivo con un sistema operativo Windows o macOS.
Nivel de importancia	Alta.

Código	**RNF_4**
Nombre	Escalabilidad
Descripción	El proyecto debe cimentarse de forma que sea sencillo la inclusión de nuevas características para el entrenamiento del modelo, el uso de distintos modelos o incluso la modificación en los archivos que componen el dataset.
Nivel de importancia	Muy alta.

Código	**RNF_5**
Nombre	Eficiencia y rendimiento
Descripción	El software implementado debe ser eficaz y tener buen rendimiento, por lo que se hará uso de las últimas versiones de las librerías y paquetes siempre que se mantenga la compatibilidad entre ellas, se evitará código deprecado y se hará uso de modelos cuyo rendimiento esté demostrado.
Nivel de importancia	Muy alta.

4.3. Casos de uso

Los diagramas de caso de uso se utilizan para visualizar cómo los diferentes roles de usuario interactúan con el sistema y comprender, en mayor detalle, las funcionalidades que tiene el mismo. Aunque en este proyecto, el sistema únicamente tiene un rol de usuario (el investigador y desarrollador), se considera de interés desarrollar brevemente esta sección para observar con mayor claridad aquello que se consigue de los requisitos y tener una visión a alto nivel de lo que se ha implementado en el código y cómo funciona.

Figura 4.1: Diagrama de casos de uso del sistema.

Como se puede observar en el diagrama 4.1, existe una gran dependencia entre la mayoría de los casos de uso con sus previos, algo bastante lógico pues la salida de uno es lo que permite funcionar correctamente al siguiente, por ejemplo, no se podrían extraer las características en ningún caso sin tener acceso antes a los audios; y mucho menos tendría sentido calcular las nuevas características sin generar los espectrogramas y separar las partes real y compleja.

Para entrenar los modelos, no se requiere siempre de los tres casos que se presentan en el diagrama con la relación *include*, pero sí que se requiere de al menos una de ellas, ya sea calculando las características en la misma ejecución mediante código o mediante OpenSMILE o bien sea cargando unas características previamente calculadas. Es por eso que se elige ese tipo de relación y no otra como podría ser *extend*.

Finalmente, mencionar que se sitúa OpenSMILE como actor pues en este proyecto tiene un papel muy relevante al extraer las características, siendo un sistema externo que ejecuta su código y actúa sobre los audios de forma que se considera un papel activo, en vez de un papel pasivo como podría ser el caso del almacenamiento Google Drive.

5

Desarrollo del sistema

5.1. Estudio de las características

Partiendo de las grabaciones de voz del dataset HABLA y los requisitos iniciales, el siguiente paso en la metodología es calcular las características que van a permitir la clasificación de las voces. Pueden considerarse dos grupos de características en este proyecto, aquellas globales sobre el total de la señal y aquellas de nueva implementación que se calculan únicamente sobre la parte compleja o la parte real de la señal de la voz.

Para el primer grupo, en la sección 5.1.1, se hace uso de un conjunto de características estandarizado que se obtienen mediante la librería OpenSMILE. Estas características conformarán el conjunto de control para contrastar los resultados de las nuevas características.

En segundo lugar, se extraen las características que realmente se están investigando, siendo las que se calculan sobre la parte real y sobre la parte compleja, por separado. En la sección 5.1.2 se analiza en detalle el proceso del cálculo de las mismas, agrupándose en seis grupos: estadísticos, Slope, Loudness, F0, Shimmer y Jitter.

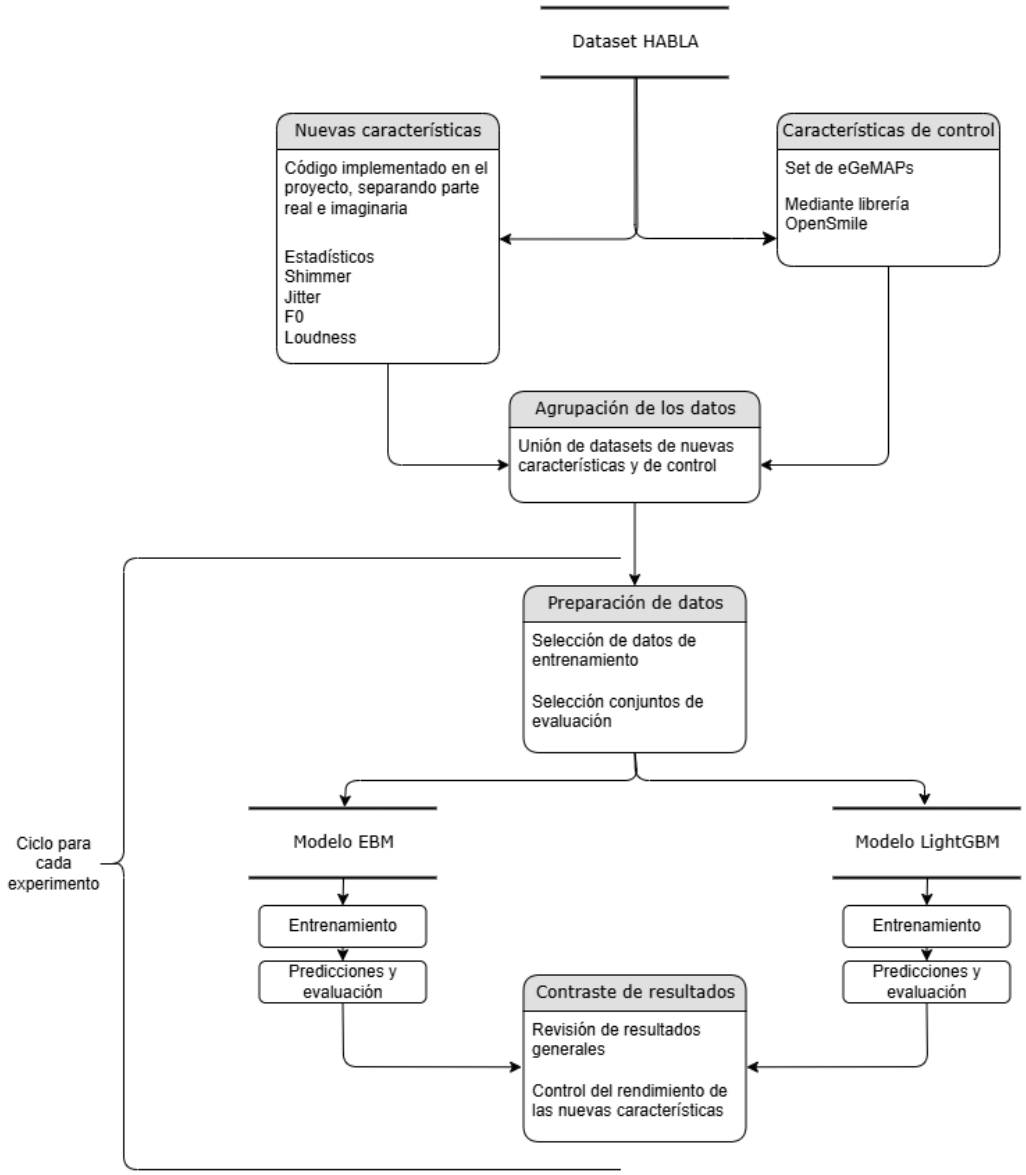


Figura 5.1: Diagrama de la metodología seguida en el proyecto.

En la figura 5.1 se puede observar un diagrama que muestra el flujo dentro de la metodología usada: partiendo del dataset HABLA se calculan las características estandarizadas de GeMAPs y las nuevas sobre la parte real y la parte compleja. Una vez obtenidas se agrupan de forma que sirven para el entrenamiento en los experimentos. Se considera que un experimento dura desde que se seleccionan y preparan los conjuntos de datos que servirán para entrenar y evaluar un modelo, hasta que finaliza con el análisis de los resultados obtenidos con el modelo.

5.1.1. OpenSMILE

OpenSMILE (Open Speech and Music Interpretation by Large Feature-Space Extraction) [22] es una librería de Python diseñada para la extracción de características de audio. Dispone de tres sets de características (`ComParE_2016`, `GeMAPS` and `eGeMAPS`) con distintos niveles y versiones que calculan una cantidad diferente de características. `ComParE_2016` es el más grande, llegando a alcanzar una cantidad de 6.373 características extraíbles.

Para este proyecto y, como se explicó en la memoria correspodiente al TFG del Grado en Matemáticas [23], se trabaja con `GeMAPSv01b` [24] debido al coste computacional. En cualquier caso, todos los conjuntos cuentan con dos niveles: se encuentra la posibilidad de calcular las características a nivel *Functionals* donde se hallan de forma global para el fichero, o se puede calcular por períodos o tramos del mismo, lo que se llaman LLDs (low-level descriptors). Para esta situación es más adecuado utilizar el nivel *Functionals* puesto que el objetivo es determinar globalmente si un audio ha sido generado o no, de forma que no se requiere un nivel de detalle a nivel de segmentos del mismo.

El proceso de extracción de las características es transparente para el usuario y la configuración del mismo es completamente sencilla, como se puede revisar en el apéndice A.2.6.

Una vez OpenSMILE devuelve las 32 características calculadas sobre los 15.800 ficheros de audio en formato dataframe, se guarda el resultado en un fichero pickle. Un dataframe es una estructura de datos bidimensional similar a una tabla con filas y columnas. Además, una ventaja que tienen, es que la estructura es definible por el usuario, de forma que puede tener índices multidimensionales. Es decir, si la clave primaria para el registro consiste en más de un dato, como podría ser el nombre del fichero y su duración, ambos formarán parte del índice. Esto sería especialmente útil si se quisiera calcular las características por períodos de tiempo, como pasaría si se calcularan en el nivel LLD. Por otro lado, un fichero pickle es un tipo de fichero que, mediante un proceso de serialización, permite el almacenamiento de estructuras de datos como conjuntos o tuplas.

5.1.2. Cálculo de nuevas características

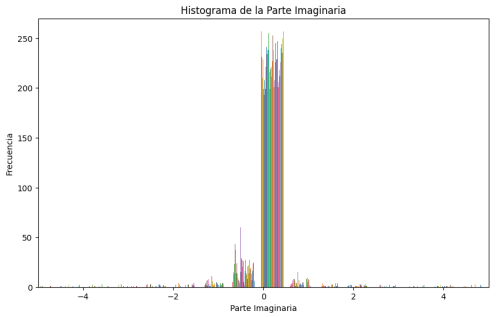
Cuando se trabaja en el estudio y clasificación de voces existen una gran cantidad de características a la disposición de los investigadores, siendo algunos conjuntos ya predefinidos y estandarizados que permiten calcularlas sobre los datasets de voces, como es el caso de los de OpenSMILE, o Time Series Feature Extraction Library (TSFEL) [25].

En un primer estudio, se decide que la mejor opción para comenzar es analizar

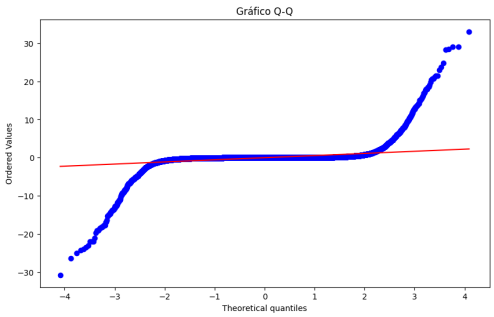
cómo se comportan ambas partes real y compleja con los estadísticos usuales, y estos son, la media, la desviación estándar y una serie de percentiles. Posteriormente, se eligen las características de `eGeMAPs` que en el primer estudio dieron mejores resultados y sus análogas, siendo estas las referentes a loudness, shimmer, jitter y slope.

Cabe mencionar que en todo el trabajo con las características se hace uso de espectrogramas pues, tal y como se vio en el primer proyecto que se enfocaba en la parte teórico-matemática [23], es lo que lleva el audio al espacio complejo y permite hacer esa separación entre la parte real y la parte compleja. Para más detalle sobre ellos a un nivel teórico, consultar el primer proyecto; mientras que el código utilizado se puede consultar en el apéndice B.

Antes de empezar con la extracción de las características, se analiza la distribución que tienen la parte real y la compleja de una muestra del dataset, para establecer si se corresponde con la distribución normal [26, 27]. Se visualizan los histogramas, el gráfico Q-Q y las pruebas de Shapiro [28] y Kolmogorov-Smirnov [29] con el código que se puede encontrar en el apéndice A.1.5.



(a) Histograma.



(b) Gráfico Q-Q.

```
Prueba de normalidad de Kolmogorov-Smirnov
Estadístico de KS: 0.3655506374389491
Valor p: 0.0
Se rechaza la hipótesis nula: los datos no siguen una distribución normal.
```

(c) Prueba de Kolmogórov-Smirnov.

Figura 5.2: Pruebas de normalidad estudiadas sobre la parte compleja de una voz del conjunto de prueba.

Las figuras 5.2 sirven de ejemplo sobre los resultados que se obtuvieron de las pruebas de normalidad. En todas las unidades de la muestra, el resultado fue que no seguían una distribución en ningún caso, por lo que se presupone que es el comportamiento que sigue la mayoría los datos del dataset.

A continuación se va a analizar en detalle cómo se han calculado mediante el código las distintas características seleccionadas. Los códigos pertinentes se

pueden encontrar en el apéndice A.2.

Estadísticos básicos

Para el cálculo la media, la desviación estándar y los percentiles se ha utilizado una misma función A.8. Esta función recibe la matriz del espectrograma de la parte real o compleja según se le pase en la llamada (`np.imag(D_stft)` selecciona la parte imaginaria de la transformada de Fourier y con `np.real(D_stft)` se obtiene la parte real). Sobre esta matriz se calculan[1], y gracias a las funciones de NumPy, los distintos percentiles[30], la media[31] y la desviación estándar[32], para que finalmente devuelva los valores de estas características calculadas.

Por otro lado, para almacenar el output de la función, en un dataframe que albergue los nombres de los ficheros, se incluyen las nuevas columnas. Mediante un bucle que itera sobre cada uno de los ficheros nombrados en el dataframe, carga el audio y calcula su transformada de Fourier gracias a la librería `librosa`. En cuanto tiene la transformada, llama a la función y al terminar asigna los valores a cada una de las columnas correspondientes (consultar Código A.8).

	filename	ImagPartPercentile_5	ImagPartPercentile_10	ImagPartPercentile_50	ImagPartPercentile_90	ImagPartPercentile_95	ImagPartMean	ImagPartStd
	DATA/asvspoof2019_LA/train_dev/arf_00295_01212...	-0.501703	-0.06	0.0	0.055859	0.529107	-0.0	0.678047
	DATA/asvspoof2019_LA/train_dev/arf_00295_01212...	-1.176957	-0.160341	0.0	0.158928	1.075645	0.0	0.638837
	DATA/asvspoof2019_LA/train_dev/arf_00295_00020...	-1.582135	-0.129684	0.0	0.120395	1.857087	0.0	0.697954
	DATA/asvspoof2019_LA/train_dev/arf_00295_00389...	-0.495066	-0.059371	-0.0	0.05981	0.649975	0.0	0.534486
	DATA/asvspoof2019_LA/train_dev/vem_08784_01339...	-0.189194	-0.004314	-0.0	0.003635	0.226406	0.000001	0.333913
	DATA/asvspoof2019_LA/train_dev/TTS-StarGAN-com...	-0.864793	-0.166892	0.0	0.162585	0.715968	-0.000001	0.855949
	DATA/asvspoof2019_LA/train_dev/vem_09697_02056...	-1.250158	-0.153491	0.0	0.144107	1.25004	-0.0	0.665372

Figura 5.3: Cálculo de las características sobre una muestra del Dataset.

Slope

El slope se refiere a la pendiente de la energía. Lo más común es usar la media y suelen dar buenos resultados cuando se analiza en el rango de 0 a 500 Hz. Se puede estudiar tanto para segmentos donde no hay voz (unvoiced, UV) o con voz (voiced, V), de forma que las características se corresponden a las de `GeMAPs` `slopeUV0-500_sma3nz_amean` y `slopeV0-500_sma3nz_amean`.

El código contiene las mismas partes que en las otras características que se calculan: la definición de la función que realiza el cálculo, la creación de las columnas vacías en el dataframe y el bucle sobre los ficheros donde se cargan, se calcula la transformada de Fourier y se llama a la función. La función es un poco más compleja que las que calcula los estadísticos básicos.

[1]El orden de cálculo es primero por filas y después por columnas, ya que las columnas representan la ventana temporal de la transformada de Fourier, como se analizó en la sección 3.2.5 del primer trabajo. [23]

Para calcular el slope se debe aplicar una regresión lineal al intervalo espectral considerado, en este caso entre los 0 y 500 Hz, filtrando esos valores en la transformada. Además, se filtran las ventanas por el percentil 20 de la energía, ya que una media de energía muy baja es indicativo de un segmento sin voz [33]. Se obtiene una recta de la forma $G = af + b$, donde G corresponde a la energía del espectro en [dB], a corresponde a su pendiente en [dB/Hz], f, a la frecuencia en [Hz], y b al desplazamiento de la recta de regresión en [dB], de forma que selecciona la correspondiente a la pendiente o slope [34]. Finalmente, se aplica la media sobre los arrays de slopes tanto para segmentos con voz como sin voz.

Loudness o Sonoridad

La sonoridad mide el nivel de volumen de la señal de voz y no sólo depende de la amplitud de una onda, sino también de su frecuencia y su espectro. Esto se debe principalmente a que nuestros oídos no son igual de sensibles a todas las frecuencias [35]. Es interesante calcular los percentiles para ver los distintos niveles de la señal. La característica correspondiente en `GeMAPs` es `loudness_sma3nz_percentile20.0`, que mide el percentil 20 de la sonoridad, proporcionando una idea del volumen más bajo de la señal. En este proyecto, sin embargo, se calculan los percentiles 10, 20, 50 y 90 tras transformar la señal en decibelios[36].

Frecuencia fundamental o pitch (F0)

`F0semitoneFrom27.5Hz_sma3nz_amean` es una característica de `GeMAPs` que en el primer experimento tuvo muy buen rendimiento. La frecuencia fundamental o pitch (F0) es esencial en el análisis de la prosodia y su promedio suele ser útil y representativo.

Para su cálculo existen varios métodos, aunque en este proyecto se opta por el uso de YIN, de la librería `librosa`[37]. Existe también en la misma librería el método `pYIN`, que usa un enfoque probabilístico para solventar posibles inconvenientes que puedan surgir, como por ejemplo que no sea capaz de calcular la frecuencia fundamental si el audio tiene un volumen demasiado bajo o contiene ruido en exceso. Gracias al dataset que se usa, no es necesario hacer uso de este método, pues la función `YIN` funciona correctamente para los audios normalizados y tiene un coste tanto en tiempo como en recursos considerablemente menor. Para las 15.800 grabaciones, extraer las características con el método `pYIN` tardaba cerca de 5 horas, mientras que con el método `YIN` eran aproximadamente 2. Evidentemente, la elección de un método u otro dependerá del proyecto, de si se ejecuta en batch como es este caso o en streaming y la urgencia que requiere el cálculo y el tipo de audio del que se dispone. En este proyecto, a pesar de trabajar en batch y no ser prioritaria la velocidad en los cálculos, es innecesario también

derrochar recursos y como que sea un código eficiente es uno de los requisitos, la elección tomada es la más razonable.

Tal y como en la característica de `GeMAPS`, en las de este proyecto también se calculan haciendo la transformación a los semitonos, algo útil ya que los semitonos se asemejan más a cómo el ser humano percibe las frecuencias y por tanto está más reconocido su uso de esta forma.
La transformación para obtener el número de semitonos desde 27.5Hz sigue la siguiente ecuación[38]:

$$n = 12 \cdot log_2(\frac{f0}{27{,}5})$$

Shimmer y Jitter

Los seres humanos usan propiedades como la prosodia o la fonética cuando reconocen voces y no sólo características espectrales como el nivel acústico de la señal o las frecuencias, siendo las primeras cualidades que dependen mucho del contexto social del hablante y de sus hábitos aprendidos e incluso de su dialecto. Es por eso que, tras el esfuerzo de incluir en el estudio de características del habla aquellas que dependen del medio temporal, algunas como el *shimmer* y el *jitter* han conseguido gran relevancia a la hora de conseguir buenos resultados en este ámbito[39].

Sin embargo, de cara a este proyecto se presenta un problema. El *shimmer* indica la variación de la amplitud y el *jitter*, la variación de la frecuencia fundamental, ambos ciclo a ciclo; por lo que no tiene sentido realizar un estudio de las mismas en el medio espectral sin considerar el dominio temporal[40]. Supone una complicación, pues la separación de la parte compleja y la parte real de la voz se realiza en el medio espectral y no en el temporal. Las características de esta sección son relevantes como para no prescindir de ellas, de forma que en pos de la investigación se plantea una posible solución: calcular la transformada de Fourier, separar las partes de la señal y luego de cada parte hacer una reconstrucción del audio mediante la herramienta `istft` (*Inverse short-time Fourier transform*). La reconstrucción de la señal por este medio no siempre es perfecta debido principalmente a cuestiones de redondeo en los cálculos y desde luego no se espera que el audio de cada parte sea igual que el audio de la totalidad, pero se considera interesante experimentar con estos aspectos.

Luego, una vez se tengan los audios de la parte real y la parte compleja de la señal, se calculan ambas características mediante la librería de Python `parselmouth`. Para utilizarla, se deben cargar los audios a través de ella, por lo que primero deben almacenarse en una ruta y pasarle esta ruta a la librería. Se debe a que el resultado de `istft` es realmente un array y no se entiende con la librería, pero al almacenarlo como WAV ya se puede leer correctamente para su uso. El script seguido para la implementacón que permite el cálculo de estas

características se puede encontrar en el github de David R. Feinberg [41].

5.2. Modelos

Para el desarrollo del trabajo se consideran dos modelos, de forma que se puedan contrastar resultados y tener una mejor visión sobre el desempeño real de las nuevas características. En las siguientes subsecciones se estudiarán brevemente los modelos Explainable Boosting Machine (EBM) y LightGBM, así como la configuración del entrenamiento en cada uno de ellos.

Sin embargo, antes de entrenar los modelos se ha de elegir qué parte del conjunto del dataset participará en el entrenamiento. Que el modelo aprenda los parámetros de una función de predicción partiendo de un conjunto, para luego probarla sobre los mismos datos genera un error de *overfitting*, pues el modelo que se limita a repetir las etiquetas de las muestras que acaba de ver con una puntuación perfecta luego no es capaz de predecir nada útil sobre datos aún no vistos. La solución es sencilla y común, basta con reservar una parte de los datos para la evaluación y entrenar el modelo con el conjunto restante. Con este fin se hace uso de la librería `sklearn`[42], que permite generar los conjuntos de entrenamiento y test de forma aleatoria manteniendo la distribución original de etiquetas, como se puede observar en el código A.15.

5.2.1. Explainable Boosting Machine (EBM)

Explainable Boosting Machine (EBM)[43] es un Modelo Aditivo Generalizado de gradiente cíclico basado en árboles de decisión con la peculiaridad de que es interpretable, ya que pertenece a la categoría llamada *glassbox* y fue diseñado con objetivos de inteligibilidad y explicabilidad en mente. Tiene una gran precisión, al igual que modelos no interpretables muy avanzados, como Random Forest y Boosted Trees, pero con el beneficio de la interpretación añadido. Los modelos de blackbox como podrían ser las Redes Generativas Antagónicas (GANs) o los Autoencoders Variacionales (VAEs) tienen muy buen rendimiento, pero en el caso de intentar ver y entender su funcionamiento, toda "explicación" proporcionada es aproximada y puede no ser tan evidente por qué tienen los resultados que tienen.

Para utilizar el modelo se inicializa una instancia de la clase *ExplainableBoostingClassifier* y se entrena con los conjuntos que se han definido para ese fin. Tras eso, se realizan las predicciones con el conjunto de test, y parar tener un poco más de sensibilidad sobre los resultados, se muestra la matriz de confusión del experimento.

Para hacer uso de su propiedad de ser interpretable, existe `InterpretML`, una librería que mediante diagramas proporciona un mayor detalle sobre las características que le han sido mayor utilidad en sus predicciones, algo imprescindible en este proyecto, pues si no sería complicado comprender qué características son útiles y cuáles no.

5.2.2. LightGBM

Con el objetivo de profundizar en el estudio de este proyecto, se plantea el uso de un modelo diferente que ejecute los mismos experimentos y aprenda de los mismos conjuntos de tal forma que se pueda observar si existe algún cambio en los resultados o en el rendimiento de las características usadas. LightGBM (Light Gradient Boost)[44] es un framework desarrollado por Microsoft de gradiente rápido que utiliza algoritmos de aprendizaje basados en árboles de decisión (GBDT). Está diseñado para ser altamente distribuido y de alto rendimiento, con la ventaja de que, a diferencia de otros algoritmos que crecen árboles por niveles (depth-wise), LightGBM crece árboles por hojas. Elige la hoja con la máxima pérdida delta para crecer y, aunque tiene un mayor riesgo de sobreajuste si el árbol crece demasiado, se puede evitar con límites y puede conducir a una reducción del error más rápida. Esto reduce tanto su coste en tiempo como en memoria de forma considerable[45].

Para el entrenamiento de este modelo se parte de los conjuntos que se han separado con sklearn. Además, hay que configurar el modelo mediante unos parámetros como el tipo de clasificación que se requiere o si ha de usar el Gradient Boosting Decision Tree u otro tipo de árbol de decisión.

Una vez entrenado el modelo se utiliza el conjunto de test para hacer las predicciones y poder evaluar el modelo. Las predicciones que hace LightGBM son probabilidades, por lo que se aplica una clasificación a binario bajo el condicionante de que sea 1 si es mayor o igual que 0.5.

Para interpretar los resultados de este modelo se hace uso de la librería `SHAP`. La librería genera un gráfico análogo al general que se consigue en el EBM gracias a InterpretML, de forma que se puede consultar un ranking sobre el rendimiento y la importancia de las características a la hora de clasificar las voces.

6

Resultados

Como se indicó en el apartado de objetivos, este documento abarca la parte técnica de todo el proyecto, tanto los aspectos que sirvieron para contextualizar y cerrar la parte teórica en la que se centró el trabajo de matemáticas "Detección de fraude de voz a través de modelos de la parte compleja de series de Fourier" [23] como para la continuación de la investigación con las nuevas características y el uso de un nuevo modelo.

Es por ello por lo que no se pueden dejar de mencionar los resultados que se presentaron en el primer documento, pues son parte del proyecto y exponerlos junto a los nuevos resultados permitirá una mejor comprensión de los mismos. Sin embargo, esta exposición se realizará de forma resumida y relativamente breve para no caer en repeticiones innecesarias, simplemente mencionando los puntos más importantes. Para un mayor detalle se deberá consultar el apartado análogo a este en el documento [23].

Como se ha podido ir viendo, la experimentación se ha realizado de forma incremental, donde cada vez se añadían nuevos elementos en base a los descubrimientos que se iban realizando, por lo que se estructura este capítulo en función de las distintas etapas de experimentos que tuvo el proyecto software. Cuando se habla de etapas, en este caso quiere decirse el período de tiempo que si bien está constituido por sprints o ciclos de trabajo con la culminación de reuniones y revisiones, su duración no era fija pues es una agregación que se está realizando en esta memoria y no tuvo peso en la metodología real. Al final, los sprints se organizaban de forma completamente flexible y en base a las necesidades que iban surgiendo, si algún aspecto de la investigación necesitaba más tiempo, se añadían

esas tareas para el siguiente sprint; y en este capítulo se están agrupando aquellos sprints que tenían un experimento o una línea de trabajo común para analizar los resultados obtenidos de los mismos.

6.1. Primera etapa

La primera etapa tuvo lugar tras los sprints centrados en la planificación del proyecto, documentación sobre el tema y preparación teórica. Una vez se había estudiado y comprendido el trasfondo matemático y se habían planteado los requisitos iniciales, se presentaba la ocasión de comenzar a desarrollar el primer experimento.

El primer experimento hacía uso únicamente del modelo EBM y se centraba en el estudio de las características estadísticas sobre la parte compleja de la señal de la voz. Además de esto, se planteó el llamado experimento 0, que consistía en observar la resolución del modelo con las características de `GeMAPs`, de forma que actuara como experimento de control. Los resultados que se obtuvieron resultaron bastante positivos.

En la tabla 6.1 se puede observar el desempeño de las características en función del puesto que obtuvieron en el ranking del top 15. Las únicas que tomaron relevancia suficiente fueron `ImagPartMean` e `ImagPartStd`, que incluso tomaron los primeros puestos al haber sido de gran ayuda al realizar las clasificaciones.

Característica	**Puesto experimento 1**
`ImagPartMean`	3
`ImagPartStd`	1
`ImagPartPercentile10`	-
`ImagPartPercentile20`	-
`ImagPartPercentile50`	-
`ImagPartPercentile90`	-

Tabla 6.1: Resultados de las características en el experimento 1 sobre el modelo EBM.

Por otro lado, como se puede ver en la tabla resumen (Tabla 6.2 del rendimiento en función de distintas métricas para el modelo EBM en el experimento base y en el primer experimento, con el primer experimento y las nuevas características, todas las métricas incrementaron en un par de décimas, algo bastante considerable una vez llegados esos niveles de rendimiento.

Experimento	Accuracy	VPP	TVP	TVN	F1 Score
Exp. 0	0.9158	0.9357	0.9488	0.8295	0.9226
Exp. 1	0.9484	0.9557	0.9737	0.8821	0.9646

Tabla 6.2: Métricas de rendimiento del experimento base y el primer experimento sobre el modelo EBM.

Se observa una mejora en el rendimiento con el experimento 1, pero que parece que únicamente recae sobre las características de la media y la desviación estándar, pues los percentiles no toman relevancia suficiente a la hora de predecir un valor.

6.2. Segunda etapa

Debido a los resultados interesantes que se habían obtenido con el primer experimento, se decidió incluir nuevos experimentos en los siguientes sprints. Concretamente, uno sobre la parte real de la señal y otro que agrupara tanto las características sobre la parte real como sobre la parte compleja.

A pesar de los escasos resultados positivos que se obtuvieron con los estadísticos de los percentiles, se volvieron a incluir en los requisitos de las características sobre la parte real, pues era más valioso confirmar que no aportan información relevante para la clasificación de las voces que el coste computacional o perjuicio que podía tener no extraer estas características y dejar la incertidumbre.

Nuevamente, los modelos entrenados tenían una gran eficacia y precisión, como se puede observar en la tabla 6.4, especialmente cuando se agrupaban las características tanto de la parte real como de la compleja en el segundo experimento.

En la tabla 6.3, se puede observar que el comportamiento de las características vuelve a ser análogo al que tuvo lugar en el primer experimento, pues los percentiles casi no tienen relevancia suficiente como para aparecer entre las quince principales y tanto la media como la desviación estándar consiguen estar entre las primeras ya sea de la parte real o de la parte imaginaria.

Característica	Puesto experimento 3	Puesto experimento 2
ImagPartMean	2	-
ImagPartStd	4	-
ImagPartPercentile10	-	-
ImagPartPercentile20	-	-
ImagPartPercentile50	-	-
ImagPartPercentile90	-	-
RealPartMean	1	1
RealPartStddev	6	3
RealPartPercentile10	-	-
RealPartPercentile20	-	-
RealPartPercentile50	-	13
RealPartPercentile90	-	-

Tabla 6.3: Resultados de las características en los experimentos 2 y 3 sobre el modelo EBM.

Se presenta el detalle de las métricas de rendimiento obtenidas para el modelo EBM en cada uno de los experimentos en la tabla 6.4. Recuérdese que el experimento 2 es el que agrupa todas las características de estadísticos, tanto de la parte real como de la parte compleja, mientras que el experimento 3 únicamente considera los estadísticos de la parte real de la voz.

Experimento	Accuracy	VPP	TVP	TVN	F1 Score
Exp. 2	0.9810	0.9839	0.9899	0.9576	0.9868
Exp. 3	0.9680	0.9777	0.9781	0.9416	0.9778

Tabla 6.4: Métricas de rendimiento del segundo experimento sobre el modelo EBM.

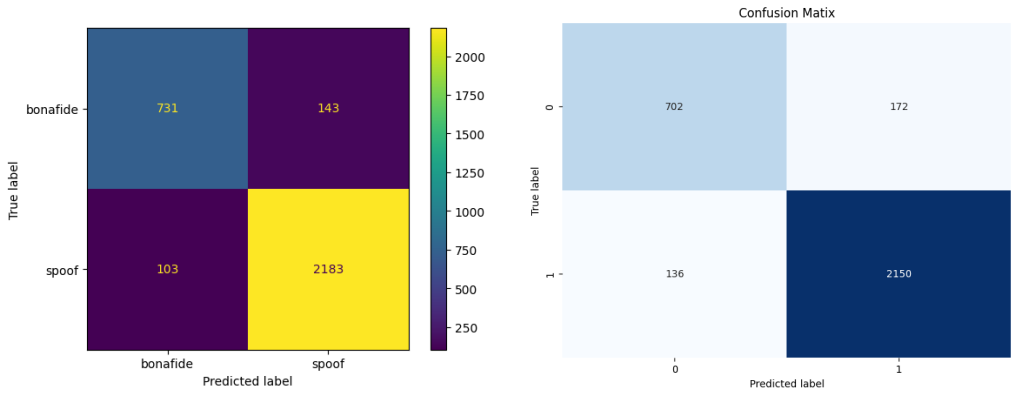
6.3. Tercera etapa

Tras haber obtenido los resultados observados en las etapas previas, se optó por profundizar la investigación. El camino seleccionado fue el cálculo de las características análogas a las más relevantes de GeMAPs junto con la inclusión de un nuevo modelo para entrenar. Es por esto que se repitieron los cuatro experimentos de las dos primeras etapas con el nuevo modelo, para así revisar y contrastar

la totalidad de los resultados, y también con el EBM para asegurar que se usan los mismos conjuntos de datos.

Se consideran dos nuevos experimentos: el experimento 4, que considera las características de `GeMAPs` y las calculadas en esta etapa, sin considerar los estadísticos; y experimento 5, que agrupa la totalidad de las características extraídas en el proyecto.

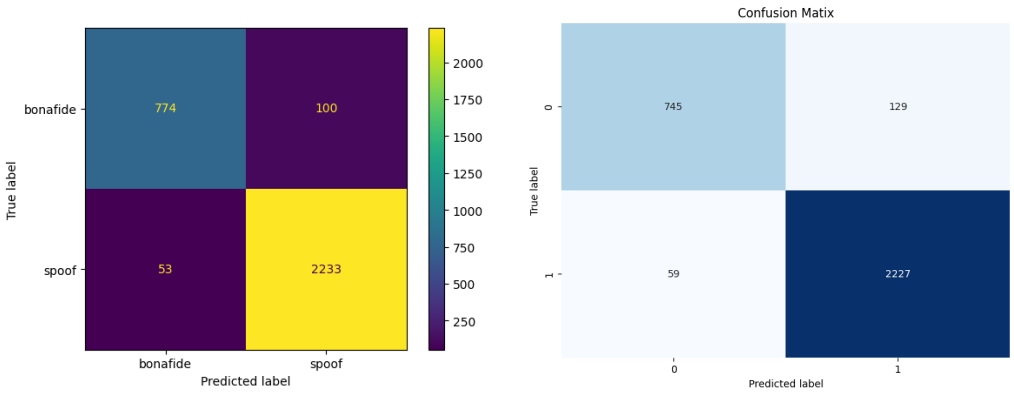
Retomando los experimentos para realizarlos con el modelo LightGBM, si se revisan las matrices de confusión del experimento base 6.1, se puede observar que la matriz obtenida del modelo LightGBM, 6.1(b), tiene más valores erróneos que la matriz 6.1(a) del modelo EBM, por lo que se ve que a pesar de estar entrenados con los mismos datos y predecir sobre los mismos, los resultados resultan ser levemente peores en el LightGBM.



(a) Modelo EBM. (b) Modelo LightGBM.

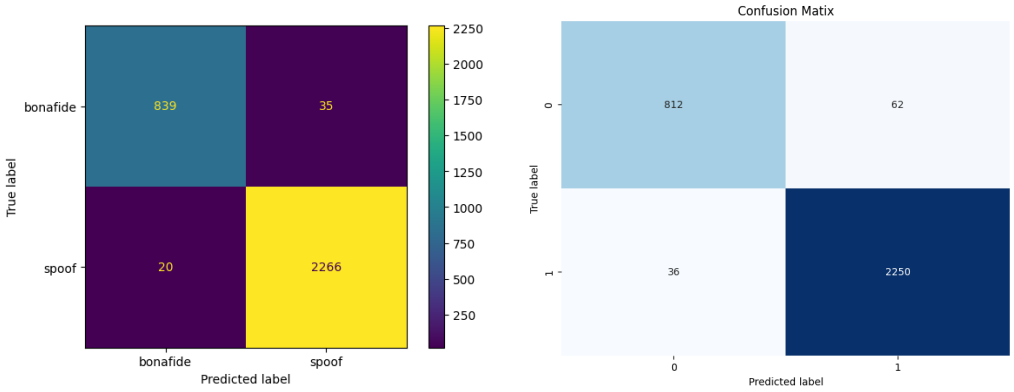
Figura 6.1: Matrices de confusión del experimento base.

En las matrices 6.2 del experimento 1, las correspondientes al experimento 2 6.3 y del tercer experimento 6.4, se observa el mismo comportamiento.



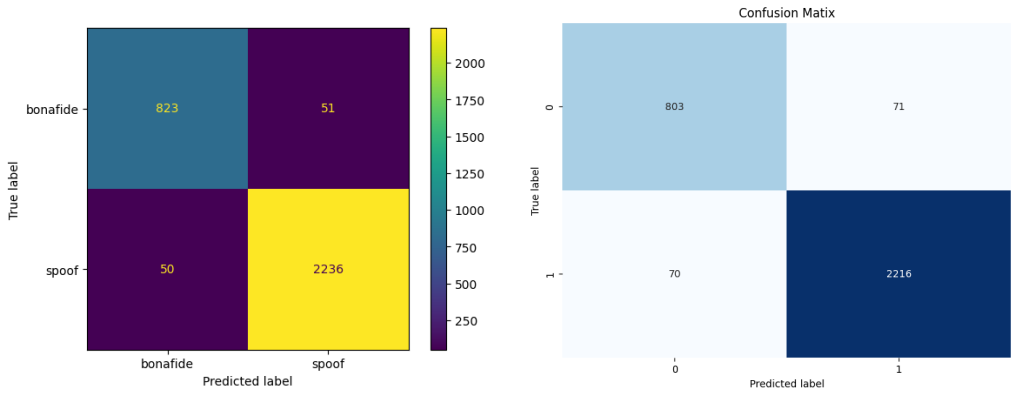
(a) Modelo EBM. (b) Modelo LightGBM.

Figura 6.2: Matrices de confusión del primer experimento.



(a) Modelo EBM. (b) Modelo LightGBM.

Figura 6.3: Matrices de confusión del segundo experimento.



(a) Modelo EBM. (b) Modelo LightGBM.

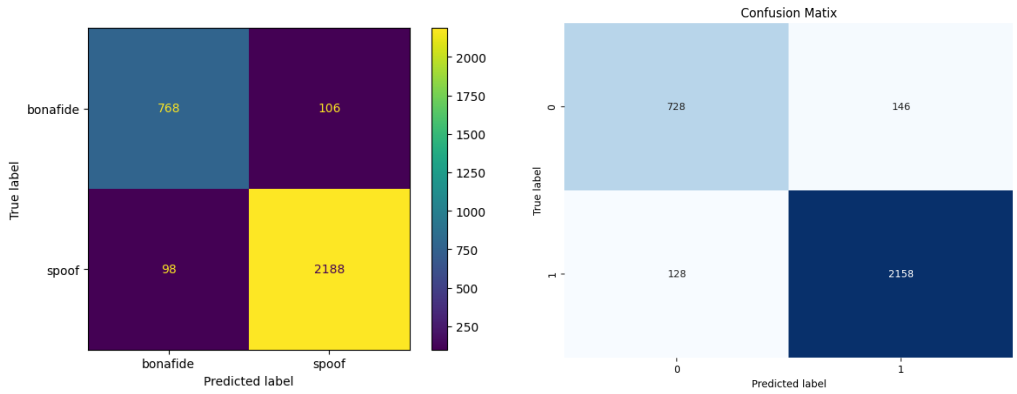
Figura 6.4: Matrices de confusión del tercer experimento.

Además, y en base a estos resultados, se pueden comprobar los índices de precisión que obtuvieron cada uno de los modelos en los distintos experimentos y que el LighGBM tiene generalmente unos valores más bajos, lo que indica un peor desempeño en la clasificación. Aunque, a pesar de que es peor que el EBM, sigue teniendo unos índices muy altos y el desempeño en general como modelo sigue siendo bastante bueno.

En el cuarto experimento, donde las características que servían para el entrenamiento del modelo eran:

- Las características extraídas con OpenSMILE sin separación de la señal.
- *Slope* en segmentos con voz y sin voz entre las frecuencias de 0-500Hz separando las partes de la señal.
- Percentiles 10, 20, 50, 90 de *Loudness* separando las partes de la señal.
- Semitonos de F0 desde la frecuencia 27.5Hz separando las partes de la señal.
- *Shimmer* y *jitter* locales separando las partes de la señal.

Se obtienen las matrices de confusión que se muestran en la Figura 6.5, en la que se encuentran más valores falsos positivos y falsos negativos que en los experimentos previos, asemejándose a unos valores entre el experimento base y el primer experimento con los estadísticos de la parte compleja.



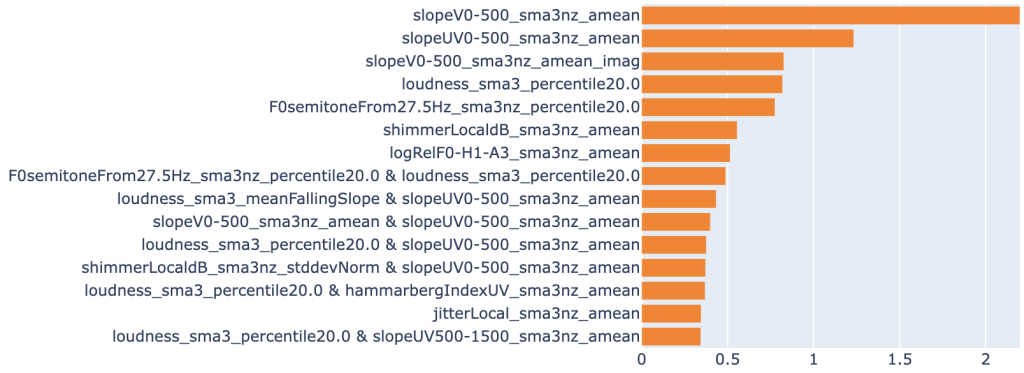
(a) Modelo EBM. (b) Modelo LightGBM.

Figura 6.5: Matrices de confusión del cuarto experimento.

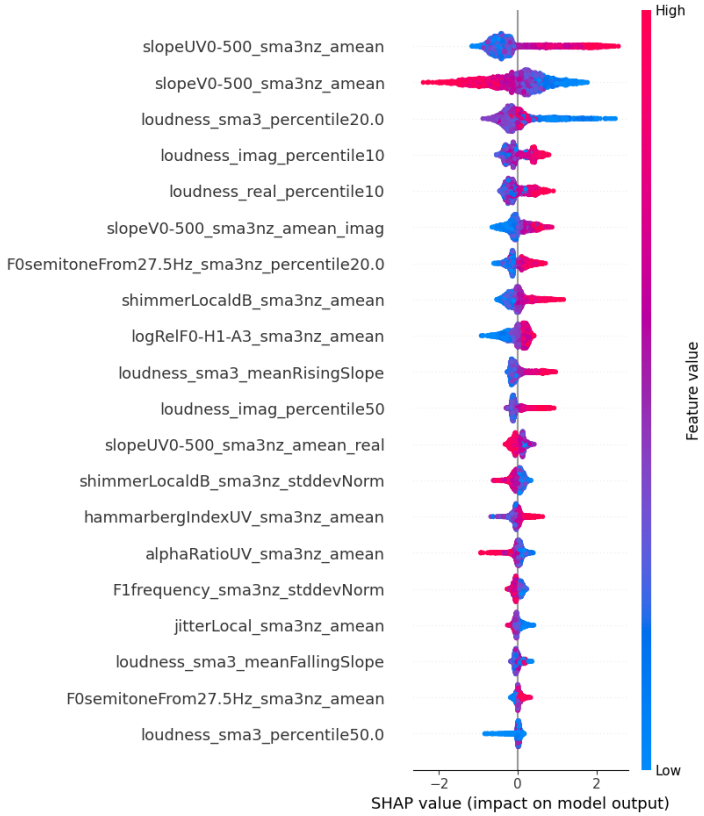
Al observar los diagramas donde se muestran los rankings de características más usadas, se puede notar que estos resultados se deben a que de los nuevos grupos de características casi no tienen características importantes. En el modelo EBM 6.6(a), la única que parece tener cierta relevancia es la referente al *slope* en segmentos con voz, `SlopeUV0-500_sma3nz_amean_imag`. Mientras que en el LightGBM 6.6(b) sí se puede encontrar una mayor presencia de ellas, con

- `loudness_imag_percentile10`
- `loudness_real_percentile10`
- `slopeV0-500_sma3nz_amean_imag`
- `loudness_imag_percentile50`

pero no parecen ser suficiente para mejorar de forma considerable el modelo.



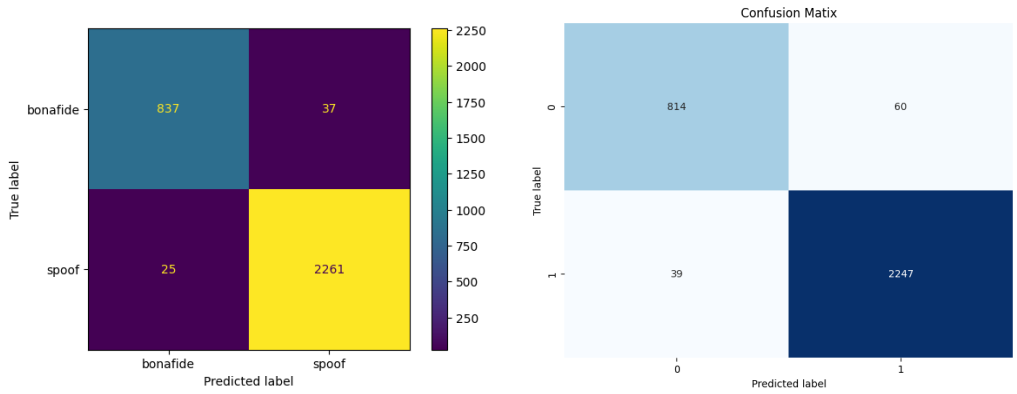
(a) Modelo EBM.



(b) Modelo LightGBM.

Figura 6.6: Diagramas con el ranking de características del cuarto experimento.

Por último, se realiza el experimento número 5, de forma que todas las características vistas en el proyecto se emplean para el entrenamiento de los dos modelos, tanto los estadísticos, como las de *slope*, *loudness*, *f0*, *shimmer* y *jitter*, junto con las extraídas de `GeMAPs`.



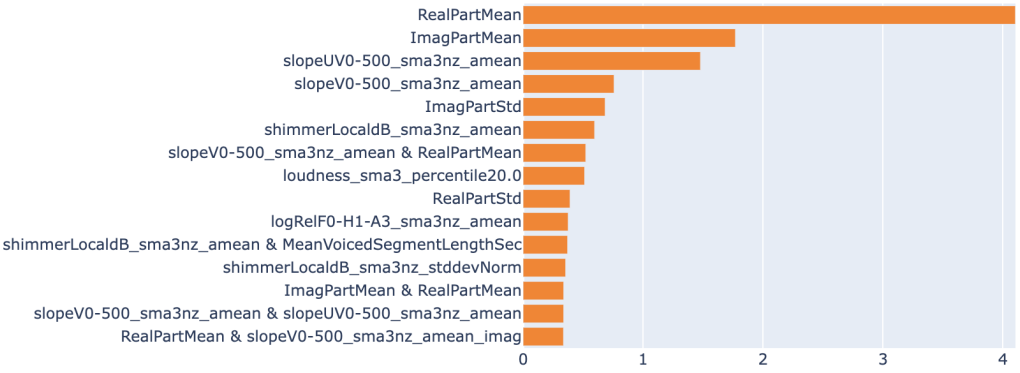
(a) Modelo EBM. (b) Modelo LightGBM.

Figura 6.7: Matrices de confusión del quinto experimento.

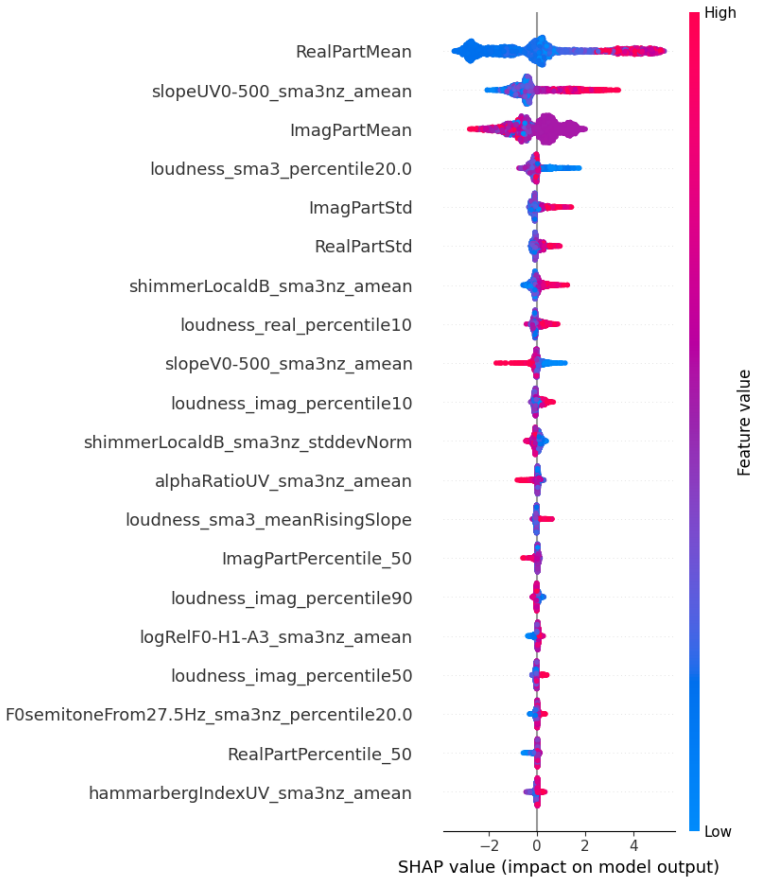
Tal y como se ve en las matrices de confusión 6.7, en el quinto experimento los resultados obtenidos se asemejan bastante a los del experimento 3, 6.4, sin reducir apenas el número de falsos positivos y falsos negativos.

Parece algo lógico considerando que, al observar el desempeño de las características referentes al *slope*, *loudness*, *f0*, *shimmer* y *jitter* del experimento 4, no estaban muy presentes en los rankins 6.6 y por tanto, aunque se añadan sobre los datos de entrenamiento del experimento 3, no mejora demasiado.

Así lo confirman los diagramas de los rankings de las características para este quinto experimento que se pueden ver en las figuras 6.8. El orden obtenido se asemeja enormemente a los obtenidos en el experimento 3, visibles en la figura 6.9; con la diferencia de que en el EBM aparece una combinación con la característica `SlopeV0-500_sma3nz_amean_imag` y en el LightGBM aparecen varios de los percentiles sobre *loudness*, pero siempre en puesto bajos.

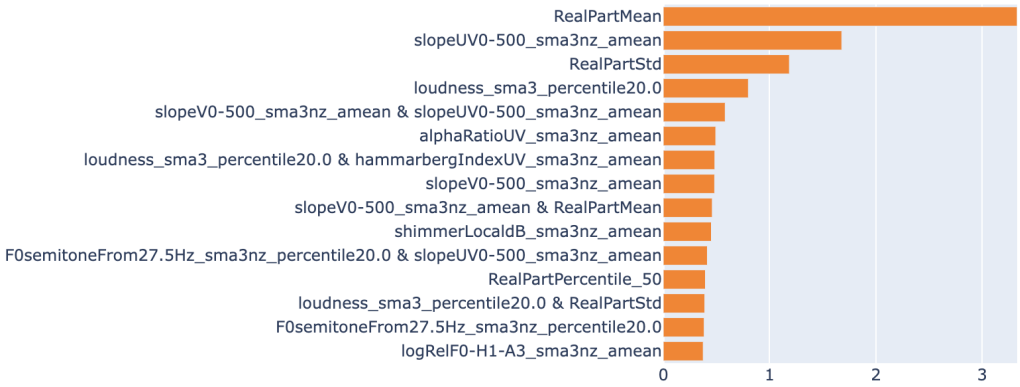


(a) Modelo EBM.

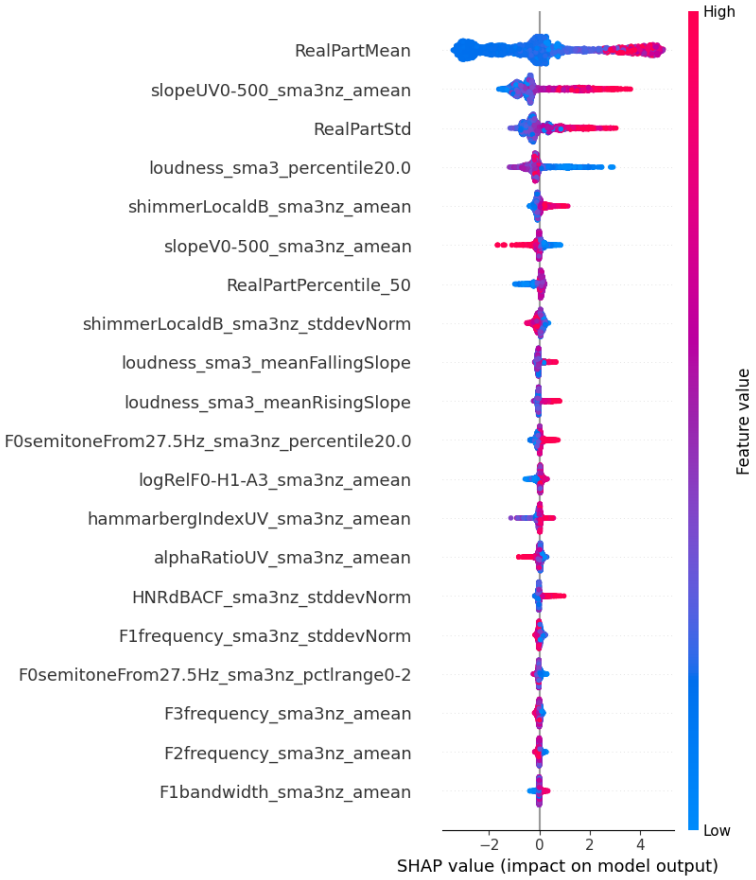


(b) Modelo LightGBM.

Figura 6.8: Diagramas con el ranking de características del quinto experimento.



(a) Modelo EBM.



(b) Modelo LightGBM.

Figura 6.9: Diagramas con el ranking de características del tercer experimento.

Experimento	Modelo	Acccuracy	VPP	TVP	TVN	F1 Score
Experimento 0	LightGBM	90.25 %	92.59 %	94.05 %	80.32 %	**93.31 %**
	EBM	**91.58 %**	**93.5 %**	**94.88 %**	**82.95 %**	92.26 %
Experimento 1	LightGBM	94.05 %	94.52 %	**97.41 %**	85.24 %	95.95 %
	EBM	**94.84 %**	**95.57 %**	97.37 %	**88.21 %**	**96.46 %**
Experimento 2	LightGBM	96.89 %	97.31 %	98.42 %	92.90 %	97.86 %
	EBM	**98.10 %**	**98.39 %**	**98.99 %**	**95.76 %**	**98.68 %**
Experimento 3	LightGBM	95.53 %	96.89 %	96.93 %	91.87 %	96.91 %
	EBM	**96.80 %**	**97.77 %**	**97.81 %**	**94.16 %**	**97.78 %**
Experimento 4	LightGBM	91.32 %	93.66 %	94.40 %	83.29 %	94.03 %
	EBM	**93.54 %**	**95.71 %**	**95.38 %**	**88.60 %**	**95.54 %**
Experimento 5	LightGBM	96.86 %	97.39 %	98.29 %	93.14 %	97.84 %
	EBM	**98.04 %**	**98.91 %**	**98.39 %**	**97.10 %**	**98.65 %**

Tabla 6.5: Métricas de rendimiento de los experimentos sobre ambos modelos.

Se puede comprobar que existe una mejora con las características calculadas en esta etapa (por ejemplo, en el experimento 4 un 91.32 % en Accuracy frente al 90.25 % del experimento base), pero es una mejora mucho menor que la que se había logrado al incluir los estadísticos de la parte real o la parte compleja (94.05 % o 95.53 %, respectivamente). Incluso, una vez se agregan todas las características al experimento, no se aprecia una mejoría real entre el experimento 5, con todas, o el experimento 2 con los estadísticos tanto de la parte real como la parte imaginaria de la señal.

Finalmente, se revisa de forma global el desempeño de las características en la siguiente tabla para cada uno de los experimentos realizados. En la tabla 6.6 se pueden observar aquellas características que han aparecido en el ranking de alguno de los modelos. Nótese que NA se atribuye a *No Aparece*, significando las características que se han usado en el experimento, pero no han tenido relevancia suficiente; mientras que cuando una característica presenta "-" indica que no participó en dicho/s experimento/s y no fue usada para el entrenamiento de los modelos. Si alguna característica aparecía en los rankings en combinación de otra característica, se indica el puesto junto con un asterisco (*).

Característica	Experimento 1		Experimento 2		Experimento 3		Experimento 4		Experimento 5	
	EBM	**LGBM**	**EBM**	**LGBM**	**EBM**	**LGBM**	**EBM**	**LGBM**	**EBM**	**LGBM**
ImagPartMean	**3, 14*, 15***	**3**	**2, 12***	**3**		-			**2, 13***	**3**
ImagPartStd	**1, 14***	**1**	**4**	**4**		-			**5**	**5**
ImagPartPercentile10	NA	NA	NA	NA		-			NA	NA
ImagPartPercentile20	NA	NA	NA	NA		-			NA	NA
ImagPartPercentile50	NA	**16**	NA	**16**		-			NA	14
ImagPartPercentile90	NA	NA	NA	NA		-			NA	NA
RealPartMean	-		**1, 5***	**1**	**1**	**1**	-		**1, 7*, 13*, 15***	**1**
RealPartStd	-		**6, 12***	**5**	**3, 13***	**3**	-		**9**	**6**
RealPartPercentile10	-		NA	NA	NA	NA	-		NA	NA
RealPartPercentile20	-		NA	NA	NA	NA	-		NA	NA
RealPartPercentile50	-		NA	**15**	**12**	**7**	-		NA	**19**
RealPartPercentile90	-		NA	NA	NA	NA	-		NA	NA
SlopeUV0-500_sma3nz_amean_imag			-				NA	NA	NA	NA
SlopeUV0-500_sma3nz_amean_real			-				NA	**15**	NA	NA
SlopeV0-500_sma3nz_amean_imag			-				**3**	**6**	**15***	NA
SlopeV0-500_sma3nz_amean_real			-				NA	NA	NA	NA
Loudness_imag_percentile10			-				NA	**4**	NA	**10**
Loudness_imag_percentile20			-				NA	NA	NA	NA
Loudness_imag_percentile50			-				NA	**14**	NA	**17**
Loudness_imag_percentile90			-				NA	NA	NA	**15**
Loudness_real_percentile10			-				NA	**5**	NA	**8**
Loudness_real_percentile20			-				NA	NA	NA	NA
Loudness_real_percentile50			-				NA	NA	NA	NA
Loudness_real_percentile90			-				NA	NA	NA	NA
ShimmerLocaldB_imag			-				NA	NA	NA	NA
ShimmerLocaldB_real			-				NA	NA	NA	NA
JitterLocal_imag			-				NA	NA	NA	NA
JitterLocal_real			-				NA	NA	NA	NA
F0semitonefrom27.5_imag			-				NA	NA	NA	NA
F0semitonefrom27.5_real			-				NA	NA	NA	NA

Tabla 6.6: Resultados de las características en cada uno de los experimentos sobre ambos modelos.

Es interesante resaltar cómo el modelo LightGBM tiene peores resultados en la clasificación de las voces, pero hace un mayor uso de las características calculadas en el proyecto, especialmente de aquellas que el EBM siquiera considera entre las 15 primeras. En cualquier caso, no es una comparativa que deba hacerse directamente ni usarse para esclarecer conclusiones en base a ello, pues el modelo EBM hace combinaciones de características y las considera como un nuevo elemento que muestra en su interpretación, mientras que el modelo LightGBM, al ser de tipo blackbox tiene ciertas limitaciones en su interpretación, como por ejemplo, que no muestra esos pares o relaciones que posiblemente genere internamente para realizar las predicciones y únicamente es capaz de mostrar la influencia de las características individualmente.

Se podría pensar que si, en el cuarto experimento, el EBM no tiene la característica `Loudness_imag_percentile10` entre aquellas que le han sido de mayor utilidad, pero LightGBM la tiene en cuarto puesto, como ha obtenido peores resultados, quiere decir que la característica en concreto y las características en general que ha tomado LightGBM suponen una peor elección. Esto no es una conclusión acertada de extraer, pues tal y como se indican, por la propia naturaleza distinta de los modelos, existen factores que no se pueden considerar en el estudio que aquí se realiza y únicamente sirve para extraer ciertas conclusiones a rasgos generales sobre el conjunto de características estudiado.

7

Conclusiones

Se puede considerar que este trabajo tiene dos caras, pues por un lado se está tratando la investigación de un tema muy importante que da solución al problema real de los deepfakes de voz, y por otro se debe aplicar correctamente el proceso software que ha albergado la investigación. Ambos presentan una parte muy importante del trabajo, por lo que de cara a las conclusiones no se puede dejar de mencionar ninguno de los aspectos.

Comenzando por la investigación del tema, aunque en este documento no se habla de los fundamentos teóricos tal y como se hacía en el otro trabajo fin de grado [23], se presentan en el capítulo 5 todos los aspectos técnicos que han resultado fundamentales para la solución del problema planteado.

Con la sección 5.1 se estudiaba la extracción de las características vistas en el otro documento junto con las nuevas. El cálculo de los estadísticos no tuvo una gran complejidad, pues se realiza mediante operaciones sencillas donde el único aspecto que hubo que considerar fue el orden de cálculo para filas y columnas; una decisión pequeña, pero que requería de comprender en profundidad el funcionamiento de la función que aplicaba la transformada de Fourier de tiempo reducido para saber cómo estaba estructurado el espectrograma de salida.

Cuatro nuevos conjuntos de características acompañaban a los estadísticos previamente calculados. Tal y como se ha mencionado en las secciones correspondientes, las características que se deseaban calcular tienen gran relevancia en el campo de tratamiento y análisis de voz y existen funciones en Python que permite su cálculo fácilmente. Estas funciones evidentemente están encapsuladas y el funcionamiento interno es transparente para el usuario, de forma que sólo

permiten saber lo que necesitan en el input y cuál será el output. Existe gran cantidad de información y documentación referente a su uso, pero prácticamente nada sobre su funcionamiento interno. Esto supone un reto cuando por las necesidades del proyecto no se puede suplir el input requerido por estas funciones y se han de implementar nuevamente. En el caso de las características relacionadas con *loudness* 5.1.2 y FO 5.1.2 bastó con profundizar un nivel, pues existen funciones que calculan parte de estas como puede ser el método `YIN`. En los casos de las características en los grupos *slope* 5.1.2 y *shimmer* y *jitter* 5.1.2, la cuestión no era tan evidente y hubo que recurrir a documentación teórica sobre estas propiedades de los sonidos para comprender en detalle qué representaban, cómo se calculaban y en base a eso implementar las nuevas funciones. Se pudo observar que mientras que el *slope* es una característica espectral y bastó con comprender sus fundamentos, para el *shimmer* y *jitter* se encontró la problemática de que no son características espectrales, pero había que hacer la transformación al medio espectral, por lo que se optó por la transformación posterior al audio y el uso de las librerías habituales. Con todo esto se completó el objetivo 2.1.

En ese mismo capítulo, en la sección 5.2 se presentaron los modelos que servirían para el estudio de las características. Se eligieron dos modelos que habían demostrado en estudios y competiciones un buen desempeño en tareas de clasificación con gran eficiencia en recursos temporales. El proceso para configurar el EBM y el LightGBM, los modelos seleccionados, junto con el posterior entrenamiento y testeo y cómo se extraían las características es lo que completa la sección, de forma que se validaba el objetivo 2.2-1.

En último lugar del desarrollo, se revisaban todos los resultados que había obtenido el proyecto. Dividiendo la línea temporal por etapas, se veía en cada una de ellas el rendimiento del los modelos a nivel global en los experimentos que se habían realizado. Además, se presentaban las características junto con la relevancia que habían tenido según habían indicado los modelos.

Gracias a lo que se observaba en la tabla 6.5, se concluía que, aunque los resultados del LightGBM eran ligeramente peores, no existió gran diferencia entre los resultados de un modelo y otro, algo que confirmaba la robustez y coherencia de los mismos. De cara a la investigación, hubiera generado incertidumbre si los resultados conseguidos en un mismo experimento para cada modelo hubieran tenido grandes diferencias, como por ejemplo, si a uno le hubiera sido de gran importancia la característica `ImagPartMean` y para el otro siquiera apareciera entre las más relevantes. Realmente las discrepancias, aunque existentes, eran ínfimas pues se encontraban principalmente en características menos relevantes, y que tal y como se explicó al final de esa sección, podía deberse a otros aspectos no relevantes para el proyecto.

Tras obtener y contrastar los resultados obtenidos con los experimentos, la conclusión más perceptible es la mejoría en cualquier caso al usar las características de partes separadas de la señal de la voz sobre aquel modelo base que empleaba

características extraídas sobre la totalidad.

Parece que las características calculadas en la segunda etapa, que se corresponden con características que normalmente tienen bastante relevancia en el análisis de audio, no han sido capaces de generar unos resultados especialmente destacables como sí sucedía con la media y la desviación estándar de la parte imaginaria y la parte real de las voces. Este resultado podría deberse a una pérdida de la información al hacer la trasformación al espectrograma y su posterior separación en partes compleja y real, tal y como se vio que sucedía incluso a un nivel teórico al calcular las características de *Shimmer* y *Jitter*. Aunque sean pérdidas de redondeo pequeñas, tal vez para este tipo de características supongan una gran diferencia al difuminar los límites entre las voces reales y falsas con dichos redondeos. Algo que cualquier investigador podrá confirmar es la tristeza que se encuentra al no conseguir los resultados esperados y, aunque los resultados globales son realmente positivos, en un mayor detalle, apena el observar que una parte tan extensa de todo lo investigado no da los frutos que se buscaban. En cualquier caso, resultados positivos o negativos, siguen siendo resultados y permite comprender más en profundidad el problema que aquí se trataba y se buscaba solucionar. Dicho todo esto, con este contraste de resultados quedaba completado el subobjetivo 2.2-2 y con él se cierra además el objetivo 2.2.

Una parte muy importante de este trabajo era plasmar todo el proceso software que se ha llevado a cabo en este proyecto. En el capítulo 4 se presentaba la metodología del proyecto. Desde un primer momento se optó por una metodología ágil, pues al comienzo no estaban completamente definidos los requisitos que iban a surgir posteriormente, por lo que completar el objetivo 2.3-1 fue bastante sencillo. Basta observar que aquella primera solución que se buscaba para conseguir la mejora en la clasificación de las voces, que se correspondía con la etapa 1 expuesta en la sección 6.1, se centraba en analizar únicamente la parte compleja de la señal de la voz. Una vez decidida la metodología que daría soporte al proyecto, se estableció la duración de los sprints y cómo se iba a trabajar tanto internamente como junto a los tutores.

La especificación de los requisitos ha estado presente a lo largo de gran parte del proyecto. Si bien es cierto que no existió necesidad de modificar aquellos que se iban estableciendo, a lo largo de los sprints han surgido nuevos intereses o líneas de investigación que se podrían seguir sin exceder los costes del proyecto, por lo que se fueron añadiendo nuevos requisitos para suplir dichas novedades. Puede considerarse la tercera etapa como aquella donde los requisitos se mantuvieron más estables, aunque el tema aquí presente es de gran interés y aunque se cierren líneas de investigación, siempre va a existir la oportunidad y las ganas de avanzar por nuevas o profundizar en las mismas. Sin embargo, fue en esta última etapa en la que se consideró que era apropiado dar por completa la especificación de requisitos para el proyecto que se presenta, completando entonces el objetivo 2.3-3.

Por último, se concluye que el uso de la metodología ágil ha sido un aspecto positivo a nivel global, pues ha facilitado la flexibilidad en la investigación. Aunque si bien es cierto, no tiene por qué ser algo que se mantenga para un futuro. En trabajos futuros, posiblemente se decida tomar una línea de trabajo bastante concreta y es muy posible que los requisitos se encuentren definidos desde el comienzo de la línea de investigación. Tal y como se ha mencionado, en la tercera etapa del proyecto 6.3 la lista de requisitos se encontraba prácticamente cerrada desde el comienzo y, aunque posteriormente mientras se experimenta pudieran surgir ciertos aspectos inesperados, lo cierto es que casi no hubo que modificar estos nuevos requisitos. De igual forma, en un proyecto nuevo que tome de base este, se podría optar por una metodología en cascada mucho más cerrada a posibles cambios pero con una planificación temporal más estable. Con todo lo aquí expuesto se valida el proceso que ha tenido lugar en el proyecto, finalmente dando por completado el objetivo 2.3.

8

Trabajos futuros

Existen numerosas líneas de investigación que aún quedan por explorar y que tienen gran potencial e interés. Sería posible elegir un dataset de voces diferente y sobre ellos calcular las mismas características que aquí se han tratado y sería algo que seguramente aportaría, no simplemente por hacer una modificación aparentemente simple, sino porque esto conlleva unas consecuencias mucho más profundas. En la sección 5.1.2 se mencionaba brevemente que aspectos fundamentales del reconocimiento de voz dependen de factores del hablante como su dialecto o el entorno social en el que se haya criado, es por ello que haber realizado este proyecto con un dataset de voces en español, puede haber resultado de forma diferente a como hubiera resultado con un dataset en inglés o con uno de un dialecto del alemán, por ejemplo. Por eso mismo, un trabajo futuro que implique este "pequeño" cambio, podría suponer una gran diferencia.

Proyectos futuros evidentes también resultarían de estudiar nuevas características o entrenar nuevos modelos. De hecho, es de este propósito de donde surgió la tercera etapa, de la ambición de profundizar en la investigación del tema con estos nuevos aspectos. Sin embargo, aunque en este proyecto, se haya tomado la decisión de extraer las características vistas y entrenar el EBM y el LightGBM, no quiere decir que hayan sido las únicas que eran posible tomar. Hoy en día existen una gran diversidad de modelos discriminativos que funcionan para este campo y que se pueden emplear para este estudio en concreto. Tal y como se veía en la sección 6.3, existían ciertas diferencias entre los resultados obtenidos con los dos modelos, a pesar de tener ambos unos índices de rendimiento muy alto y ser ambos modelos basados en árboles de decisión; es imposible evitar la curiosidad que genera el pensar en los resultados que podrían obtenerse del uso

de modelos mucho más diferentes entre sí.

Por último añadir que, con este proyecto no sólo se presentan trabajos futuros enfocados al campo de la investigación, también se presentan trabajos más prácticos y tangibles. Dado el buen resultado obtenido en este proyecto, se considera que es lo suficientemente relevante como para querer que esté disponible a una cantidad mayor de usuarios, por lo que sería de interés generar un producto con una interfaz mucho más accesible. Quizá enfocándolo al desarrollo de una aplicación de detección de deepfakes y donde se incluyeran entre las características, las aquí calculadas. Dar soporte a este proyecto con un entorno que pudiera usar cualquiera resultaría en un trabajo futuro satisfactorio.

Bibliografía

[1] L. Verdoliva, "Media forensics and deepfakes: An overview," *IEEE Journal on Selected Topics in Signal Processing*, vol. 14, 2020.

[2] C. Bregler, M. Covell, and M. Slaney, "Video rewrite: Driving visual speech with audio," in *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1997*, 1997.

[3] J. J. Beard, "Clones, bones and twilight zones: Protecting the digital persona of the quick, the dead and the imaginary," *Journal of the Copyright Society of the U.S.A.*, vol. 49, 2001.

[4] J. Meese, B. Nansen, T. Kohn, M. Arnold, and M. Gibbs, "Posthumous personhood and the affordances of digital media," *Mortality*, vol. 20, 2015.

[5] Z. Yang, "Deepfakes of your dead loved ones are a booming chinese business," 5 2024. [Online]. Available: https://www.technologyreview.com/2024/05/07/1092116/deepfakes-dead-chinese-business-grief/#:~:text=Avatars%20of%20the%20dead%20are,that%20can%20move%20and%20speak.

[6] T. T. Nguyen, Q. V. H. Nguyen, D. T. Nguyen, D. T. Nguyen, T. Huynh-The, S. Nahavandi, T. T. Nguyen, Q. V. Pham, and C. M. Nguyen, "Deep learning for deepfakes creation and detection: A survey," *Computer Vision and Image Understanding*, vol. 223, 2022.

[7] J. Vincent, "New ai deepfake app creates nude images of women in seconds," 6 2019. [Online]. Available: https://www.theverge.com/2019/6/27/18760896/deepfake-nude-ai-app-women-deepnude-non-consensual-pornography

[8] S. Agarwal, H. Farid, Y. Gu, M. He, K. Nagano, and H. Li, "Protecting world leaders against deep fakes," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, vol. 2019-June, 2019.

[9] B. Paris and J. Donovan, "Deepfakes and cheap fakes," *Data and Society*, 2019.

[10] L. A. Peralta, "Los usos criminales de los 'deepfakes' se disparan: estafas, pornografía y suplantación de identidad," *EL PAÍS*, 10 2024. [Online]. Available: https://elpais.com/proyecto-tendencias/2024-10-02/los-usos-criminales-de-los-deepfakes-se-disparan-estafas-pornografia-y-suplantacion-de-identidad.html

[11] Hughes, "A bill to be entitled an act relating to the creation of a criminal offense for fabricating a deceptive video with intent to influence the outcome of an election." 3 2019. [Online]. Available: https://capitol.texas.gov/tlodocs/86R/billtext/html/SB00751S.htm

[12] A. Birrer and N. Just, "What we know and don't know about deepfakes: An investigation into the state of the research and regulatory landscape," *New Media & Society*, 5 2024.

[13] Y. Wang, R. J. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, and Q. Le, "Tacotron: Towards end-to-end speech synthesis," in *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, vol. 2017-August, 2017.

[14] S. Y. Lim, D. K. Chae, and S. C. Lee, "Detecting deepfake voice using explainable deep learning techniques," *Applied Sciences (Switzerland)*, vol. 12, 2022.

[15] T. Chen, A. Kumar, P. Nagarsheth, G. Sivaraman, and E. Khoury, "Generalization of audio deepfake detection," 2020.

[16] M. McUba, A. Singh, R. A. Ikuesan, and H. Venter, "The effect of deep learning methods on deepfake audio detection for digital investigation," in *Procedia Computer Science*, vol. 219, 2023.

[17] P. A. T. Flórez, R. Manrique, and B. P. Nunes, "Habla: A dataset of latin american spanish accents for voice anti-spoofing," vol. 2023-August, 2023.

[18] J. Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," vol. 2017-October, 2017.

[19] V. Popov, I. Vovk, V. Gogoryan, T. Sadekova, M. Kudinov, and J. Wei, "Diffusion-based voice conversion with fast maximum likelihood sampling scheme," 2022.

[20] H. Kameoka, T. Kaneko, K. Tanaka, and N. Hojo, "Stargan-vc: Non-parallel many-to-many voice conversion using star generative adversarial networks," 2018.

[21] Microsoft, "Azure text-to-speech," 2023, last accessed 6 July 2024. [Online]. Available: https://azure.microsoft.com/es-es/products/ai-services/text-to-speech

[22] F. Eyben, M. Wöllmer, and B. Schuller, "opensmile - the munich versatile and fast open-source audio feature extractor," *Proc. ACM Multimedia*, 2010.

[23] S. G. Terroba, "Detección de fraude de voz a través de modelos de la parte compleja de series de fourier," 2024.

[24] F. Eyben, K. R. Scherer, B. W. Schuller, J. Sundberg, E. Andre, C. Busso, L. Y. Devillers, J. Epps, P. Laukka, S. S. Narayanan, and K. P. Truong, "The geneva minimalistic acoustic parameter set (gemaps) for voice research and affective computing," *IEEE Transactions on Affective Computing*, vol. 7, pp. 190–202, 4 2016.

[25] M. Barandas, D. Folgado, L. Fernandes, S. Santos, M. Abreu, P. Bota, H. Liu, T. Schultz, and H. Gamboa, "Tsfel: Time series feature extraction library," *SoftwareX*, vol. 11, p. 100456, 2020.

[26] V. P. García, "Contrastes de normalidad," 2021, last accessed 6 July 2024.

[27] M. Molina, "Análisis de normalidad," 2022, last accessed 6 July 2024. [Online]. Available: https://anestesiar.org/2022/analisis-de-normalidad-una-imagen-vale-mas-que-mil-palabras/

[28] S. S. SHAPIRO and M. B. WILK, "An analysis of variance test for normality (complete samples)," *Biometrika*, vol. 52, pp. 591–611, 12 1965.

[29] F. J. Massey, "The kolmogorov-smirnov test for goodness of fit," *Journal of the American Statistical Association*, vol. 46, p. 68, 3 1951.

[30] N. Developers, "numpy.nanpercentile," 2024, last accessed 6 July 2024. [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.nanpercentile.html#numpy-nanpercentile

[31] N. Developers., "numpy.mean," 2024, last accessed 6 July 2024. [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.mean.html#numpy-mean

[32] ——, "numpy.std," 2024, last accessed 6 July 2024. [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.std.html#numpy-std

[33] P. University, "Digital signal processing with applications laboratory - 9," 2010.

[34] L. Tamarit, M. Goudbeek, and K. Scherer, "Spectral slope measurements in emotionally expressive speech," *Proceedings of speech analysis and processing for knowledge discovery*, 2008.

[35] J. Wolfe, "Loudness and spectra," 2015. [Online]. Available: https://www.phys.unsw.edu.au/jw/loudness.html

[36] librosa development team, "librosa.amplitude_to_db," 2015. [Online]. Available: https://librosa.org/doc/main/generated/librosa.amplitude_to_db.html

[37] ——, "Librosa.yin," *Librosa*, 2015. [Online]. Available: https://librosa.org/doc/main/generated/librosa.yin.html

[38] J. Wolfe, "Note names, midi numbers and frequencies," *The University New South Wales*, 2005. [Online]. Available: https://newt.phys.unsw.edu.au/jw/notes.html

[39] M. Farrús, J. Hernando, and P. Ejarque, "Jitter and shimmer measurements for speaker recognition," in *International Speech Communication Association - 8th Annual Conference of the International Speech Communication Association, Interspeech 2007*, vol. 2, 2007.

[40] M. Brockmann-Bauser, J. E. Bohlender, and D. D. Mehta, "Acoustic perturbation measures improve with increasing vocal intensity in individuals with and without voice disorders," *Journal of Voice*, vol. 32, 2018.

[41] D. R. Feinberg, "Parselmouth praat scripts in python," Jan 2022. [Online]. Available: osf.io/6dwr3

[42] scikit-learn developers, "Train_test_split," 2010. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html#sklearn.model_selection.train_test_split

[43] Y. Lou, R. Caruana, J. Gehrke, and G. Hooker, "Accurate intelligible models with pairwise interactions," vol. Part F128815, 2013.

[44] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T. Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems*, vol. 2017-December, 2017.

[45] A. V. Bustamante, "Entendiendo a fondo el lightgbm," *Medium*, 1 2024.

Apéndices

A

Código de ejecución del experimento

En este apéndice se incluyen los códigos principales de los que se ha hecho uso a lo largo del proyecto.
Se pueden agrupar en tres secciones principales: Configuración del entorno, preparativos y tratamiento de los datos; estudio de características y entrenamiento y uso de los modelos.

El Google Colab está accesible mediante el siguiente link: `https://colab.research.google.com/drive/1p-Vh78H9Ka0YWIBgRbP6ORJGwuke3m6d?usp=sharing`

A.1. Configuración, preparativos y tratamiento de datos

A.1.1. Configuración del entorno global

Esta primera parte se encarga de configurar todo el entorno de ejecución. Como se trabaja mediante un colab o notebook se debe indicar el directorio del que se hará uso, que en este caso es uno albergado en Google Drive. Como Colab es un tipo de jupyter notebook, además se configura el tipo de salida que tengan las celdas para que se muestren perfectamente sin que se reduzcan y tengan un scroll. Finalmente, se importan las librerías que se usarán a lo largo del código.

Este primer bloque de código es esencial que se ejecute para el funcionamiento del resto. El resto de bloques no tienen por qué ser necesarios para la ejecución de otros.

Código A.1: Configuración del entorno global.

```
# Load the Drive helper and mount
from google.colab import drive

# This will prompt for authorization.
drive.mount('/content/drive')

%cd /content/drive/MyDrive/HABLA

%%javascript
IPython.OutputArea.prototype._should_scroll = function(lines) {
    return false;
}

%pip install torch==2.5.0 descartes opensmile audb praat-
   parselmouth

import os

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import matplotlib.image as mpimg
from matplotlib.offsetbox import AnnotationBbox, OffsetImage
import scipy.stats as stats
from scipy.stats import shapiro
from scipy.stats import kstest
import pickle

# Map 1 library
import plotly.express as px

# Map 2 libraries
import descartes
import geopandas as gpd
from shapely.geometry import Point, Polygon

# Librosa Libraries
import librosa
import librosa.display
import IPython.display as ipd

import sklearn

import warnings
```

```
warnings.filterwarnings('ignore')
```

A.1.2. Carga de los ficheros y visualización de audios

Con el fin de poder trabajar con ejemplos concretos para visualizarlos y ver en qué consisten, se implementa esta parte del código. Carga los dataframes que indican qué ficheros se van a contener en cada conjunto.

Permite ver el índice de un fichero en el dataframe de forma que junto con la ruta completa de acceso al fichero se puede reproducir y escuchar lo que dice. Hay que tener en cuenta que los ficheros que son visibles desde el dataframe que aquí se carga son todos los disponibles en el dataset HABLA y que generalmente el usuario sólo copia el subconjunto que va a utilizar, por lo que puede que no pueda reproducir el audio si introduce cualquier índice dentro del rango del dataframe.

Código A.2: Visualización de audios.

```
%cd /content/drive/MyDrive/HABLA
# Import data
protocol_df = pd.read_csv("./DATA/asvspoof2019_LA/protocol.txt"
    ,header=None,sep=" ")
train_csv = pd.read_csv("./DATA/asvspoof2019_LA/scp/train.lst",
    header=None)
test_csv = pd.read_csv("./DATA/asvspoof2019_LA/scp/test.lst",
    header=None)


train_csv.columns = ['filename']
test_csv.columns = ['filename']
protocol_df.columns = ['speaker','filename', 'None','TTS','
    label']
protocol_df['label'].value_counts()

#run to check the example index
protocol_df[(protocol_df['filename']=='arf_00295_01212219131')]

# Create Full Path so we can access data more easily
base_dir = './DATA/asvspoof2019_LA/train_dev/'
protocol_df['full_path'] = base_dir + protocol_df['filename'] +
     '.wav'

# Now let's sample a fiew audio files. The index here is the
    one checked before.
example_real = protocol_df['full_path'][32080]
example_fake = protocol_df['full_path'][70822]

#display the audio
from IPython.display import Audio, display
```

```
if os.path.exists(example_real):
    print(f"Archivo real encontrado: {example_real}")
    display(Audio(example_real))
else:
    print(f"Archivo real no encontrado: {example_real}")
```

A.1.3. Graficación de las ondas sonoras

Este´código permite generar gráficos sobre las ondas sonoras. Para representar las ondas se hace uso de dos características del sonido que se extraen mediante la librería `librosa`: el sonido (y) como un array de vibraciones en distintas intensidades de presión y la frecuencia de muestreo (sr) que es el número de muestras de audio transportadas por segundo, medido en Hz o kHz. En esta parte del código también se trata el audio mediante la función `librosa.effects.trim`.

Código A.3: Gráficación de ondas sonoras.

```
# Importing the 2 files
y_real, sr_real = librosa.load(example_real)
y_fake, sr_fake = librosa.load(example_fake)

fig, ax = plt.subplots(2, figsize = (16, 9))
fig.suptitle('Sound Waves', fontsize=16)
librosa.display.waveshow(y = y_real , sr = sr_real, color = "#
    A300F9", ax=ax[0])
librosa.display.waveshow(y = y_fake, sr = sr_fake, color = "
    #4300FF", ax=ax[1])

#trimming the audio
audio_real, _ = librosa.effects.trim(y_real,top_db=38)
audio_fake, _ = librosa.effects.trim(y_fake,top_db=40)

fig, ax = plt.subplots(2, figsize = (16, 9))
fig.suptitle('Sound Waves', fontsize=16)

librosa.display.waveshow(y = audio_real , sr = sr_real, color =
     "#A300F9", ax=ax[0])
librosa.display.waveshow(y = audio_fake, sr = sr_fake, color =
    "#4300FF", ax=ax[1])
```

A.1.4. Espectrogramas

En esta sección sólo se indica el código generador de los espectrogramas de cada parte para un ejemplo concreto, en el Colab se puede encontrar el código para la generación del espectrograma usual sin separar partes de la señal y para el otro ejemplo.

Código A.4: Espectrogramas de la parte real e imaginaria.

```
n_fft=512
hop_length=256
# Short-time Fourier transform (STFT)
D_real_real = np.real(librosa.stft(audio_real, n_fft = n_fft,
    hop_length = hop_length))
D_real_imag = np.imag(librosa.stft(audio_real, n_fft = n_fft,
    hop_length = hop_length))
# Convert an amplitude spectrogram to Decibels-scaled
    spectrogram.
DB_real_real = librosa.amplitude_to_db(D_real_real, ref = np.
    max)
DB_real_imag = librosa.amplitude_to_db(D_real_imag, ref = np.
    max)

# === PLOT ===
fig, ax = plt.subplots(1, 2, figsize=(12, 6))
fig.suptitle('Log Frequency Spectrogram Real Voice', fontsize
    =16)
img=librosa.display.specshow(DB_real_real, sr = sr_real,
    hop_length = hop_length, x_axis = 'time',
                          y_axis = 'log', cmap = 'cool', ax=ax
                              [0])
ax[0].set_title('Real Voice Real Part', fontsize=13)
plt.colorbar(img,ax=ax[0])

img=librosa.display.specshow(DB_real_imag, sr = sr_real,
    hop_length = hop_length, x_axis = 'time',
                          y_axis = 'log', cmap = 'cool', ax=ax
                              [1])
ax[1].set_title('Real Voice Imaginary Part', fontsize=13)
plt.colorbar(img,ax=ax[1])
```

A.1.5. Pruebas sobre la distribución

En esta sección se encuentra el código para realizar las pruebas de normalidad sobre los datos de la señal de audio que se mencionaban en la sección 5.1.2

En esta sección sólo se indica el código generador de los espectrogramas de cada parte para un ejemplo concreto, en el Colab se puede encontrar el código para la generación del espectrograma usual sin separar partes de la señal y para el otro ejemplo.

Código A.5: Histograma para la distribución de las partes real y compleja.

```
plt.figure(figsize=(10, 6))
plt.hist(imag_part_real, bins=100, alpha=0.9)
plt.xlim(-5, 5)
plt.title('Histograma de la Parte Imaginaria')
plt.xlabel('Parte Imaginaria')
plt.ylabel('Frecuencia')
plt.show()
```

Código A.6: Gráfico Q-Q para la distribución de las partes real y compleja.

```
plt.figure(figsize=(10, 6))
stats.probplot(imag_part_real.flatten(), dist="norm", plot=plt)
plt.xlim(-5, 5)
plt.title('Gr{\'a}fico Q-Q')
plt.show()
```

Código A.7: Pruebas de Shapiro y Kolmogorov-Smirnov para las partes real y compleja.

```
stat, p = shapiro(imag_part_real.flatten())
print('Prueba de normalidad de Shapiro-Wilk')
print('Estad{\'i}stico de Shapiro:', stat)
print('Valor p:', p)
if p > 0.05:
    print("No se puede rechazar la hip{\'o}tesis nula: los
       datos parecen seguir una distribuci{\'o}n normal.")
else:
    print("Se rechaza la hip{\'o}tesis nula: los datos no
       siguen una distribuci{\'o}n normal.")
stat, p = kstest(real_part_real.flatten(), 'norm')
print('Prueba de normalidad de Kolmogorov-Smirnov')
print('Estad{\'i}stico de KS:', stat)
print('Valor p:', p)
if p > 0.05:
    print("No se puede rechazar la hip{\'o}tesis nula: los
       datos parecen seguir una distribuci{\'o}n normal.")
else:
    print("Se rechaza la hip{\'o}tesis nula: los datos no
       siguen una distribuci{\'o}n normal.")
```

A.2. Estudio de características

En esta sección se encuentran los códigos usados para, partiendo de una serie de ficheros, calcular las características sobre cada uno.

En todos los casos se cargan primero los nombres de los ficheros desde un archivo pickle y luego se itera sobre ese listado para ir cargando los audios y llamar a la función definida en el mismo código. Dentro de dicho bucle correspondiente al cálculo de los estadísticos en el Colab se puede encontrar una comprobación para evitar posibles errores con la existencia de los ficheros, ya que al trabajar con datasets grandes a veces las tecnologías de almacenamiento pueden tener errores al hacer copias o migraciones de los datos.

Las características calculadas se almacenan de forma temporal en un dataframe que posteriormente se guarda en formato pickle.

A.2.1. Cálculo de estadísticos

Código A.8: Almacenamiento de los estadísticos básicos y la llamada a la función para cada fichero.

```
def calculate_features(y):
  # Calculate the percentiles
  percentile_5 = np.nanpercentile(np.nanpercentile(y, 5, axis
     =1),5)
  percentile_10 = np.nanpercentile(np.nanpercentile(y, 10, axis
     =1),10)
  percentile_50 = np.nanpercentile(np.nanpercentile(y, 50, axis
     =1),50)
  percentile_90 = np.nanpercentile(np.nanpercentile(y, 90, axis
     =1),90)
  percentile_95 = np.nanpercentile(np.nanpercentile(y, 95, axis
     =1),95)
  # Calculate the mean
  mean = np.mean(np.mean(y, axis=1))
  # Calculate the stddev
  std = np.std(np.std(y, axis=1))
  return percentile_5, percentile_10, percentile_50,
     percentile_90, percentile_95, mean, std

df1['ImagPartPercentile_5'] = None
df1['ImagPartPercentile_10'] = None
df1['ImagPartPercentile_50'] = None
df1['ImagPartPercentile_90'] = None
df1['ImagPartPercentile_95'] = None
df1['ImagPartMean'] = None
df1['ImagPartStd'] = None

for filename in df1['filename']:
    Audio, _ = librosa.load(filename)
    D_stft= librosa.stft(Audio, n_fft=n_fft, hop_length=
       hop_length)
    percentile_5, percentile_10, percentile_50, percentile_90,
       percentile_95, mean, std = calculate_features(np.imag(
       D_stft))
    df1.loc[df1['filename'] == filename, 'ImagPartPercentile_5'
       ] = percentile_5
    df1.loc[df1['filename'] == filename, 'ImagPartPercentile_10
       '] = percentile_10
    df1.loc[df1['filename'] == filename, 'ImagPartPercentile_50
       '] = percentile_50
    df1.loc[df1['filename'] == filename, 'ImagPartPercentile_90
       '] = percentile_90
```

```
30      df1.loc[df1['filename'] == filename, 'ImagPartPercentile_95
            '] = percentile_95
31      df1.loc[df1['filename'] == filename, 'ImagPartMean'] = mean
32      df1.loc[df1['filename'] == filename, 'ImagPartStd'] = std
```

A.2.2. Cálculo de características de Slope

Código A.9: Función para calcular el slope en las frecuencias 0-500Hz en segmentos con voz y sin voz.

```
1  def calculate_slope(y, sr):
2      freqs = librosa.fft_frequencies(sr=sr, n_fft=n_fft)
3      mask = (freqs >= 0) & (freqs <= 500)
4      D_low_freq = np.abs(y[mask, :]) # Selection of frequencies
           in the range
5      unvoiced_segments = np.mean(D_low_freq, axis=0) < np.
           percentile(D_low_freq, 25) # Filter out voiced segments
6      voiced_segments = np.mean(D_low_freq, axis=0) > np.
           percentile(D_low_freq, 25) # Filter out voiced segments
7      slopes_uv = []
8      slopes_v = []
9      for i in range(D_low_freq.shape[1]):  # Iterate on each
           segment
10         if unvoiced_segments[i]:  # If the segment is voiceless
11             slope, _, _, _, _ = linregress(freqs[mask],
                   D_low_freq[:, i])
12             slopes_uv.append(slope)
13         if voiced_segments[i]:  # If the segment is with voice
14             slope, _, _, _, _ = linregress(freqs[mask],
                   D_low_freq[:, i])
15             slopes_v.append(slope)
16     return np.mean(slopes_uv), np.mean(slopes_v)
17
18 df1['slopeUV0-500_sma3nz_amean_imag'] = None
19 df1['slopeUV0-500_sma3nz_amean_real'] = None
20 df1['slopeV0-500_sma3nz_amean_imag'] = None
21 df1['slopeV0-500_sma3nz_amean_real'] = None
22
23 for filename in df1['filename']:
24         Audio, sr = librosa.load(filename)
25         D_stft= librosa.stft(Audio, n_fft=n_fft, hop_length=
               hop_length)
26         slopeUV_imag, slopeV_imag = calculate_slope(np.imag(
               D_stft), sr)
27         slopeUV_real, slopeV_real= calculate_slope(np.real(
               D_stft), sr)
```

```
28          df1.loc[df1['filename'] == filename, 'slopeUV0-500
               _sma3nz_amean_imag'] = slopeUV_imag
29          df1.loc[df1['filename'] == filename, 'slopeUV0-500
               _sma3nz_amean_real'] = slopeUV_real
30          df1.loc[df1['filename'] == filename, 'slopeV0-500
               _sma3nz_amean_imag'] = slopeV_imag
31          df1.loc[df1['filename'] == filename, 'slopeV0-500
               _sma3nz_amean_real'] = slopeV_real
```

A.2.3. Cálculo de características de loudness

Código A.10: Función para calcular los percentiles de loudness.

```
1  def calculate_loudness_percentile(y):
2      loudness= librosa.amplitude_to_db(y)
3      loudness_percentile10 = np.percentile(loudness, 10)
4      loudness_percentile20 = np.percentile(loudness, 20)
5      loudness_percentile50 = np.percentile(loudness, 50)
6      loudness_percentile90 = np.percentile(loudness, 90)
7      return loudness_percentile10, loudness_percentile20,
          loudness_percentile50, loudness_percentile90
8
9
10 df1['loudness_imag_percentile10'] = None
11 df1['loudness_imag_percentile20'] = None
12 df1['loudness_imag_percentile50'] = None
13 df1['loudness_imag_percentile90'] = None
14 df1['loudness_real_percentile10'] = None
15 df1['loudness_real_percentile20'] = None
16 df1['loudness_real_percentile50'] = None
17 df1['loudness_real_percentile90'] = None
18
19 for filename in df1['filename']:
20   Audio, _ = librosa.load(filename)
21   Audio_trim, _ = librosa.effects.trim(Audio, top_db=38)
22   D_stft= librosa.stft(Audio_trim, n_fft=n_fft, hop_length=
        hop_length)
23   loudness_imag_percentile10, loudness_imag_percentile20,
        loudness_imag_percentile50, loudness_imag_percentile90 =
        calculate_loudness_percentile(np.imag(D_stft))
24   loudness_real_percentile10, loudness_real_percentile20,
        loudness_real_percentile50, loudness_real_percentile90 =
        calculate_loudness_percentile(np.real(D_stft))
25   df1.loc[df1['filename'] == filename, '
        loudness_imag_percentile10'] = loudness_imag_percentile10
```

```
26   df1.loc[df1['filename'] == filename, '
        loudness_imag_percentile20'] = loudness_imag_percentile20
27   df1.loc[df1['filename'] == filename, '
        loudness_imag_percentile50'] = loudness_imag_percentile50
28   df1.loc[df1['filename'] == filename, '
        loudness_imag_percentile90'] = loudness_imag_percentile90
29   df1.loc[df1['filename'] == filename, '
        loudness_real_percentile10'] = loudness_real_percentile10
30   df1.loc[df1['filename'] == filename, '
        loudness_real_percentile20'] = loudness_real_percentile20
31   df1.loc[df1['filename'] == filename, '
        loudness_real_percentile50'] = loudness_real_percentile50
32   df1.loc[df1['filename'] == filename, '
        loudness_real_percentile90'] = loudness_real_percentile90
```

A.2.4. Cálculo de características de F0

Código A.11: Función para calcular los semitonos de F0 desde 27.5Hz.

```
1  def calculate_f0(y):
2      f0 = librosa.yin(y, fmin=librosa.note_to_hz('C2'), fmax=
          librosa.note_to_hz('C5'))
3      f0semitone = 12 * np.log2(f0 / 27.5)
4      f0semitone_mean = np.nanmean(f0semitone)
5      return f0semitone_mean
6
7  df1['f0semitonefrom27.5_imag'] = None
8  df1['f0semitonefrom27.5_real'] = None
9
10 for filename in df1['filename']:
11   Audio, _ = librosa.load(filename)
12   D_stft= librosa.stft(Audio, n_fft=n_fft, hop_length=
        hop_length)
13   f0_imag = calculate_f0(np.imag(D_stft))
14   f0_real = calculate_f0(np.real(D_stft))
15   df1.loc[df1['filename'] == filename, 'f0semitonefrom27.5_imag
        '] = f0_imag
16   df1.loc[df1['filename'] == filename, 'f0semitonefrom27.5_real
        '] = f0_real
```

A.2.5. Cálculo de características de Shimmer y Jitter

Código A.12: Función para calcular las características de Shimmer y jitter.

```
1  import soundfile as sf
```

```
import parselmouth
from parselmouth.praat import call
from IPython.display import Audio, display

def calculate_shimmer_jitter(ruta):
    audio = parselmouth.Sound(ruta)
    print(audio)
    pointProcess = call(audio, "To PointProcess (periodic, cc)", 27.5, 440)
    localJitter = call(pointProcess, "Get jitter (local)", 0, 0, 0.0001, 0.02, 1.3)
    localShimmer = call([audio, pointProcess], "Get shimmer (local_dB)", 0, 0, 0.0001, 0.02, 1.3, 1.6)
    return localJitter, localShimmer

i=0
df1['shimmerLocaldB_imag'] = None
df1['shimmerLocaldB_real'] = None
df1['jitterLocal_imag'] = None
df1['jitterLocal_real'] = None
base_dir = './DATA/asvspoof2019_LA/train_dev_imag/'

for filename in df1['filename']:
        audio, sr = librosa.load(filename)
        D_stft= librosa.stft(audio, n_fft=n_fft, hop_length=hop_length)
        audio_imag = librosa.istft(np.imag(D_stft))
        audio_real = librosa.istft(np.real(D_stft))
        nombre_archivo = df1['filename'].iloc[i].replace("DATA/asvspoof2019_LA/train_dev/", "").replace(".wav", "")
        ruta_imag = base_dir + nombre_archivo + '_imag' + '.wav'
        ruta_real = base_dir + nombre_archivo + '_real' + '.wav'
        sf.write(ruta_imag, audio_imag, sr, format="WAV")
        sf.write(ruta_real, audio_real, sr, format="WAV")
        jitter_imag, shimmer_imag = calculate_shimmer_jitter(ruta_imag)
        jitter_real, shimmer_real = calculate_shimmer_jitter(ruta_real)
        df1.loc[df1['filename'] == filename, 'shimmerLocaldB_imag'] = shimmer_imag
        df1.loc[df1['filename'] == filename, 'shimmerLocaldB_real'] = shimmer_real
        df1.loc[df1['filename'] == filename, 'jitterLocal_imag'] = jitter_imag
```

```
36          df1.loc[df1['filename'] == filename, 'jitterLocal_real'
               ] = jitter_real
37          i = i+1
```

A.2.6. Extracción de características con OpenSMILE

En esta sección se presenta cómo gracias al toolkit *OpenSMILE* se extraen las características de GeMAPS en los ficheros indicados en el dataframe. Estas características se ven referenciadas en la sección 5.1.1 y en el Apéndice B.

En este código se puede configurar si se quieren las características del set GeMAPS(v01a, v01b), emobase, eGeMAPS (v01a, v01b, v02) o ComParE_2016 y el nivel que se desea entre Functionals, LowLevelDescriptors o LowLevelDescriptors_Deltas.

Código A.13: Extracción de características con OpenSMILE.

```
import opensmile
import audb

#displays a graph for the distribution of the data in the
    dataset
db.tables['files'].df.tts.value_counts().plot(kind='pie')
df2 = db.tables['files'].df

#the processor is configured and a new dataframe is generated
    with the database index and the calculated characteristics
smile = opensmile.Smile(
    feature_set=opensmile.FeatureSet.GeMAPSv01b,
    feature_level=opensmile.FeatureLevel.Functionals,
)

feats_df = smile.process_files(df2.index)
feats_df.shape
```

A.3. Entrenamiento y uso de los modelos

A.3.1. Selección de datos para el entrenamiento del modelo

En esta sección se observa el código que ha sido necesario para elegir los datos, concretamente las características, que serían necesarias para cada uno de los experimentos realizados en el proyecto. Se presenta en un comienzo el código para el tratamiento de los dataframes al unir los que albergan distintas características. Aunque uno de los dataframes tenga un índice complejo, es decir, que tenga más de un dato o atributo por índice como es el caso del dataframe de OpenSMILE, se puede unir con otro como los calculados en este trabajo siempre que uno de esos atributos sea común en los índices de ambos dataframes. Funciona de manera bastante similar a las tablas de una base de datos, incluso hay distintos tipos de JOINs, pero en este caso se hace uno básico que corresponde a un LEFT JOIN, es decir se toma el dataframe "feats_df" como principal y se le añaden las columnas del "df1".

Código A.14: Selección de datos para el entrenamiento del modelo.

```
#feats_df.to_pickle('habladb.pkl') #line to store dataframe in
    pickle file
feats_df = pd.read_pickle('habladb.pkl') #line to read pickle
    file

#Transformation of the new dataframe and joining to the
    OpenSMILE dataframe
df1 = df1.rename(columns={'filename': 'file'})
feats_df_new = feats_df.join(df1.set_index('file'))
feats_df_new

#reading of the csv file indexing the files for training
df = pd.read_csv('protocol_train_dev.csv')
df
```

Código A.15: Selección de los datos de entrenamiento y test.

```
X = feats_df_new.loc[:, feats_df_new.columns != 'label']
y = df['label']
seed = 42
np.random.seed(seed)

X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.20, random_state=seed)
```

A.3.2. Modelo EBM

Código A.16: Configuración y entrenamiento del modelo EBM.

```
from interpret.glassbox import ExplainableBoostingClassifier
ebm = ExplainableBoostingClassifier()
ebm.fit(X_train, y_train)
```

Código A.17: Evaluación del modelo EBM.

```
predictions = ebm.predict(X_test)
cm = confusion_matrix(y_test, predictions)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
    display_labels=ebm.classes_)
disp.plot()
plt.show()
```

Código A.18: Gráficos para la interpretación del modelo.

```
#confusion matrix
predictions = ebm.predict(X_test)
cm = confusion_matrix(y_test, predictions)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
    display_labels=ebm.classes_)
disp.plot()
plt.show()

#ranking of global importance
ebm_global = ebm.explain_global(name='EBM')
show(ebm_global)
```

A.3.3. Modelo LightGBM

Código A.19: Configuración y entrenamiento del modelo LightGBM.

```
# Create LightGBM dataset objects
lgb_train = lgb.Dataset(X_train, y_train)
lgb_eval = lgb.Dataset(X_test, Y_test, reference=lgb_train)
# Model configuration
params = {
    'objective': 'binary',
    'metric': 'binary_logloss',
    'boosting_type': 'gbdt',
    'learning_rate': 0.05,
    'num_leaves': 31,
    'max_depth': -1,
    'verbose': -1
}
# Train the model
clf = lgb.train(params, lgb_train, num_boost_round=100)
```

Código A.20: Predicciones y evaluación del modelo LightGBM.

```
# Predictions
y_pred = gbm.predict(test_data, num_iteration=gbm.
    best_iteration)
y_pred_binary = (y_pred >= 0.5).astype(int)
# Evaluate the model
accuracy = accuracy_score(test_label, y_pred_binary)
print(f"Accuracy: {accuracy}")
```

B

Listado de características extraídas con OpenSMILE

Las características que se han extraído con GeMAPS constituyen la lista siguiente, agrupándolas en los grupos indicados en la sección 5.1.1.

Parámetros relacionados con la frecuencia:

- Pitch, F0 logarítmica en una escala de frecuencia de semitonos, comenzando en 27,5 Hz (semitono 0).
- Jitter, desviaciones en longitudes de períodos F0 individuales consecutivos.
- Formante 1, 2 y 3 frecuencia, frecuencia central del primer, segundo y tercer formante
- Formante 1, ancho de banda del primer formante.
- Formante 2-3 ancho de banda añadido para completar los parámetros de los formantes 1-3.

Parámetros relacionados con la energía/amplitud:

- Shimmer, diferencia de las amplitudes pico de periodos F0 consecutivos.
- Loudness, estimación de la intensidad de la señal percibida a partir de un espectro auditivo.
- Relación entre armónicos y ruido (HNR), relación entre la energía de los componentes armónicos y la energía de los componentes ruidosos.

Parámetros espectrales (equilibrio/forma/dinámica):

- Alpha Ratio, relación entre la energía sumada de 50-1000 Hz y 1-5 kHz

- Hammarberg Index, relación entre el pico de energía más fuerte en la región de 0-2 kHz y el pico más fuerte en la región de 2-5 kHz.
- Pendiente espectral 0-500 Hz y 500-1500 Hz, pendiente de regresión lineal del espectro de potencia logarítmica dentro de las dos bandas dadas.
- Energía relativa de los formantes 1, 2 y 3, así como la relación entre la energía del pico armónico espectral en la frecuencia central del primer, segundo y tercer formante y la energía del pico espectral en F0.
- Diferencia armónica H1-H2, relación entre la energía del primer armónico F0 (H1) y la energía del segundo armónico F0 (H2).
- Diferencia armónica H1-A3, relación entre la energía del primer armónico F0 (H1) y la energía del armónico más agudo del tercer rango de formantes (A3).
- MFCC 1-4 Mel-Frequency Cepstral Coefficients 1-4.
- Diferencia de flujo espectral de los espectros de dos fotogramas consecutivos.

Características temporales:

- La tasa de picos de sonoridad, es decir, el número de picos de sonoridad por segundo.
- La longitud media y la desviación estándar de las regiones con voz continua (F0 ¿0).
- La longitud media y la desviación estándar de las regiones sin voz (F0 = 0; aproximación a las pausas).
- El número de regiones con voz continua por segundo (pseudo tasa silábica).