UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería de Telecomunicación

Detección de ruido de origen humano en el Parc de l'Albufera mediante técnicas de aprendizaje máquina

Trabajo Fin de Grado

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

AUTOR/A: Lanau Carrasco, Alba

Tutor/a: Piñero Sipán, María Gemma

CURSO ACADÉMICO: 2022/2023

Resumen

Las técnicas de *Machine Learning* han desempeñado un papel fundamental al introducir nuevos enfoques en el procesamiento de datos, lo que a su vez ha permitido un análisis más profundo de las señales de audio. Estas señales se conforman por una superposición de múltiples fuentes sonoras, sin embargo, no todas ellas resultan relevantes. Los estudios medioambientales se han centrado, en gran medida, en el análisis de sonidos. Uno de estos trabajos se ha llevado a cabo en el contexto del Parc de L'Albufera de Valencia. En este Trabajo Fin de Grado se han examinado las grabaciones de audio realizadas en el parque con el objetivo de detectar y clasificar los sonidos de origen humano para diferenciarlos de aquellos producidos por la fauna. El proceso implica la extracción de características y la aplicación de técnicas de aprendizaje no supervisado para realizar la clasificación. A este respecto, se han empleado los algoritmos *k-means*, *k-medoids* y los modelos de mezcla gaussianos. El primer método ha demostrado ser el más eficaz, logrando resultados precisos en la tarea de clasificación de las señales sonoras.

Resum

Les tècniques de *Machine Learning* han exercit un paper fonamental en introduir nous enfocaments en el processament de dades, la qual cosa al seu torn ha permés una anàlisi més profunda dels senyals d'àudio. Aquests senyals es conformen per una superposició de múltiples fonts sonores, no obstant això, no totes elles resulten rellevants. Els estudis mediambientals s'han centrat, en gran manera, en l'anàlisi de sons. Un d'aquests treballs s'ha dut a terme en el context del Parc de L'Albufera de València. En aquest TFG s'han examinat els enregistraments d'àudio realitzades al parc amb l'objectiu de detectar i classificar els sons d'origen humà per a diferenciar-los d'aquells produïts per la fauna. El procés implica l'extracció de característiques i l'aplicació de tècniques d'aprenentatge no supervisat per a realitzar la classificació. Referent a això, s'han emprat els algorismes *k-means, k-medoids* i els models gaussians. El primer mètode ha demostrat ser el més eficaç, aconseguint resultats precisos en la tasca de classificació dels senyals sonors.

Abstract

Machine Learning techniques have played a fundamental role in introducing new approaches to data processing, which in turn has allowed a deeper analysis of audio signals. These signals are made up of a superposition of multiple sound sources. However, not all of them are relevant. Environmental studies have largely focused on sound analysis. One of these studies has been carried out in the context of the Parc de L'Albufera in Valencia. In this work, audio recordings taken in the park have been examined for detecting and classifying sounds of human origin in order to differentiate them from those produced by wildlife. The process involves the extraction of features and the application of unsupervised learning techniques to perform the classification. In this regard, k-means, k-medoids and Gaussian mixture models have been employed. The first method has proven to be the most effective, achieving accurate results in the sound signal classification task.

A mi tutora por acompañarme en este proceso y guiarme hasta el final

A mis padres, mi hermana y mis abuelos por apoyarme siempre

Índice general

I Memoria

Índice de figuras

Índice de tablas

Listado de siglas empleadas

ML Machine Learning

DL Deep Leaning

SED Sound Event Detection

GMM Gaussian Mixture Models

HMM Hidden Markov Models

DNN Deep Neural Network

TF Transformada de Fourier

DFT Discrete Fourier Transform

FFT Fast Fourier Transform

STFT Short Time Fourier Transform

MFCC Mel Frequency Cepstral Coefficients

DCT Discrete Cosine Transform

SSE Sum of Squared Error

EM Expectation Maximization

DBSCAN Density-based Spatial clustering

PCA Principal Components Analysis

Parte I

Memoria

Capítulo 1.

Introducción

1.1 Motivación y descripción del problema

Los estudios medioambientales permiten conciliar la relación entre el ser humano y su actividad con los entornos naturales.

Tradicionalmente estos estudios han requerido de muchos recursos humanos, tanto en tiempo como en número personas, para llevar a término la recopilación de datos y su posterior análisis, debido a la falta de técnicas que automatizaran estos procesos.

Actualmente todos estos procesos se han visto agilizados con el desarrollo de nuevas técnicas de adquisición de audio e imagen, así como de procesamiento de datos. Por un lado, la automatización del proceso reduce el tiempo empleado y el número de personas que intervienen, por otro, el uso de los modelos de aprendizaje de maquina (*machine learning*, ML) y técnicas de aprendizaje profundo (*deep learning*, DL*)* permiten extraer de manera más eficaz la información útil para los biólogos.

L'Albufera es un parque natural protegido que sirve de hábitat para una amplia variedad de aves acuáticas, lo que lo convierte en un entorno de gran relevancia biológica. Sin embargo, en este entorno no solo convive flora y fauna, sino que también es escenario de actividades humanas como el cultivo de los arrozales, la pesca y la explotación turísticas. Estas actividades producen contaminación sonora. En este escenario se lleva a cabo un proyecto biológico centrado en la identificación y clasificación de aves acuáticas mediante el análisis de grabaciones sonoras. Como es de suponer estas grabaciones contienen tanto los cantos de las aves como ruidos de origen humano. Este proyecto se enfoca en emplear las técnicas de detección de eventos sonoros para capturar estos ruidos no deseados y en su posterior clasificación mediante algoritmos de ML y DL no supervisados.

1.2 Estado del arte

1.2.1 Algoritmo de Machine Learning no supervisado

La información está presente en todos los aspectos de la vida y en este último siglo su volumen no ha dejado de crecer de manera constante. Esto plantea nuevos desafíos y dilemas que requieren nuestra atención y soluciones innovadoras.

Lo más interesante es descubrir la utilidad intrínseca de esta información. Para alcanzar este propósito, se realiza el análisis de datos, un proceso que aplica técnicas y procedimiento a los datos en bruto con el fin de extraer conocimientos útiles que puedan guiar en la toma de decisiones informadas.

A través del análisis de datos, se consigue identificar y extraer patrones de los conjuntos de datos. Para lograr este fin intervienen diversas disciplinas, matemáticas, inteligencia artificial y *machine learning*. El resultado permite obtener respuestas a preguntas como "¿qué está pasando?", "¿por

qué ha ocurrido?", "¿qué podría pasar?". Al mismo tiempo este análisis presenta una serie de desafíos. Entre ellos, la necesidad de manejar los datos mientras estos mantienen su integridad, garantizar su accesibilidad en cualquier instante, hacer representaciones visuales que sean comprensibles para los clientes y verificar que los datos sean correctos y de calidad, con el finde evitar obtener resultados incorrectos. [1]

Machine learning es una rama de la Inteligencia Artificial que consiste en dejar que los algoritmos aprendan de forma autónoma a realizar predicciones a partir de datos, descubriendo patrones en ellos. Se puede abordar desde dos perspectivas: el aprendizaje supervisado y el aprendizaje no supervisado, existe una clara diferencia entre ambos. En el primer caso los conjuntos de datos de entrada son etiquetados para que la maquina pueda medir su precisión, este etiquetado requiere de la intervención humana. Al contrario, el segundo caso emplea datos sin procesar y sin etiquetar, dejando a la maquina interpretarlos y procesarlos como considere.

El siguiente ejemplo muestra esta diferencia. Si ML fuera un niño aprendiendo a montar en bicicleta, el aprendizaje supervisado sería el padre que acompaña al niño mientras monta y el aprendizaje no supervisado consistiría en dejar que el niño aprendiera por sí solo.

Este proyecto se centra en los métodos de clasificación no supervisados.

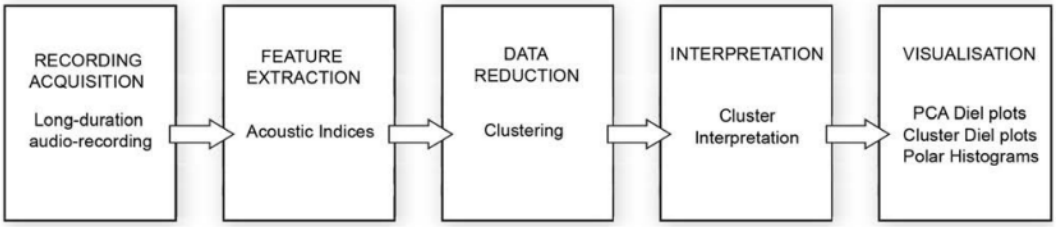

Figura 1.1. Diagrama procesamiento de datos

Este proceso se inicia con la adquisición de la base de datos, que podrían ser una serie de imágenes o sonidos sin etiquetar. Posteriormente es necesario definir una serie de condiciones o características que guiarán la clasificación. Estas permitirán a la maquina diferenciar los datos y agruparlos basándose en sus similitudes. Una vez obtenida la clasificación se deben interpretar los resultados y comprender los patrones obtenidos. Para ello puede ser útil visualizarlos en graficas o histogramas.[2] Esta serie de pasos se muestran en la Figura 1.1

Su eficacia reside en estudiar la estructura intrínseca de los datos y encontrar los patrones ocultos, extrayendo así ideas valiosas.

Las principales aplicaciones de esta técnica son la segmentación de conjuntos de datos por atributos compartidos, la detección de anomalías que no encajan en ningún grupo, identificar conjuntos de elementos que puede aparecer juntos en su conjunto de datos y la simplificación del dataset. [3]

Entre sus aplicaciones prácticas se encuentran:

- Visión computacional: reconocimiento de objetos.
- Imágenes médicas: detección y clasificación de imágenes en radiología y anatomía patológica.
- Perfiles del cliente: compresión de los hábitos de compra del cliente.
- Búsqueda de noticias: agrupa artículos relacionados.

La técnica de análisis de agrupación se puede separar en dos clases *hard y soft clustering*. Los algoritmos de *hard clustering* clasifican cada elemento de datos en un único clúster, mientras que en los algoritmos de *soft clustering* cada elemento puede pertenecer a más de un clúster, por ello tiene asignado un vector de probabilidad que indica su nivel de pertenencia a cada clúster. [4]

1.2.2 Detección de eventos sonoros

Sitúense en el comedor de su casa, cierren los ojos y escuchen. A nuestro alrededor se puede detectar diferentes sonidos como la música de la radio, ruidos de la vajilla en la cocina, el timbre sonando y el tráfico de la calle. Nuestro cerebro es capaz de detectar todos esos sonidos, filtrarlos y focalizar su atención en el que le interesa. Las investigaciones actuales se centran en la búsqueda de técnicas que permitan a la maquina realizar esta misma tarea, y es lo que se conoce como Detección de Eventos Sonoros, por sus siglas inglesas *Sound Event Detection* (SED). Tradicionalmente el estudio de una escena se centraba en analizar las imágenes, ahora técnicas como el SED están ganando terreno. La detección de eventos sonoros es la tarea de reconocer dichos eventos y su tiempo inicial y final en una grabación. Los sonidos no se presentan de manera aislada, sino que están compuestos por una superposición de ellos, a esto se le denomina *polyphonic* SED. [5]

En la detección de sonidos se localizan dos campos de estudio:

- La voz y la música que se distinguen por sus cualidades sonoras ya que contiene frecuencias fundamentales y armónicos que permiten distinguirlas. Estas características se utilizan en aplicaciones como reconocimiento de voz, reconocimiento de locutores y recomendadores de música.

- Los sonidos ambientales [6] [7], como puede ser la lluvia, el canto de los pájaros, el claxon de un coche, etc. Su detección y clasificación presentan una serie de desafíos a los que debe hacer frente el SED. Por un lado, estos desafíos están relacionados con la diversidad de su naturaleza y como se presentan en el entorno, es decir; su duración varia y se muestran de manera intermitente, siendo impredecibles. Además, su captación requiere del uso de micrófonos que pueden encontrarse lejos de la fuente del sonido esto dificulta la detección pues las características del sonido variarán (la presión) e incluso otros sonidos o ruidos pueden enmascararlo.

  Por otro lado, presenta retos relacionados con la recopilación de datos y los procesos de etiquetado, pues existen un número ilimitado de sonidos que se presentan simultáneamente y no existe reglas predefinidas que ayuden en su detección.

  Estos factores obstaculizan la creación de un modelo de detección de eventos sonoros universal. Cada aplicación deberá construir su propio modelo según su conjunto de datos y el propósito específico que pretenda cumplir.

Para efectuar la detección de eventos sonoros tradicionalmente, se han empleado técnicas de aprendizaje supervisado [8] creando un modelo de SED que permita realizar predicciones de múltiples clases de sonidos que aparecen al mismo tiempo. Este enfoque conocido como *multiclass multilabel classification* se basa en buscar patrones de actividad correspondientes a múltiples etiquetas en un mismo segmento de audio obteniendo como resultado la Figura 1.2. en la que un único segmento puede estar asociado con múltiples clases de sonidos.

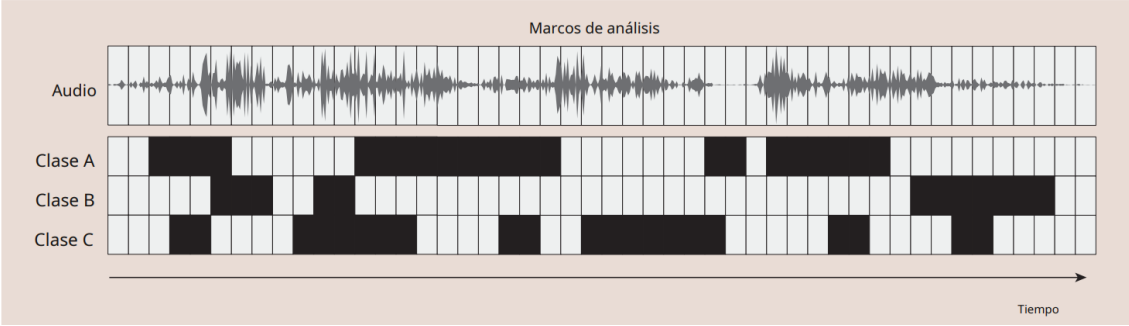

Figura 1.2. Clasificación *multilabel* de segmentos de audios [5].

Inicialmente para llevar a cabo la SED se utilizaron técnicas de clasificación como los *Gaussian Mixture Models* (GMM) y los *Hidden Markov Models* (HMM) que funcionaban para la clasificación de voz y música, sin embargo, esto no dio resultado pues los eventos sonoros no tienen patrones predecibles. Se centraron en utilizar técnicas que permitan la clasificación multietiqueta, tales como, clasificación binaria para cada clase de evento de sonido [9], varias rondas de decodificación de Viterbi [10] o preprocesamiento que implicaba aislar la fuente de sonido [11]. En los últimos años ha ganado relevancia el uso de *Deep Neural Network* (DNN) en modelos de SED. Esto se debe a que permite resolver de manera más eficiente el problema de *multiclass multilabel classification*, a través de la activación simultánea de múltiples neuronas en la capa de salida.

Las bases de datos que usan las aplicaciones SED necesitan contar con un conjunto de datos amplio para su correcto funcionamiento. Esta necesidad surge al considerar la variabilidad con la que los sonidos de cada clase pueden presentarse en el entorno, lo que requiere cubrir todas las posibles situaciones. En el caso de sonidos ambientales es difícil capturar el sonido de forma aislada. Por ejemplo, el ruido de una sierra eléctrica en una obra en la calle requiere la recopilación de muchos audios para que el programa sea capaz de diferenciar entre este ruido y cualquier otro similar producido en la obra, como el de un martillo eléctrico.

En el proceso de etiquetado de los audios se pueden emplear etiquetas fuertes o etiquetas débiles. La diferencia radica en que las etiquetas fuertes son más precisas pues indican el tiempo de inicio y fin de esa clase de sonido, mientras que las etiquetas débiles simplemente señalan la presencia de ese sonido en la grabación. La manera de generar etiquetas precisas es creando señales de audio sintéticas mezclando el evento sonoro con ruido de fondo obtenido en gran diversidad de datos. Se pueden emplear conjunto de datos públicamente disponibles como son TUT Sound Event 2016 [9] o FSDnoisy 18k.

Las técnicas de SED se empleaban para reconocer sonidos medioambientales en la naturaleza y en entornos urbanos. Las utilidades son muy diversas: mejorar la seguridad de las ciudades, detectar la presencia de animales. Estos sistemas también pueden disponer de cámaras que se activarán cuando un sonido sea detectado [12] [13].

La evolución en los algoritmos de *Machine learning* y *Deep learning* han permitido ampliar sus aplicaciones llegando a abarcar al campo de la medicina, facilitando información útil a los doctores para mejorar el diagnóstico [14] [15].

El presente proyecto consiste en la detección de ruido de origen humano en el Parc de l'Albufera de Valencia mediante técnicas de aprendizaje máquina. Para ello se ha de generar una base de datos compuesta por audios extraídos de distinto nodos colocados en el parque. Estos audios son de especial interés porque contienen sonidos de pájaros, pero a su vez tiene contaminación

acústica de origen humano, como conversaciones, el motor de las lanchas o los aviones. Para poder detectar estos ruidos se ha realizado un preprocesado de las señales mediante un cambio de domino de tiempo a tiempo-frecuencia, obteniendo espectrogramas de Mel para cada muestra. Posteriormente se ha obtenido un conjunto de características que servirán de entrada para un clasificador no supervisado.

1.3 Objetivos del proyecto

El objetivo general de este trabajo fin de grado es analizar las grabaciones realizadas en el Parc de l'Albufera de Valencia con el objeto de detectar sonidos de origen humano y diferenciarlos de los sonidos producidos por la fauna del parque.

Me centraré en diseñar, implementar y validar técnicas de aprendizaje de máquina para el análisis de audios, estudiando así la conveniencia de estas. De esta manera, se busca demostrar la flexibilidad de esta tecnología al aplicarla en tareas de clasificación.

Para hacer frente al problema de detección de eventos sonoros producidos por humanos planteo utilizar algoritmos de *Machine Learning* no supervisados capaces de detectar patrones en los datos y generar agrupaciones válidas de estos. De esta manera tras identificarlos se podrán eliminar de las grabaciones con el fin de estudiar los sonidos propios de las aves y el resto de fauna del parque.

Con la finalidad de ejecutar el objetivo principal he distribuido la tarea en una serie de objetivos secundarios, o pasos a seguir, que son los siguientes:

Selección y preparación de la base de datos a partir de las grabaciones realizadas en el parque.

Documentación y evaluación de las prestaciones de los algoritmos de *Machine Learning* no supervisado.

Evaluación de los resultados obtenidos y extracción de conclusiones.

1.4 Estructura del documento

El capítulo 2 se centra en la metodología, exponiendo el marco teórico del proyecto. En él se explica la teoría en la que se basa el procesado de la señal y se describen las características extraídas de los audios. Posteriormente se enumeran y desglosan los distintos métodos de clasificación no supervisados y como se aplican estos al entorno de Matlab. Concluye con una explicación del Análisis de Componentes Principales que se ejecutará más adelante.

El capítulo 3 se inicia con la descripción de la base de datos empleada para la detección del ruido humano y luego se centra en las pruebas realizadas. Se aplican los diferentes algoritmos, se realizan modificaciones y se analizan los resultados.

En el capítulo 4 se cierra la memoria con las conclusiones y las líneas futuras.

Capítulo 2.

Metodología

En este capítulo se procede a detallar las técnicas utilizadas en el procesado de las señales de audio y los diferentes algoritmos de *clustering* empleados para organizar estas señales en categorías.

2.1 Técnicas de procesado

Si nos adentramos en el análisis de una señal de audio en crudo, nos encontramos con gran cantidad de información que podría no resultar útil. Sin embargo, gracias a la creación y desarrollo de técnicas de procesamiento y extracción de características es posible encontrar patrones o propiedades en la señal que permitan revelar aspectos clave.

Para poder llevarlo a cabo, este trabajo parte de la obtención del espectrograma de la señal. De este modo se obtiene información en el dominio temporal y frecuencial, generando una imagen de la señal.

A continuación, se explican las técnicas empleadas para la generación del espectrograma.

2.1.1 Short Time Fourier Transform en tiempo discreto

Una señal de audio está compuesta por múltiples señales de onda u ondas sonoras con diferentes frecuencias que se superponen entre sí. Esta señal se puede representar a lo largo del tiempo mostrando sus amplitudes. Con el objetivo de conseguir más información sobre ella se puede aplicar la Transformada de Fourier. La Transformada de Fourier (TF) es una fórmula matemática empleada para transformar una señal del dominio de tiempo al dominio de la frecuencia y viceversa, permitiendo analizar el contenido espectral de una señal y comprender su comportamiento en frecuencia. Así podemos extraer las frecuencias correspondientes a cada sonido. Figura 2.1.

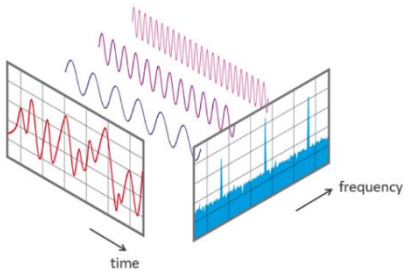

Figura 2.1. Transformación del dominio de tiempo al dominio de la frecuencia [16]

El uso de la TF abarca gran variedad de situaciones por ello existen diversas variantes. El más indicado para el análisis y procesado de señales es la Transformada de Fourier Discreta (*Discrete Fourier Transform*, DFT). Es similar a la FT pero se aplica a una secuencia finita de la señal tras ser muestreada y su contenido en frecuencia también es muestreado. Esto se expresa en la siguiente fórmula [17] :

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-\frac{j2\pi k}{N} n} \qquad k = 0, \dots, N-1 \tag{2.1}$$

A partir de este algoritmo surge la Transformada Rápida de Fourier (*Fast Fourier Transform*, FFT) utilizada para calcular de manera más eficiente la DFT.

La Transformada de Fourier de Tiempo Corto (*Short Time Fourier Transform*, STFT) se emplea en señales digitales de audio, al tratarse de señales no estacionarias. Esta función trabaja con tramas de la señal original, para ello utiliza una ventana con una longitud determinada que divide la señal y posteriormente aplica la FFT a cada trama para determinar las frecuencias contenidas en ese segmento. Suele permitir solape entre ellas. Este proceso se repite hasta recorrer toda la señal. La STFT se expresa [17]:

$$X[n,k] = \sum_{n=0}^{N-1} w[m] x[n+m] e^{-\frac{j2\pi k}{N} n} \tag{2.2}$$

Donde n es una variable discreta que indica el instante de tiempo, $x[n+m]$ es la señal de entrada y $w[m]$ la ventana que la divide. El resultado obtenido tiene valores complejos, estos se presentan en una matriz que muestra la fase y la magnitud para cada punto en frecuencia y tiempo.

2.1.2 Espectrograma

El espectrograma es el módulo de la STFT, habitualmente se calcula en unidades logarítmicas o decibelios. Proporciona una representación visual de la distribución de energía de la señal. En el eje vertical se muestra la frecuencia y la intensidad del sonido, indicada con niveles de colores en decibelios. En el eje horizontal se sitúa la línea temporal [18]. Esto se muestra en Figura 2.2.

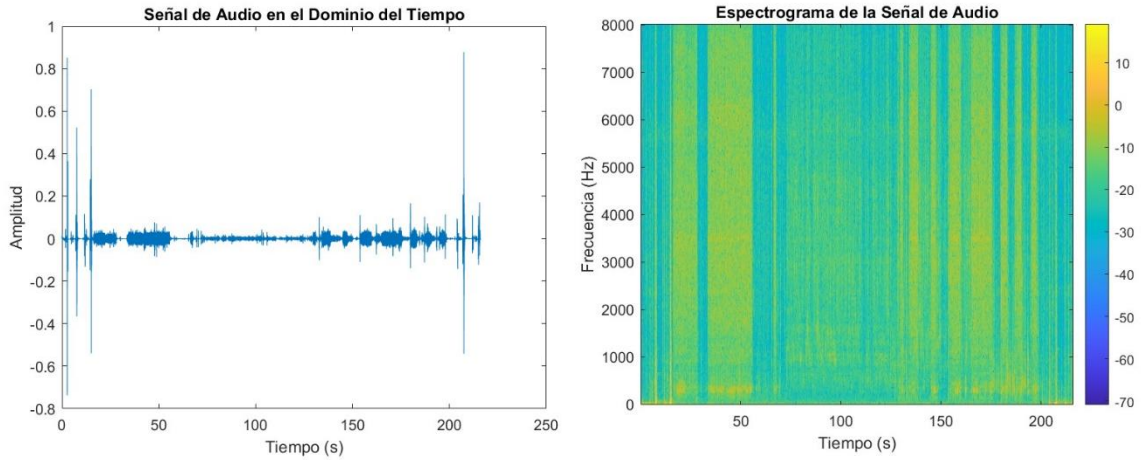

Figura 2.2. Señal en tiempo y espectrograma de la señal

Un parámetro a considerar es el tamaño de la ventana utilizada en la STFT, pues afecta directamente a los niveles de resolución del espectrograma.

2.1.3 Espectrograma de Mel

El espectrograma de Mel es una variación del espectrograma, muestra el contenido en frecuencia de una señal de audio, pero cambia el eje de frecuencias.

Los humanos no percibimos las frecuencias de manera lineal, es decir, captamos mejor los cambios en bajas frecuencias que en altas frecuencias [19]. Para poder plasmar esto en los espectrogramas se utiliza la escala de Mel, una escala de percepción que se ajusta a la respuesta en frecuencia no lineal del oído humano. Esta asigna a cada todo con una frecuencia real medida en Hz un tono subjetivo en la escala de Mel, esto se realiza utilizando la ecuación 2.3.

$$f_{mel} = 2595 \log_{10}(1 + \frac{f}{700}) \tag{2.3}$$

Se trata de una técnica ampliamente utilizada en el procesamiento de señales de audio.

Para crear el espectrograma de Mel se comienza fragmentando la señal de entrada. Para ello se utiliza una ventana de determinado tamaño que muestree la señal y se van haciendo saltos con solape entre muestras (*overlap)* cada vez que se muestrea la siguiente ventana. Posteriormente se calcula la FFT de cada fragmento haciendo así un cambio al dominio de la frecuencia. Luego se filtra cada fragmento al pasar por el banco de filtro de Mel, donde las magnitudes del espectro se multiplican por el valor de cada filtro. Finalmente se suman las magnitudes de cada filtro y se concatenan. De esta manera se consigue respetar cada trama en la escala logarítmica de Mel. Este proceso se muestra en la Figura 2.3.

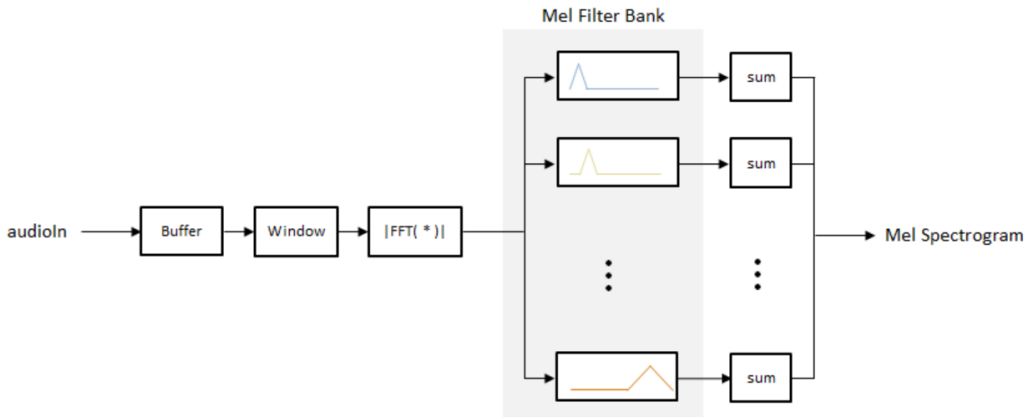

Figura 2.3. Obtención del espectrograma de Mel [20]

La Figura 2.4 muestra el banco de filtro de Mel se trata de una serie de filtros triangulares, MelBands, superpuestos y equidistantes en la escala Mel. Cada filtro recoge la energía de la señal en una banda de frecuencia distinta. El eje horizontal representa las bandas de frecuencias que abarca cada filtro.

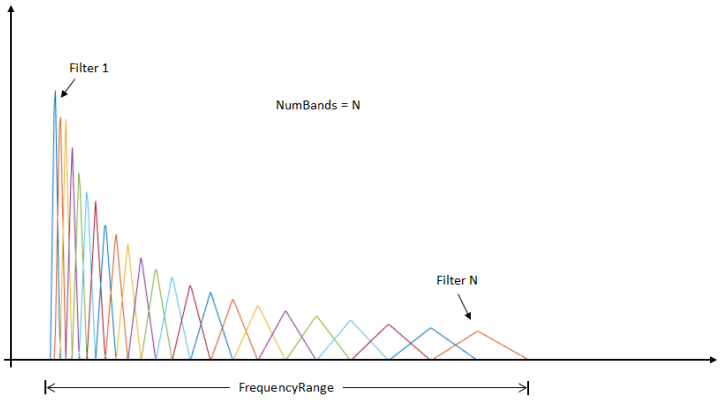

Figura 2.4. Banco de filtros de Mel [20].

El banco de filtros de Mel suele estar normalizado, de no estarlo las magnitudes de las MelBands más altas terminarían siendo valores más grandes y no correspondería con la percepción del oído humano.

Esta técnica de filtrado se emplea principalmente para realizar reconocimiento de voz y clasificación de sonidos, detención de eventos en el ambiente. Debido a que permite extraer características relevantes para la clasificación.

2.2 Características

Para mejorar la precisión de la clasificación se requiere llevar a cabo una etapa de preprocesamiento en cada audio con el propósito de identificar y extraer la información más significativa. Esto permitirá la clasificación de las señales de audio en diferentes categorías.

En esta etapa se realizará la extracción de características o *Feature Extraction* sobre los audios. De tal forma que cada audio quedará descrito por una serie de valores, es decir, un vector de características.

La selección de las características a extraer es un proceso complejo que requiere un conocimiento previo del problema a abordar. Es esencial determinar qué características son las más pertinentes, ya que estas contribuirán en mayor medida a conseguir información sobre los datos.

Existen muchas características y cada una proporciona información diferente. Puede darse el caso de que dos características tengan cierta correlación, lo que implica que ambas comparten la misma información y esto no sería muy útil para la clasificación. En contraposición, si las características presentan una menor correlación entre sí, más información proporcionaran para describir el audio.

Algunas veces no resulta evidente qué componentes de la señal aportan más información por ello lo mejor suele ser extraer todas las características disponibles y luego evaluar qué combinación de estas produce el mejor resultado para la terea que se esté realizando.

A continuación, realizo una enumeración de las características más utilizadas para analizar y reconocer sonidos.

2.2.1 Coeficientes Cepstrales de frecuencia Mel

Los coeficientes cepstrales de frecuencia Mel (*Mel Frequency Cepstral Coefficients*, MFCC) proporcionan una representación compacta de las características espectrales de una señal de audio considerando la percepción no lineal del oído humano [21].

Las MFCC forman un vector de características, se trata de una serie de coeficientes que describen la forma de la envolvente espectral en escala Mel obtenida de la señal.

Las MFCC se calculan siguiendo unos pasos, estos se observan en la Figura 2.5.

Figura 2.5 Proceso para generar las MFCC

En primer lugar, se realiza un preénfasis de la señal para realzar las altas frecuencias y así mejorar la relación seña a ruido. Luego se divide la señal de entrada en tramas con el uso de una ventana que permite cierto acoplamiento. A continuación, se aplica la Short Time Fourier Transform, STFT, a cada trama. Los coeficientes resultantes se filtran usando el banco de filtro de Mel, en cada banda se obtiene la energía de la señal en diferentes frecuencias. La magnitud obtenida de cada banda se pasa a una escala logarítmica, con el fin de que estos valores se asemejen a la percepción de oído humano. Finalmente se aplica la Transformada Discreta de Coseno (*Discrete Cosine Tranform*, DCT) sobre los coeficientes logarítmicos para decorrelar la información. De esta manera se consigue el vector de coeficientes MFCC que representan las características más importantes de la señal. La Figura 2.6 muestra un ejemplo.

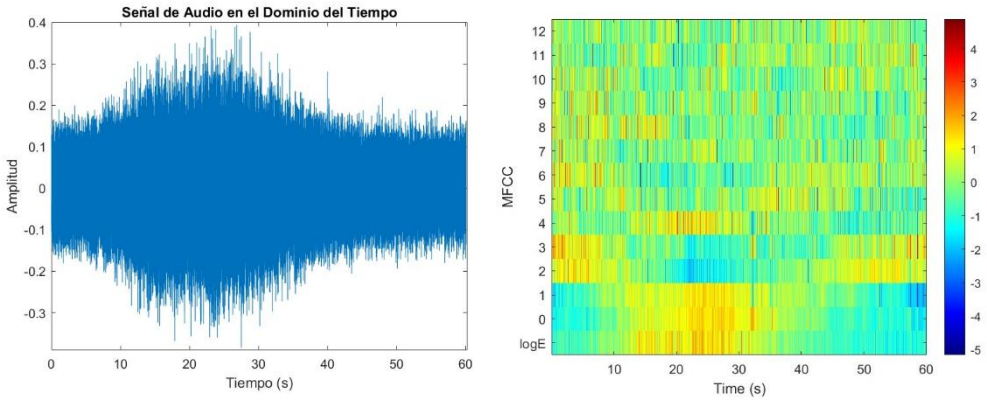

Figura 2.6. Señal de audio en el tiempo y los coeficientes MFCC obtenidos.

2.2.2 Delta MFCC

Se emplea para comprender como evoluciona el espectro de potencia de la señal a lo largo del tiempo, es decir, proporciona una idea de la trayectoria que siguen los coeficientes MFCC entre fotogramas.

Se calcula tomando la primera derivada de las características MFCC, que en la matemática discreta se aproxima teniendo en cuenta una ventana anterior y posterior de los valores contiguos a la muestra donde se quiere calcular la derivada [22]. Esto aparece en la fórmula 2.4

$$delta = \frac{\sum_{k=-M}^{M} k\, x(k)}{\sum_{k=-M}^{M} k^2} \tag{2.4}$$

2.2.3 Flujo espectral

El flujo espectral (*Spectral Flux*) es una medida que define la variación media del espectro en potencia de una señal. Se obtiene haciendo una comparación del espectro de potencia de una trama temporal de la señal con el espectro en potencia de la trama anterior.

Su calcula considerando la STFT de la señal, que proporciona las componentes espectrales en el dominio de la frecuencia. A partir de ellas se obtiene el *Spectral Flux* ya que mide la diferencia de energía de las componentes espectrales entre dos tramas consecutivas. Se calcula según la expresión 2.5

$$SF = \frac{1}{(N-1)(K-1)} \sum_{n=1}^{N-1} \sum_{k=1}^{K-1} [\log\,(A(n,k)+\delta) - \log\,(A(n-1,k)+\delta)]^2 \tag{2.5}$$

A(n,k) representa el valor de los coeficientes DTF de la trama n , se le suma un diminuto valor para evitar *overflow* y se calcula su logaritmo. Así se consigue la STFT. A este término se le resta el segundo término que representa la STFT del fotograma anterior A(n-1). Así hasta recorrer las N tramas de cada ventana. Figura 2.7.

Es una característica efectiva para distinguir entre sonidos del entorno, ya que su flujo espectral muestra valores elevados y experimenta cambios más notables en comparación con el habla [23].

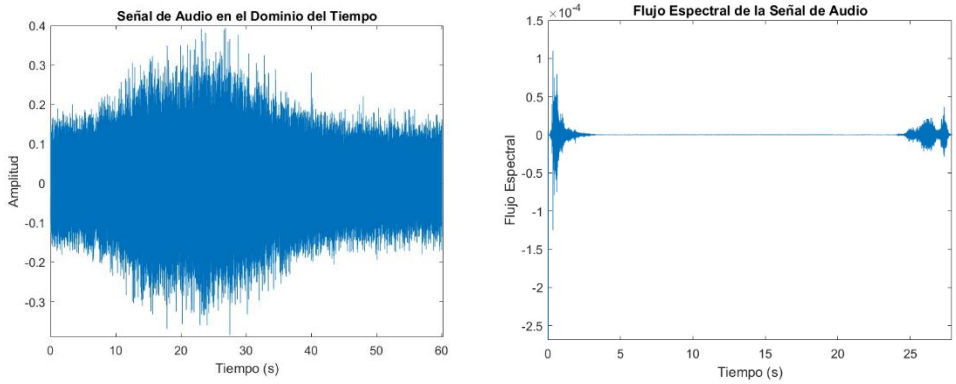

Figura 2.7. Señal de audio en el tiempo y flujo espectral.

2.2.4 Centroide espectral

El Centroide espectral (*Spectral Centroid*) indica donde se encuentra el centro de gravedad del espectro, es una propiedad empleada en el análisis de señales para describir las características de un espectro.

Para entenderlo podemos recurrir a la Figura 2.8. Cada instante de tiempo está representado por una serie de frecuencias y sus amplitudes, el centroide espectral indica en qué frecuencia se concentra el contenido espectral en ese instante de la señal.

Se calcula empleando la siguiente fórmula 2.6

$$SC_i = \frac{\sum_{m=0}^{N-1} f(m) X_i(m)}{\sum_{m=0}^{N-1} X_i(m)} \tag{2.6}$$

SCi es el centroide espectral de la trama i-ésima, f(m) representa la frecuencia central de una banda y X(m) su amplitud correspondiente, esto se suma y se divide por la suma total de las amplitudes [24].

Está fuertemente relacionado con el brillo del sonido. De manera que los valores de centroide más altos corresponden con rangos del espectro donde la energía está centrada en frecuencias más altas. Por el contrario, un valor más bajo de centroide indica que la energía está en frecuencias más bajas, sonidos más suaves o graves [21].

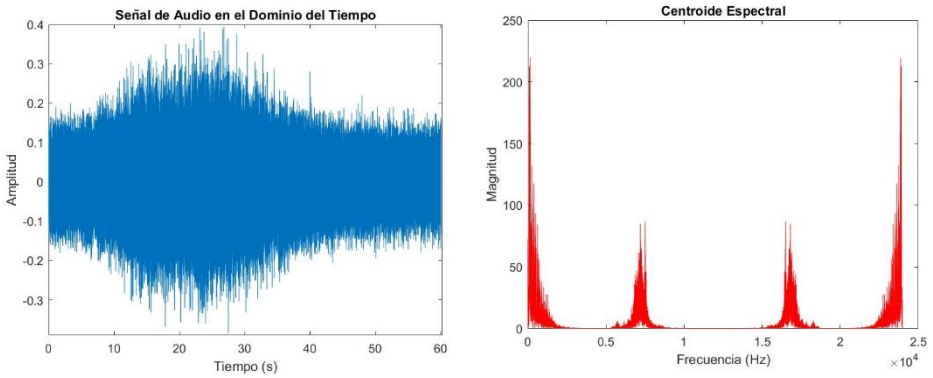

Figura 2.8. Señal de audio en el tiempo y centroide espectral.

2.2.5 Dispersión espectral

La dispersión espectral (*Spectral Spread*) se define como la concentración de energía alrededor del centroide [25]. Por tanto, se relaciona con la extensión en frecuencia, el ancho de banda, de la señal donde se concentra la energía.

Se puede calcular considerando la varianza de las frecuencias por sus amplitudes, como se observa en la fórmula 2.7

$$\sigma^2 = \int (x - \mu)^2 \, p(x) \; \delta x \tag{2.7}$$

Cuanto mayor sea la varianza, mayor será la dispersión espectral y por tanto más grande el rango de frecuencias por el que se distribuye la energía. Este sería el caso de sonidos brillantes. Cuando la dispersión espectral es baja corresponde con sonidos más suaves.

2.2.6 Asimetría espectral

La asimetría espectral (*Spectral Skewness*) proporciona información sobre la simetría que presenta el espectro alrededor del centroide [25].Es decir, mide la diferencia que hay entre el espectro que está por debajo y por encima de la frecuencia media, centroide. Se calcula con la fórmula 2.8

$$m_3 = \int (x-\mu)^3\, p(x)\, \delta x \qquad \gamma = \frac{m_3}{\sigma^3} \tag{2.8}$$

El espectro puede ser simétrico o asimétrico. Si la asimetría espectral presenta un valor negativo, la energía disminuye rápidamente si las frecuencias superan el centroide, pero se expande en un patrón más amplio hacia frecuencias más bajas. Si es positivo ocurre lo contrario. Figura 2.9.

$\gamma_1 = 0$ → Distribución simétrica

$\gamma_1 < 0$ → Hay más energía en la derecha

$\gamma_1 > 0$ →Hay más energía en la izquierda

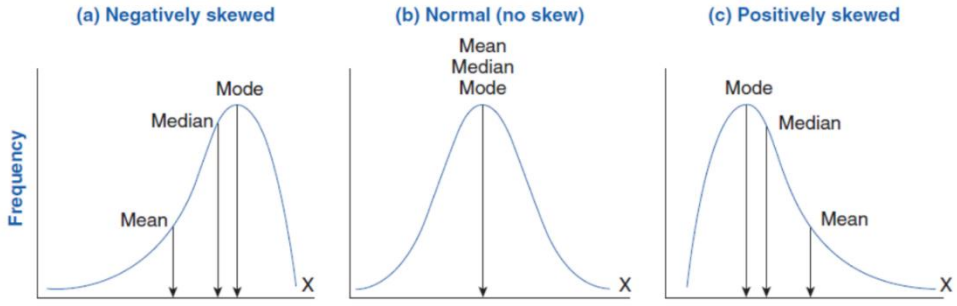

Figura 2.9. Representación de la asimetría espectral [26]

2.2.7 Curtosis espectral

La curtosis espectral (*spectral kurtosis*) indica si el espectro se parece a la forma de una curva de campana de Gauss, dicho de otro modo, describe el nivel de 'planicie' del espectro alrededor del centroide [25]. Sigue la fórmula 2.9, Figura 2.10.

$$m_4 = \int (x-\mu)^4\, p(x)\, \delta x \qquad \gamma_2 = \frac{m_4}{\sigma^4} \tag{2.9}$$

$\gamma_2 = 0$ → Comportamiento espectral gaussiano

$\gamma_2 > 0$ → Distribución espectral con pico

$\gamma_2 < 0$ → Distribución espectral uniforme, plana

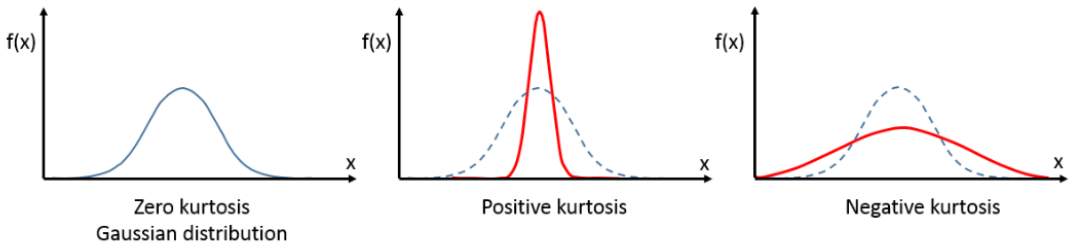

Figura 2.10. Representaciones de la curtosis espectral [27]

2.2.8 Roll-off espectral

El roll-off espectral (*Spectral Roll-off*) indica un valor determinado de frecuencia por debajo del cual se acumula un porcentaje de la energía total del espectro de la señal [28]. En este trabajo se utiliza el roll-off al 0.95 de probabilidad, esto indica que el valor de roll-off obtenido sería la frecuencia en la cual el 95% de la energía total de la señal se encuentra por debajo de ese punto [25]. Figura 2.11.

En la expresión 2.10 el termino fc es la frecuencia de roll-off obtenida al 95%.

$$\sum_{0}^{fc} a^2(f) = 0.95 \sum_{0}^{sr/2} a^2(f) \qquad (2.10)$$

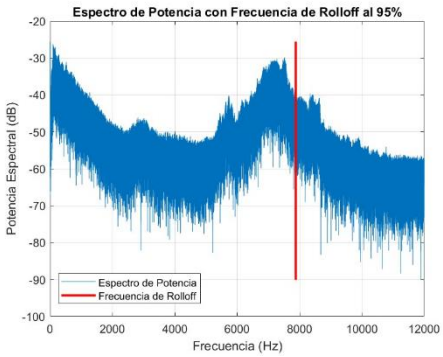

Figura 2.11. Representación del Roll-off espectral

2.3 Métodos de *clustering* propuestos

Este trabajo se centra en los métodos de ML no supervisados, también conocidos como técnicas de *clustering*. La idea básica es que los puntos de datos se pueden descomponer en subconjuntos o *clústeres*. Para llevar a cabo este proceso, es esencial que cada punto de datos pueda ser representado mediante un vector de características. Por lo tanto, como paso inicial, se ha de realizar un preprocesamiento de los datos, extrayendo de ellos una serie de características conocidas como "*features*". En este caso en particular se extraerán las características explicadas en el punto anterior 2.2

A partir de este punto, es posible comenzar con el agrupamiento de los datos, *clustering*. Este proceso implica desarrollar un modelo o hipótesis que examina las características de un punto de datos y generar una predicción sobre el índice del grupo al que pertenece ese punto de datos.

La técnica de *clustering* no requiere de etiquetas previas para los datos, realiza la agrupación basándose en la geometría intrínseca de los puntos de datos y buscando similitudes en sus características.

Existen dos tipos de *clustering*:

- *Soft clustering*: cada punto de datos se puede asignar a varios grupos, generando así un vector de índices. Cada índice tiene asociado un grado de pertenencia a ese grupo que toma valores entre {0,1}

$$\hat{y} = \hat{y}, \hat{y}^2, \dots, \hat{y}^k \tag{2.11}$$

- *Hard clustering*: cada punto de datos es asignado a un solo grupo. El algoritmo trata de predecir dicho agrupamiento en función de sus *features*.

$$\hat{y} = \mathrm{h}(\mathrm{x}) \in \{1, \dots, k\} \tag{2.12}$$

x: la muestra de datos y^: indica de clústeres precedido

A continuación, procedo a describir las principales técnicas de *clustering*.

2.3.1 K-means

K-means es el algoritmo de *Machine leaning* no supervisado más común y conocido utilizado para la agrupación de datos en clústeres o grupos basados en similitudes en sus características. Esta técnica se considera un método de *Hard clutering*, por tanto, cada punto de datos solo pertenece a un único clúster [1].

El algoritmo *k-means* consiste en agrupar las muestras de datos por clases según las similitudes entre ellas. Si las muestras de datos están cercanas entre sí, es más probable que compartan características similares y por ello pertenezcan al mismo grupo. Cada clúster tiene asociado un centroide, que son el punto medio de los objetos en los clústeres.

K-means funciona de la siguiente manera. Se comienza con una base de datos D formada por muestras, previamente se extraen una serie de característica sobre cada una de ellas, por tanto, cada una será identificada por su vector de características. El objetivo es realizar predicciones de

las asignaciones de las muestras basándose en la geometría intrínseca de los vectores de características [4], como se muestra en la figura 2.13.

$$\hat{y} = h(x) \tag{2.13}$$

$\hat{y}$ es el índice del clúster al que pertenece la muestra de datos x.

El procedimiento que sigue este método es:

1. Se inicia escogiendo el número de clústeres, k.
   El número de grupos depende del contenido de la base de datos, aunque también existen algoritmos que calculan este valor, como el "método del codo".

2. Seleccionar el centroide inicial de cada clúster, esto puede hacerse con vectores aleatorios cuya distribución de probabilidad se ajusta a los datos o forma manual.

3. Asignar cada muestra de datos a un clúster.
   Cada muestra se asigna al centroide más cercano a él, para determinar la cercanía entre ellos se necesita algún tipo de medida de proximidad, en este caso se utilizará la Distancia Euclidiana.
   La distancia euclidiana se calcula empleando ecuación 2.14.

$$dist\big((x,y)(a,b)\big) = \sqrt{(x-a)^2 + (y-b)^2} \tag{2.14}$$

   Esta fórmula es sencilla de interpretar, como datos de entrada se utilizan las coordenadas de la muestra que queremos clasificar y las coordenadas del centroide de un grupo. Se prueba con el centroide de cada agrupación y finalmente el punto se asigna al clúster que minimiza este cálculo.

4. Actualizar los centroides de cada clúster.
   Una vez asignados los puntos a los centroides de todos lo clústeres, el siguiente paso es ajustar los centroides. Para ello se emplea la fórmula 2.15.

$$\mu^{(c)} := \frac{1}{|\{i: \hat{y}^{(i)} = c\}|} \sum_{i:\hat{y}^{(i)}=c} x^{(i)} \tag{2.15}$$

   Se procede a calcular el nuevo centro del conjunto, tomando en consideración el grupo y la totalidad de las muestras adscritas al mismo.
5. Computar la Suma de Errores Cuadrados (*Sum of Squared Error,* SSE)
   El objetivo de la función es minimizar la distancia al cuadrado entre el centroide y cada muestra del clúster. La fórmula 2.16 muestra el cálculo de la SSE, primero encuentra la Distancia Euclidiana de cada muestra con el centroide del clúster al que pertenece y luego realiza la suma de este valor recorriendo todos los grupos.

$$SSE = \sum_{j=1}^{n} \sum_{x \in C_i} dist(C_i, x)^2 \tag{2.16}$$

Para lograr una predicción óptima para cada muestra de datos, es necesario repetir el proceso, que comprende los pasos 3, 4 y 5. Este ciclo se repite (Figura 2.12) hasta que los centroides de los grupos se mantengan estables o hasta que las muestras no sean reasignadas a otras agrupaciones. En esencia, en cada iteración, el objetivo es minimizar la SSE y alcanzar la asignación más precisa y consistente de las muestras a sus respectivos grupos.

Figura 2.12. Representa el procedimiento de *k-means*

En ciertas situaciones, el algoritmo puede llevar a centroides que resulten en un mínimo local en lugar de un mínimo global en el cálculo de la suma de errores cuadrados, lo que podría resultar en una clasificación menos precisa. Esto está relacionado con los valores iniciales asumidos por los centroides. Por tanto, para evitarlo se suele repetir el algoritmo de *k-mean* varias veces.

2.3.2 *K-medoids*

Es un algoritmo de *Machine learning* utilizado para agrupar datos en clústeres. Es similar a *k-means*, sin embargo; en este método se utilizan medoides en vez de centroides.

Los medoides son muestras reales de datos dentro de un clúster consideradas las más representativas de ese clúster. En otras palabras, el medoide es la muestra que está más cerca de todas las demás muestras del mismo. (Figura 2.13.) Esto hace que sea diseñado parar tratar la sensibilidad de datos atípicos [29].

Hay tres tipos de algoritmo para *k-medoids clustering*:

- PAM (particionamiento en torno a la agrupación)
- CLARA (*clustering* de grandes aplicaciones)
- CLARANS (agrupación aleatoria de grandes aplicaciones)

En este trabajo se utiliza el algoritmo PAM, este tiene el siguiente funcionamiento:

1. Elegir el número de grupos, k y con ello k número de muestras aleatorias que representarán los medoides iniciales.

2. Asignar cada muestra de datos al clúster más cercano.
   Para ello se utilizan fórmulas de distancia como la distancia euclidiana o la distancia de Manhattan.

3. Calcular coste total.
   Este proceso implica calcular la suma de las distancias entre una muestra y su medoide correspondiente dentro del mismo grupo.

4. Elegir una nueva muestra aleatorio como medoide.
   Con este fin, se realiza nuevamente el cálculo del coste total y si el resultado es inferior al anterior se sustituye el medoide por el calculado. Por el contrario, si es superior se deja el medoide previo.

El paso 4 se repetirá hasta que se encuentre el medoide más idóneo.

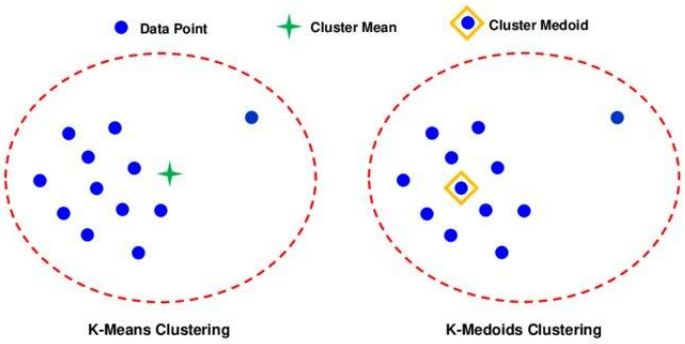

Figura 2.13. Ejemplo que compara la clasificación empleando *k-mean y k-medoids*[30]

2.3.3 Gaussian Mixture Models

Los modelos de mezcla gaussiana (*Gaussian Mixture Model*, GMM) son un algoritmo de agrupamiento que sigue un modelo probabilístico para modelar conjuntos de datos mediante la combinación de varias distribuciones Gaussianas [31].

Este modelo se clasifica como método de *soft clustering* de manera que cada muestra de datos no se asigna exclusivamente a un solo grupo, sino que su pertenencia a varios grupos se determina mediante una medida de probabilidad basada en su grado de afinidad con cada grupo. Este método es muy útil encontrando patrones cuando la distribución de muestras no es claramente separable en distintos grupos. Véase la Figura 2.14 donde una clasificación empleando *k-means* sería menos precisa.

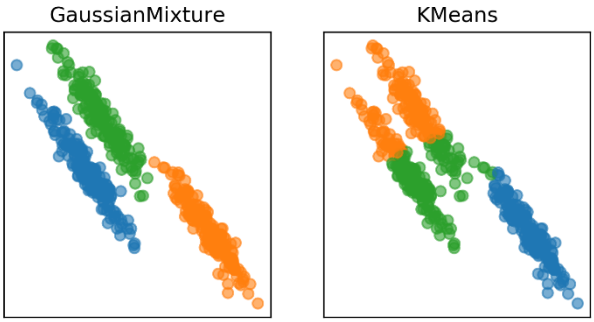

Figura 2.14. Ejemplo que compara la clasificación empleando *el modelo gaussiano y k-medoids* [32]**.**

Una distribución gaussiana en un espacio unidimensional, se caracteriza por su media, μ, y varianza, σ , sin embargo; los modelos de mezcla gaussiana trabajan con espacios multidimensionales [4]. Por tanto, los GMM asumen que cada clúster está representado por una distribución normal multivariante y presenta un vector de medias (μ), una matriz de covarianza (ε) y la probabilidad de este clúster (p). La expresión 2.17. Es la función de densidad de probabilidad de un clúster.

$$f(x|\mu,\varepsilon) = \frac{1}{\sqrt{2\pi|\sigma|}} e^{-\frac{(x-\mu)^t \sum (x-\mu)}{2}} \quad (2.17)$$

El grado de pertenencia de una muestra a un grupo depende de la geometría de la base de datos. Cada muestra tiene asociado un vector $\hat{y}$ de probabilidades de pertenencia. Los valores de este vector están relacionados con los GMM parámetros $\{\mu, \varepsilon, \hat{p}\}$ que son calculados a partir del algoritmo de esperanza-maximización (*Expectation Maximization*, EM).

EM se define de la siguiente forma:

1. Preparar la base de datos, D, el número de clústeres, k

2. Asignar valores iniciales a los 3 parámetros $\{\mu, \varepsilon, \hat{p}\}$

3. Actualizar el vector $\hat{y}$ es decir, los grados de pertenencia.
   Para ello se emplea la siguiente fórmula 2.18. $f(x; \mu, \varepsilon)$ es el modelo probabilístico y $\hat{p}$ es la probabilidad del clúster c.

$$\hat{y}_c^{(i)} := \frac{\hat{p}_c \, f(x^{(i)}; \hat{\mu}^{(c)}, \hat{\varepsilon}^{(c)})}{\sum_{c'=1}^{k} \hat{p}_{c'} \, f(x^{(i)}; \hat{\mu}^{(c')}, \hat{\varepsilon}^{(c')})} \quad (2.18)$$

4. Actualizar lo GMM parámetros $\{\mu, \varepsilon, \hat{p}_c\}$

El paso 3 y 4 se repiten hasta que se cumple el criterio asignado.

Al igual que en los anteriores algoritmos se recomienda repetir el algoritmo varias veces para evitar caer en mínimos locales.

2.4 Otros métodos de *clustering*

2.4.1 *Hierarchical clustering*

Hierarchical clustering también conocido como agrupamiento jerárquico, es otro enfoque en el análisis de datos. Se basa en un algoritmo de *machine learning* que tiene como objetivo dividir un conjunto de datos en clústeres de manera jerárquica y estructurada [33]. Véase Figura 2.15.

Los clústeres se representan en un árbol jerárquico llamado dendrograma que muestra cómo los grupos se dividen y fusionan.

Existen dos métodos de agrupamiento jerárquico [1]:

- Aglomerativo: este enfoque comienza considerando cada muestra como un clúster individual. Posteriormente estos grupos se van fusionando entre sí en función de sus similitudes hasta que se alcanza el número de grupos deseados o hasta que solo queda un clúster.

- Divisivo: esta técnica comienza agrupando todas las muestras en un único clúster. Luego, gradualmente, se procede a dividir esté en clústeres más pequeños en función de las similitudes que hay entre los puntos. Este proceso continúa hasta que se alcanza el número de grupos deseados o hasta que ya no sea posible realizar más divisiones.

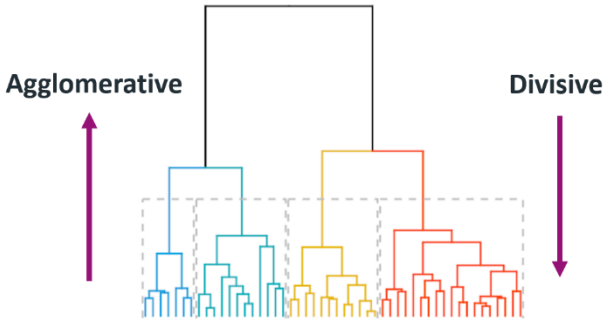

Figura 2.15. Clasificación empleando *Hierarchical clustering* [34]

Es importante destacar que, a diferencia de las técnicas de agrupamiento mencionadas anteriormente, el agrupamiento jerárquico, no requiere la especificación previa del número de clústeres que dividirán los datos.

Para medir la similitud entre los datos y unir o separar grupos, se utilizan medidas de proximidad entre clústeres. La técnica a utilizar dependerá del tipo de agrupamiento. Entre estas, encontramos " *Furthest Neighbor Method*", que mide la distancia entre las muestras más alejadas de diferentes clústeres y " *Nearest Neighbor Method*" que mide la distancia entre las muestras más cercanas. Además "*Average Linkage Method*", que mide la distancia entre cada par de objetos de ambos clústeres y el "Centroid Method" que calcula la distancia entre los centroides de diferentes clústeres. Estas medidas son necesarias para determinar cómo los clústeres se fusionan o dividen a lo largo de las etapas del agrupamiento jerárquico.

2.4.2 DBSCAN clustering algorithm

El algoritmo de clusterización basado en la densidad (*Density-based spatial clustering*, DBSCAN) es otra técnica de agrupamiento, pero menos conocida que la comentadas anteriormente [1]. Se base en medir la densidad de muestras en el espacio de características.

Una de sus ventajas es que logra descubrir agrupaciones de tamaños irregulares y que es útil para trabajar con ruido y valores atípicos.

En DBSCAN hay que tener en cuenta dos parámetros; MinPts que indica el número mínimo de muestras que tiene una región para ser considerada densa y que conforme un clúster y Eps que indica el radio del clúster.

Hay 3 tipos de muestras:

- *Core points*: son los centros de los grupos. Una muestra será considerada centro si hay al menos MinPts muestras dentro de una distancia de Eps a su alrededor.
- *Border points*: son las muestras que no tiene MinPts muestras dentro de su radio Eps pero tiene al menos una muestra.
- *Noise points*: aquellas que no tienen ningúna muestra cerca.

El algoritmo funciona de la siguiente manera:

Tras determinar los parámetros, hay que identificar las muestras consideradas centros y definir un grupo alrededor de cada una de ellas, utilizando un radio Eps. A continuación, se expanden los clústeres de la siguiente manera: cada muestra dentro del círculo alrededor de una muestra central se incorpora al clúster correspondiente. Si alguna de ellas es una muestra central los clústeres se fusionan. Además, aquellas muestras que no estén dentro de ningún grupo serán consideradas ruido. Este proceso se observa en la Figura 2.16.

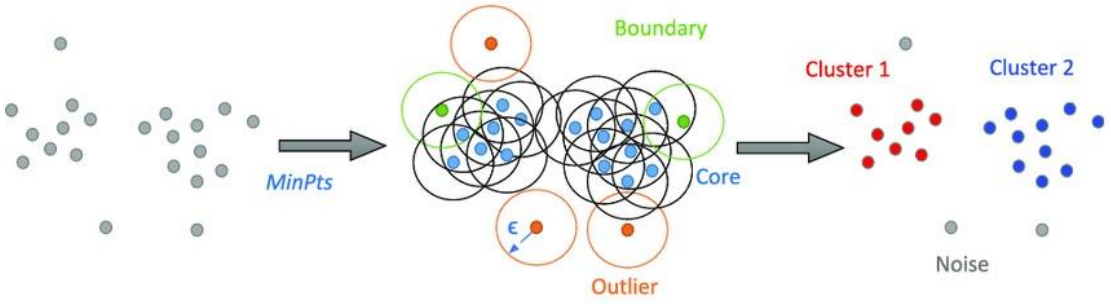

Figura 2.16. Proceso de clasificación empleando *Density-based spatial clustering* [35]

2.5 Descripción funciones de *clustering* en Matlab

En este trabajo, empleo la plataforma de programación Matlab para desarrollar un código que lleva a cabo la clasificación de los archivos de audio.

A continuación, proporcionaré una descripción superficial sobre las líneas de código utilizadas para implementar las técnicas de clasificación descritas previamente.

k-Means clustering:

```
[idx,C] = kmeans(X,k)
```

X → matriz formada por los vectores de características, las filas corresponden observaciones y las columnas corresponden a variables.

k → el número de clústeres.

`idx` → vector contiene los índices del grupo al que pertenece cada observación tras la clasificación.

`C` → matriz con la ubicación de los centroides de cada grupo.

Adicionalmente se pueden añadir parámetros extra que afectaran a la clasificación para ello se utiliza la siguiente instrucción → `Name,Value`

En el próximo capitulo profundizaremos en su uso.

k-medoids clustering:

```
[idx,C] = kmedoids(X,k,Name,Value)
```

Es muy similar al código anterior, la diferencia está en que C indica los medoides.

`Name,Value` → este parámetro servirán para indicar que tipo de algoritmos se empleará: PAM, CLARA, CLARANS.

Hierarchical clustering:

```
Z = linkage(X,method,metric)
dendrogram(Z)
grp = cluster(Z,"Cutoff",C)
```

`linkage` → se utiliza para crear un árbol jerárquico, con estos parámetros de entrada:

- X → matriz formada por los vectores de características.
- `method` → es el método empleado para calcular la distancia entre clústeres: "single".
- `metric` → indica el método para calcular la distancia entre puntos: "euclidean".

`dendrogram` → permite dibujar el dendrograma obtenido.

`cluster` → sirve para definir grupos a partir de un árbol aglomerativo de grupos jerárquicos.

- `Cutoff,C` → umbral para definir grupos: "maxclust",3.

Density-based spatial clustering:

```
idx = dbscan(X,epsilon,minpts)
```

`dbscan` → sirve para realizar una agrupación espacial basada en la densidad.

- `X` → matriz formada por los vectores de características.
- `epsilon` → indicar el radio del clúster.
- `minpts` → indicar el mínimo número de puntos dentro de un clúster.

Gaussian mixture models:

```
gm = fitgmdist(X,k)
[ind,nlogP,P] = cluster(gm,X)
```

`fitgmdist` → se emplea para crear un modelo de mezcla gaussiano proporcionando como entrada los datos y el número de clústeres.

- `gm` → es el modelo de mezcla gaussiana adaptado a los datos.

`cluster` → sirve para construir clúster a partir de una distribución de mezcla gaussiana.

- ind → vector que contiene los índices del grupo al que tiene mayor probabilidad de pertenecer cada observación tras la clasificación.
- P → matriz formada por vectores que indican la probabilidad que tienen cada observación de pertenecer a cada clúster.

2.6 Análisis de Componentes Principales

El Análisis de Componentes Principales (*principal component analysis*, PCA) es un método de reducción de dimensionalidad empleado para el análisis de datos. Esta técnica permite reducir el conjunto de datos mientras se conserva la mayor variabilidad de estos, es decir, consigue resumir la información contenida en los datos y seleccionar la más importante [36].

El PCA forma unos nuevos vectores de características llamados componentes principales creados mediante una combinación lineal de las características originales. Este enfoque permite reducir la dimensionalidad de los datos filtrando la información e identificando que características son más relevantes para el posterior proceso de agrupación de los puntos de datos.

Este análisis se realiza siguiente el siguiente procedimiento [4]:

Objetivo: obtener un conjunto de características relevantes a partir del total de características 2.19

$$x = (x_1, \dots, x_n)^T \tag{2.19}$$

1. Estandarizar las características de los datos para que todas contribuyan por igual al análisis

2. Calcular de la matriz de covarianza.
   Esta matriz permite ver qué relación existe entre las características del conjunto de datos. Busca que variables están correladas y por tanto contienen información redundante.

3. Calcular los valores y vectores propios.
   Los vectores propios presentan la varianza de los datos en diferentes direcciones y lo valores propios de cada vector propio indican la cantidad de variabilidad de los datos en esa dirección. Se obtienen a partir de la matriz de covarianza de la siguiente forma:

$$Q = (u^{(1)}, \dots, u^{(d)}) \begin{pmatrix} \lambda^{(1)} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda^{(d)} \end{pmatrix} (u^{(1)}, \dots, u^{(d)})^T \tag{2.20}$$

   $u =$ vectores propios ortonormales y $\lambda =$ valores propios
   Los vectores propios conforman la matriz de compresión W_{pca} . Fórmula 2.21

$$W_{pca} := (u^{(1)}, \dots, u^{(n)})^T \tag{2.21}$$

4. Generar el nuevo vector de características.
   Se obtiene aplicando la matriz de compresión W_{pca} al vector de características z. El vector x contiene las Componentes Principales (PC). Fórmula 2.22

$$x = W_{pca} \times z \tag{2.22}$$

Las componentes principales no están correladas entre sí y por tanto, son ortogonales entre ellas.

Las CP capturan la máxima variación en los datos, es decir la mayor cantidad de información posible. Para entenderlo observar la Figura 2.17 donde aparecen representadas la 1º y 2º componente principal y las muestras de datos, se observa como estas componentes forman unos nuevos ejes que buscan unir la mayor cantidad de muestras posible. Cuantas más muestras tenga ese eje, mayor cantidad de información conseguida y mayor variación encontrada.

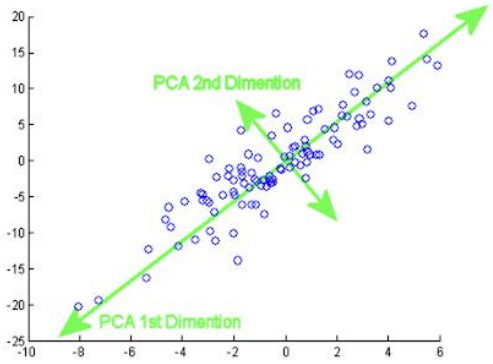

Figura 2.17. Ejemplo de PCA con dos componentes principales [37]

Cabe tener en cuenta que la primera componente principal concentra la mayor parte de la información sobre los datos, la segunda componente la información restante y así sucesivamente.

Esta técnica hace que los algoritmos de clasificación sean más eficientes y conseguir mejores resultados ya que reduce la redundancia y correlación entre las características.

Capítulo 3.

Resultados

3.1 Bases de datos

3.1.1 Área de estudio

El Parque Natural de la Albufera se encuentra en la costa del Golfo de Valencia en el este de España y tiene una superficie de 21,120 ha. El parque tiene una laguna poco profunda con una extensión de 23,94km^2, está rodeada de 223km^2 de arrozales y separada del mar por una estrecha barra litoral arenosa, constituida por dunas. La Albufera constituye un humedal costero de gran valor para el medio ambiente, pues en el habitan especies en peligro de extinción, como el fartet y el samaruc. Fue declarado Parque Natural en 1986 y desde 1989 está reconocido como "Humedal de Importancia Internacional" por la Convención sobre los Humedales. También es integrante de la Red Natura 2000 y considerada aérea ZEPA (Zona de Especial Protección para las Aves) [38]. Esto lo convierte en uno de los humedales más valiosos de la Comunidad Valenciana y de España.

Las dunas de la Devesa del Parque Natural constituyen un paraje protegido a las orillas del Mediterráneo, siendo uno de los pocos de su tipo. Este entorno crea un hábitat natural que alberga una rica diversidad de flora y fauna. El parque es conocido por su riqueza de especies de aves, entre las que destacan el pato colorado, la cuchara común y la garza. Cuenta con un total de 80000 aves acuáticas de media al año. Sin embargo, existen 45 especies animal en peligro de extinción muchas de ellas aves acuáticas.

El proyecto que se está llevando a cabo en la Albufera se denomina Proyecto Daphne y ha sido financiado por la Junta de Andalucía, participando en él la Universidad de Sevilla y la Universitat Politècnica de València. Su objetivo es estudiar la biodiversidad existente en la Albufera mediante la clasificación automática de los sonidos utilizando algoritmos de inteligencia artificial, concretamente de aprendizaje profundo.

3.1.2 Técnicas de grabación

Las grabaciones empleadas para crear la base de datos se han recogido a través de diez nodos acústicos colocados en la zona de la laguna como se muestra en la Figura 3.1. Cinco de ellos se instalaron en cuatro islas dentro de la laguna (Mata de l'Antina, Mata de Sac Roc (tiene dos nodos), La Maseguerota y Mata del Fang). Los otros cinco nodos se instalaron en los terrenos que rodean la laguna: dos nodos en el Tancat de Milia, otros dos nodos en La Tancadeta y un nodo en la oficina técnica del parque en El Palmar. Estos nodos acústicos son dispositivos comerciales de la empresa *Wildlife Acoustics* y únicamente tiene acceso a ellos personal autorizado. Aquellas localizaciones que cuentan con dos nodos, estos están separados por 80 metros el uno del otro. Un parámetro relevante para el análisis de los audios recogido es la frecuencia de muestreo con la que trabajan los nodos en este caso es de Fs=24000 Hz. [39]

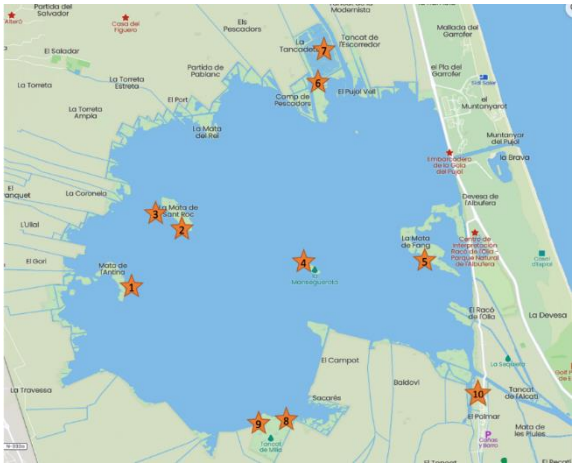

Figura 3.1. Localización de los nodos en el Parque Natural de la Albufera [39]

El objetivo de la instalación es estudiar el comportamiento de las aves en el entorno de la Albufera, por ello estos nodos fueron programados para grabar antes y después del amanecer y el atardecer, períodos del día en los cuales las aves tienden a mostrar una mayor actividad. Sin embargo, no solo registran el sonido de las aves, sino también el paisaje sonoro que les acompaña. Es por ello que los audios presentan ruidos de aviones, lanchas o conversaciones.

Actualmente disponemos de grabaciones desde junio de 2022 hasta junio de 2023.Están almacenadas en archivos de audio .wav que tienen una duración de 60 min cada uno.

3.1.3 *Dataset* Inicial

Las grabaciones de audio recopiladas por los nodos contienen un amplio conjunto de sonidos naturales como el canto de pájaros, ruidos de animales que se mueven cerca de los nodos, ruido de viento o lluvia y otros ruidos de origen humano como el de los aviones debido a la cercanía del aeropuerto de Valencia, el sonido de lanchas que cruzan la laguna, conversaciones o incluso algún coche en las matas cercanas a la carretera.

Este trabajo se enfoca en detectar y clasificar aquellos ruidos de origen humano para así poder, posteriormente, limpiar las grabaciones de ellos. Se construye un conjunto de datos utilizando las grabaciones de los dos nodos de La Tancadeta, el nodo de la Mata del Fang, el nodo de El Palmar, el nodo de la Mata del l'Antina y los dos nodos de la Mata de Sant Roc. Se eligen estos porque están situados cerca de áreas donde hay actividad humana, principalmente en el nodo colocado en la oficina.

Para generar la base de datos se seleccionan aquellos audios que considero tiene ruidos de interés para este trabajo. Se escogen audios grabados entre el 3 de junio de 2022 y el 2 de junio de 2023, centrándome en las grabaciones de la mañana entre las 6:00h y las 12:00h. Realizo esta elección porque de madrugada hay mayor cantidad de vuelos y a partir de las 10 se producen visitas escolares a la laguna.

El conjunto de datos se crea de la siguiente manera, primero escojo audios de 60 min comprendidos en las franjas horarias explicadas. A continuación, empleo el programa *Audacity*, un editor de audio, para observar la señal en tiempo, poder detectar y recortar fragmentos de la señal que contengan algún audio de mi interés. Me centro en recopilar audios que contengan: voces humanas, ruido de aviones, ruido de lanchas y sonido de fondo. Con ellas formo tramas de

audio con una duración de 1 min y las almaceno en una carpeta para posteriormente utilizarlas en el código de Matlab.

El dataset inicial contienen 6 grabaciones con una duración total de 6 minutos, como se muestra en la siguiente tabla

Contenido del audio	*Número de audios*	*Tiempo total*	*Número frames*
Ruido de aviones	2	2 min	40
Voces humanas	2	2 min	40
Ruido de lanchas	2	2 min	40
Número total de frames: 120			

Tabla 3.1. Resumen de la base de datos inicial

3.2 Clasificación mediante *k-means*

En este apartado, se encuentra una descripción detallada de cómo utilizo los diversos métodos de clasificación, las pruebas que realizo y un análisis de los resultados obtenidos.

Antes de proceder con el proceso de clasificación, es necesario realizar la extracción de características y la preparación de la matriz que contendrá los datos pertinentes.

Para llevar a cabo esta tarea, he desarrollado un código de Matlab que accede a los archivos de la base de datos y emplea una serie de funciones para extraer de manera adecuada las características de los audios y organizarlas en la matriz correspondiente. Las características para capturar en los audios son las siguientes:

Espectro de Mel	Flujo espectral	Asimetría espectral
MFCC	Centroide espectral	Curtosis espectral
Delta MFCC	Dispersión espectral	Roll-off espectral

Tabla 3.2. Características extraídas de los audios

Elijo estas porque proporcionan información espectral de la señal que puede resultar relevante para la clasificación. Se puede encontrar una descripción más detallada de ellas en el apartado 2.2.

El estudio de los archivos de audios se realiza mediante la segmentación en fragmentos y análisis individual de cada una de estas tramas. Esto conduce a la obtención de un vector de características para cada trama o *frame*, lo que resulta en una matriz de características que corresponde a todos los audios.

Esta matriz se convierte en la entrada al clasificador. En ella están representadas las propiedades tanto acústicas como estructurales de los audios. Estos datos se emplean con el fin de ejecutar la distribución y agrupación de las observaciones, permitiendo a los algoritmos de clasificación identificar patrones.

En el anexo II se encuentra el código Matlab empleado para la extracción de características.

3.2.1 Pruebas exploratorias

Análisis histograma de las características

Para iniciar el proceso, entre todas las características extraídas, traté de identificar aquellas que podrían tener más influencia en la clasificación. Para lograrlo, generé un histograma individual para cada característica, extrayendo los valores de la matriz que entra al clasificador. Aunque había aplicado previamente una normalización a las características, al examinar los histogramas noté que algunas presentaban valores que se alejaban del centro. Esto es evidente en la Figura 3.2 donde aparece una gran concentración de observaciones entre los valores 2-2.5. Estos valores más alejados tienden a ser más discriminativos y pueden dar lugar a la formación de un único clúster. Mientras que los valores concentrados alrededor del cero pueden ser más difíciles de clasificar, dado que su diferenciación es menos evidente.

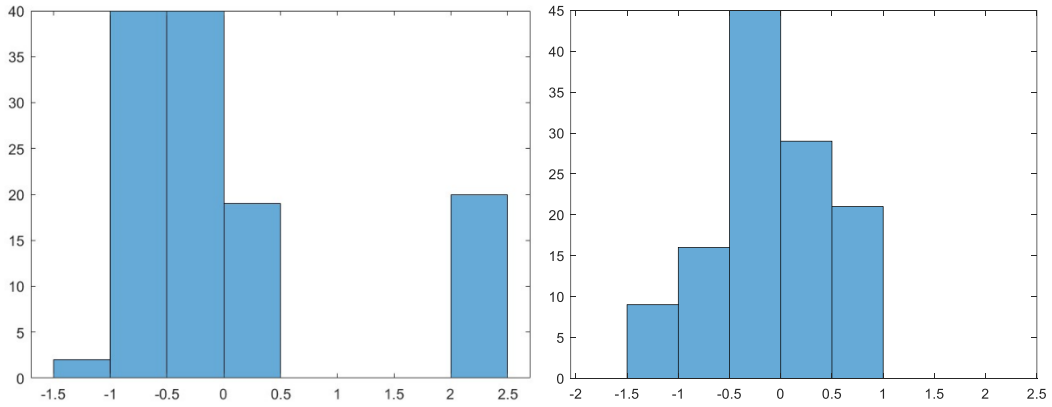

Figura 3.2. Media del Roll-off espectral y Media del MFCC

Aplicación del algoritmo *K-means*

Procedo a realizar la clasificación empleando el algoritmo de *k-means.* Para ello emplearé la función de Matlab *kmeans()* esto se puede observar en el anexo II. Esta función devuelve como resultado un vector que contiene el índice del grupo o clúster al que pertenece cada observación y las coordenadas del centroide de cada grupo.

En primer lugar, tuve que decidir qué valor de k, el decir, el número de agrupaciones en las que se dividirían los datos. La elección de este parámetro afecta en gran medida a los resultados de la clasificación y tiene que adaptarse al conjunto de datos de entrada. Este trabajo utiliza como *dataset* inicial un total de 6 audios de tres clases diferentes:

- Ruido de aviones, ruido de lanchas y voces de niños

Esto descarta el valor de k=1 y k=2. Tampoco me interesa utilizar k=3 y k=4 puesto que obtendría una agrupación muy ajustada y por tanto no permitiría que el clasificador por sí solo fuera capaz

de descubrir y profundizar en los matices de estos sonidos o de capturar otros que aparecen presentes en los audios como el ruido de fondo.

A continuación, procedo a hacer un análisis de como resultaría la clasificación empleando diferentes números de grupos, k=5, k=6, k=7 y k=8. Al mismo tiempo analizo que consecuencias tendría añadir o eliminar características.

Comienzo, por estudiar qué resultado se obtendrían utilizando k=5 clases diferentes. Esta elección busca detectar matices presentes en cada audio, ya que estos están formados por superposiciones de varios sonidos. Para lograrlo, realizo tres pruebas diferentes, modificando la matriz de características que entra al clasificador. De las agrupaciones obtenidas en ambas 3 pruebas deduje lo siguiente:

En la 1º prueba donde están presentes todas las características, se observa que cada audio es agrupado en un grupo distinto, pero no logra identificar relaciones entre los audios que comparten el mismo tipo de sonido por ejemplo los dos audios de lanchas.

En las 2º prueba al quitar algunas variables, las más influyentes ganan más peso en la clasificación y sirven para detectar matices como por ejemplo el avión cuando se acerca y se aleja, también es capaz de detectar el traqueteo de la lancha y agrupa los dos audios de lancha en el mismo grupo, Figura 3.3 (los puntos de la clase 3 (color rojo) de la imagen de la derecha, mientras que en la otra imagen lo divide entre puntos de la clase 2 (verdes) y la clase 4 (lilas)).

En la 3º prueba se elimina la variable MFCC Delta porque se observa que hay una alta correlación entre MFCC y MFCC Delta, esperando que no afecte a la clasificación y se obtengan los resultados de la 1º prueba. Sin. Embargo ocurre algo muy similar que con la 3º prueba.

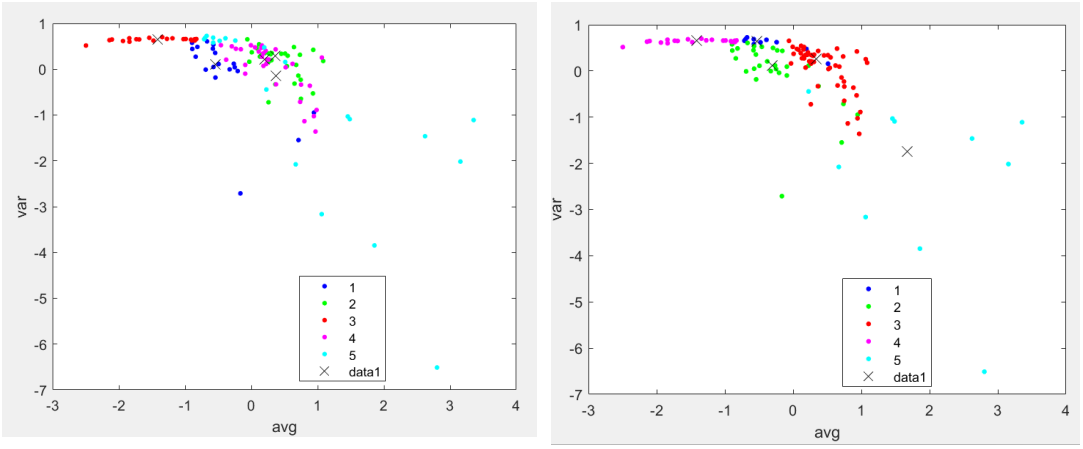

Figura 3.3. Característica asimetría espectral de la 1º y 2º prueba

Todo esto me lleva a considerar que k=5 trabaja con grandes rasgos, es decir, si se busca detectar matices en los audios como los gritos estridentes de los niños y cuando estos se alejan detecta el ruido de fondo o la presencia de voces en los audios de las lachas es recomendable utilizar un valor de k más elevado que permita crear agrupaciones para estos matices más precisos.

Procedo a hacer el análisis utilizando k=6 y k=7. Para ello realizo 4 pruebas diferentes donde modifico la matriz de características que entra al clasificador.

- 1º prueba: la matriz contiene todas las características.
- 2º prueba: elimino las características flujo espectral y centroide espectral. De esta manera elimino dos características que parecen poco influyentes.
- 3º prueba: elimino flujo espectral y curtosis espectral*:* pues deduzco que son las características menos influyentes.
- 4º prueba: elimino asimetría espectral y espectral Roll-off. Opté por eliminar las características más influyentes para verificar si la clasificación se deterioraba significativamente.

También he analizado los resultados obtenidos mediante graficas 2D sobre cada característica para así intentar deducir cuál de ellas parece tener mayor peso en las agrupaciones.

1º prueba: parece que realizan un análisis correcto de los sonidos de lanchas y los audios de aviones, incluso son capaces de detectar cuando el avión se acerca y se aleja pues esto lo agrupan en distintas clases. Sin embargo, en el caso de los audios de los niños, encuentra dificultades en la diferenciación. Identifican los gritos cuando están cerca del micrófono, pero al alejarse, comienzan a percibir el ruido de fondo, lo que impide mantener una distinción clara. Aunque k=7, detecta la presencia de voces de niños como sonidos de fondo en los audios de lanchas.

2º prueba: Con k=7, el análisis de los aviones y las lanchas es similar al anterior. Mientras, en el caso de los audios de niños es capaz de detectar las voces en ambos audios además, de apreciar un poco de ruido de fondo. En Figura 3.4 vemos como la clasificación varia y se observa una mayor presencia de puntos de la clase 7 (negros) en la imagen de la derecha que representan las voces de los niños.

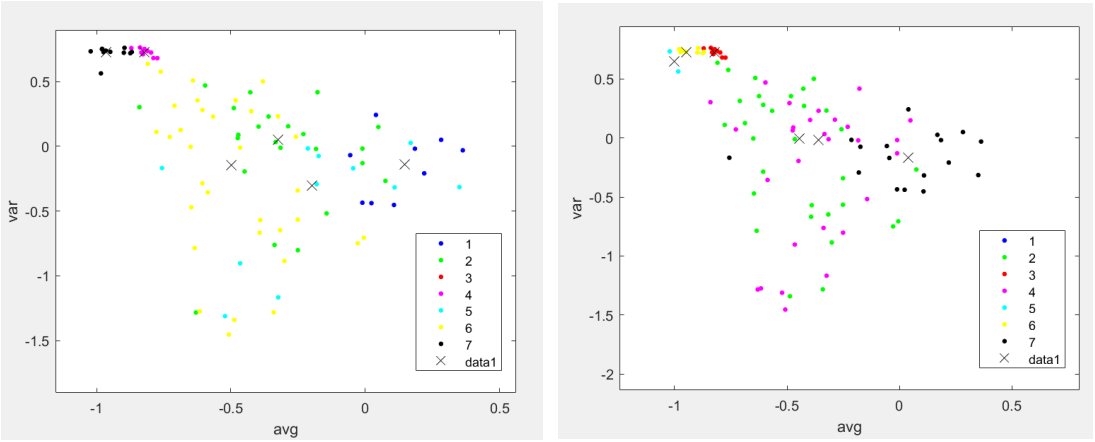

Figura 3.4. 1º prueba y 2º prueba

Con k=6, eliminar esas dos características afecta mucho a la clasificación contrariamente a lo esperado y las agrupaciones resultantes son erróneas.

3º prueba: eliminar estas dos características bastante robustas afecta directamente a la detección de las voces de los niños.

4º prueba: tanto k=6 como k=7 no detectan el efecto de cuando un avión se acerca y se aleja. En la imagen de la derecha hay un único grupo de puntos de la clase 1 (azules), mientras que en la otra imagen esos puntos pertenecen a dos clases diferentes. Figura 3.5.

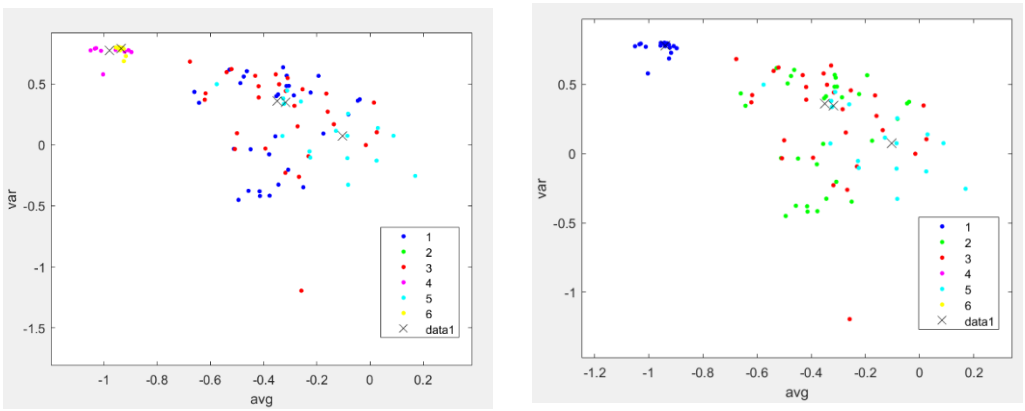

Figura 3.5. 3º prueba con k=6 y 4º prueba con k=6

Además, k=6 agrupa las voces de los niños correctamente. Sin embargo, k=7 no es capaz de ver esa continuidad de las voces de los niños en ambos audios como sí ocurre en la 2º prueba, considera que son dos audios muy diferentes.

Haciendo una comparativa entre k=6 y k=7 se observa que hay bastante similitud. Las diferencias están en qué al aumentar una clase, se logra capturar una mayor cantidad de matices en los resultados. En ocasiones, este aumento permite detectar detalles como que el sonido del avión se devuelve más estridente al acercarse al nodo que lo registra. En otras situaciones es posible identificar tres tipos diferentes de sonidos en audios con múltiples fuentes sonoras, por ejemplo, en el caso de los audios de las lanchas puede detectar cuando está cerca, lejos y también la presencia de algunas voces. Incluso permite hacer una detección más precisa en los audios con voces de niños, donde en el segundo audio, estos sonidos son bastante sutiles y a menudo pasan inadvertidos. Es importante tener en cuenta que estos resultados están influenciados por las características utilizadas en el proceso de clasificación.

Finalmente, estudio el caso de k=8. Realizo una clasificación inicial usando todas las características. Al observar los resultados obtenidos se muestra que el uso de tantas clases para este conjunto de audios no es adecuado, dado que hay una clase que contiene únicamente 5 muestras de un total de 120, no son suficientes como para considerar a este un grupo de interés sólido.

Generar 8 grupos distintos parece ser una elección ambiciosa para este conjunto de audios, tal vez con mayor variedad de clases de sonidos podría brindar resultados más significativos.

De esta primera etapa del estudio, se puede concluir que los resultados no son muy definitivos porque cada vez que se repite la clasificación utilizando *k-means* las agrupaciones resultantes varían ligeramente, aunque el modelo no llega a perder la capacidad de diferenciar entre las clases existentes en la base de datos.

La razón detrás de estas variaciones se debe a la naturaleza aleatoria de la inicialización de los centroides del algoritmo de *k-means*.

3.2.2 Evaluación de la robustez de *k-means*

En esta siguiente etapa, mi objetivo se centra en averiguar si el algoritmo *k-means* demuestra una clasificación robusta y, por tanto, los resultados obtenidos pueden considerarse concluyentes y fiables. Esto es especialmente relevante, ya que es común que los resultados varíen en cada ejecución del algoritmo.

Para abordar este problema implemento dos modificaciones que considero serán útiles para llegar a una deducción sólida. En primer lugar, decido repetir el proceso de clasificación varias veces usando el algoritmo *k-means*, en este caso opto por utilizar el vector de características completo, ya que considero que toda la información que aporta es útil. Por otro lado, aleatorizo la lectura de los archivos de audio de forma que en cada nueva iteración del algoritmo los archivos de audio se leerán en un orden diferente.

3.2.2.1 Repetir centroides

Inicialmente se prepara un código que realiza unas 4 repeticiones del proceso de clasificación cambiando en cada una de ellas el orden de lectura de los audios. Además, realizo una modificación en los parámetros de clasificación que utiliza *k-means*.

```
[ind, centros,~,Dis] =
kmeans(buffvectoresNor,k,'Distance','cityblock','Replicates',8,'Display','iter');
```

El algoritmo con estos parámetros hace lo siguiente: escoge unos centroides iniciales aleatorios, con esto forma las agrupaciones y luego calcula la suma de las distancias entre el centroide y cada punto usando la medida '*cityblock*'. La instrucción *"Replicates, 8"* indica que repetirá este proceso 8 veces aleatorizando los centros iniciales y finalmente se quedará con el centroide que dé la suma con el valor más bajo.

En la siguiente ronda de la clasificación le indico al clasificador *k-means* que utilice como centroides iniciales los centroides seleccionados en la clasificación anterior. Esto se debe a mi suposición de que si estos centroides fueron elegidos la mejor opción en la clasificación anterior es probable que proporcionen la mejor agrupación en esta ronda también. Repito este proceso en las 4 rondas. Para corroborar esto utilizo el parámetro '*Display*' que muestras los cálculos de las sumas, esto se observa en la siguiente imagen.

```
     4       1         4         2885.92
     5       1         3         2867.52
     6       1         1          2861.1
Best total sum of distances = 2829.96
  iter   phase       num             sum
     1       1       120         2849.98
Best total sum of distances = 2849.98
  iter   phase       num             sum
     1       1       120         2829.96
Best total sum of distances = 2829.96
  iter   phase       num             sum
     1       1       120         2840.18
Best total sum of distances = 2840.18
```

Figura 3.6. Muestra los resultados de las sumas de las clasificaciones

En la Figura 3.6 se muestra como en la 2º, 3º y 4º ronda de clasificación, el algoritmo no itera en busca de un centroide mejor, lo que sugiere que los centroides previamente seleccionados eran

adecuados. No obstante, esto contrasta con los valores obtenidos de las sumas de las distancias, ya que, dado que el algoritmo busca minimizar esta distancia, se esperaría que cada iteración seleccionara un valor menor. Curiosamente, en la 2º iteración se elige un valor superior al obtenido anteriormente, lo cual puede ser desconcertante.

Las conclusiones extraídas de esta imagen sugieren que la estrategia empleada no ha sido beneficiosa, ya que ha generado resultados contradictorios. Por lo tanto, esta idea se descarta y se busca una nueva forma de abordar el problema.

3.2.2.2 Matriz de similitud

Después de la ineficacia del enfoque anterior, procedo a construir una matriz de similitud. Esta matriz permite evaluar el nivel de similitud entre diversos patrones, de este modo espero determinar si el algoritmo *k-means* logra ofrecer una clasificación robusta.

Para empezar, desarrollo un código que repita la clasificación 50 veces, variando la secuencia de lectura de los archivos de audio en cada iteración. Luego almaceno en un Excel los índices de las agrupaciones obtenidas en cada repetición. Posteriormente, procedo a construir la matriz de similitud a partir del Excel. En este proceso, se compara cada *frame* con todas las demás posibles *frames* y se calcula la similitud entre ellas para cada repetición de *k-means*, sumando un "1" si ambas *frames* pertenecen a la misma clase o un "0" si son de clases distintas. De esta forma se obtiene una similitud máxima igual al número de repeticiones (50) y una similitud mínima de 0. Además, opto por crear una versión de esta matriz con un umbral de similitud. Fijo el valor del umbral en 45, correspondiente al 90% de las iteraciones. Esto significa, que solo se incluirán en la matriz valores de similitud superiores a 45.

Los resultados obtenidos a partir de las matrices de similitud sirven para estudiar cómo se agrupan las clases y averiguar si los audios seleccionados son representativos de su clase.

En este proceso genero dos figuras que plasman los resultados.

La Figura 3.7 muestra la matriz completa de similitud entre *frames*. Cada celda de la matriz indica el nivel de similitud de las *frames* fila y de las *frames* columna. Los valores están representados en una escala de colores que aparece en el margen derecho de la figura. Estos indican cuantas veces dos fragmentos de audio fueron clasificados igual por el algoritmo *k-means*.

La diagonal de la matriz representa la comparación de una *frame* consigo misma por ello tiene el valor 50. Un tono azul oscuro indica una mayor similitud entre *frames* e indica que pertenecen al mismo grupo o clase, mientras que un tono claro significa una menor coincidencia en la clasificación por tanto las *frames* son más diferentes y serán de distintos grupos.

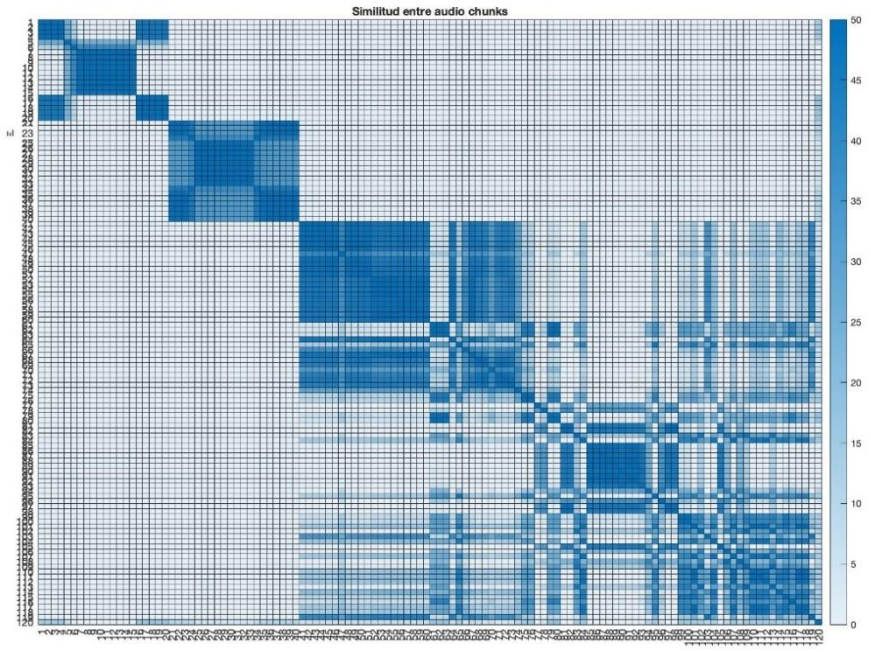

Figura 3.7. Matriz de similitud usando *k-means*

La Figura 3.8 es la matriz de similitud con umbral. Esta matriz solo muestra las semejanzas que superan un umbral del 90% de similitud, es por ello que solo aparecen valores de azul oscuro. Esta matriz destaca las semejanzas más robustas y consistentes entre los fragmentos de audio.

Los clústeres que aparecen definidos en la diagonal principal indican que los audios que los conforman son muy representativos de esa clase. Dicho de otro modo, los audios de dos clases distintas tienen grandes diferencias. Aquellos clústeres alejados de la diagonal presentan confusión entre las clases, es decir los audios que forman un clúster son muy similares a los que forman otro clúster diferente.

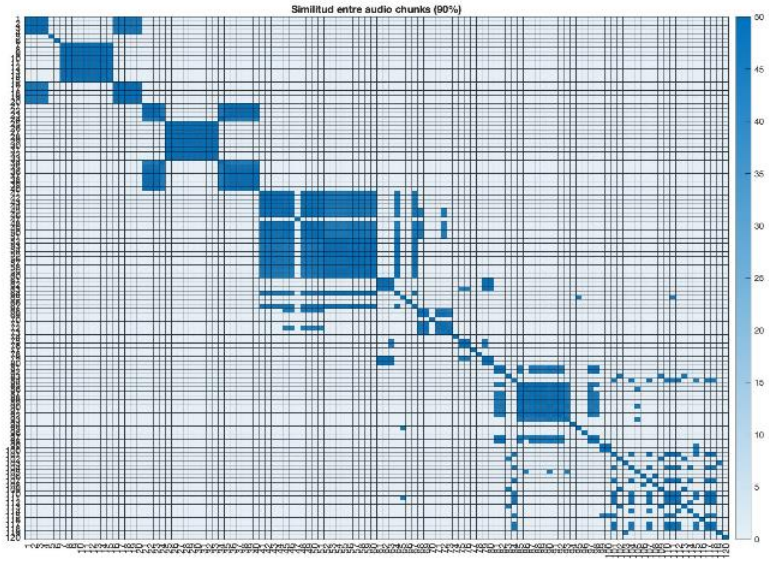

Figura 3.8. Matriz de similitud con umbral usando *k-means*

Partiendo de un análisis general de la Figura 3.7, se diferencian 3 claras agrupaciones. La primera y la segunda pertenecen ambas a audios de aviones y las tercera a un audio de lanchas.

Analizando las 20 primeras *frames* que pertenecen al primer audio, ruido de avión, se puede observar una agrupación central y 2 agrupaciones que lo rodean, ambas de 4 *frames*. Esto siguiere que este audio es clasificado en al menos dos clases distintas, lo cual se debe a la diversidad del sonido.
Sin embargo, al observar las siguientes 20 *frames* que corresponden al segundo audio, el ruido de otro avión se aprecia como forma un único y sólido cuadrado. Esto quiere decir que todas las *frames* de este audio son agrupadas en una misma clase.

Esta deducción también se basa en el análisis de los índices asignados a cada *frame*, Figura 3.9. Se comprueba que el primer audio tiene dos clases, la 2 y la 4, mientras que el segundo una sola, la 3. Al escuchar el primer audio, es evidente que esta variación en las clases coincide con el momento en que el avión se acerca y aleja.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
15	2	2	2	2	2	4	4	4	4	4	4	4	4	4	4	2	2	2	2	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
16	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	4	4	4	4	4	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
17	3	3	3	3	3	5	5	5	5	5	5	5	5	5	5	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2

Figura 3.9. Índices asignados a cada trama de los audios de ruido de aviones.

Estos dos audios muestran agrupaciones que son claramente separables de las demás, a pesar de la variabilidad dentro de cada agrupación. Además, los valores de sus clases no vuelven a aparecer en esa iteración. Esto hace pensar el algoritmo no supervisado *k-means* es capaz de hacer formaciones robustas y distinguir con precisión los sonidos de aviones.

He decidido considerar la existencia de 7 clases. 3 clases se utilizan para la clasificación de los dos primeros audios logrando una clara división. Los 4 audios restantes se dividen en 4 clases, pero a diferencia de los anteriores, estas no siguen una organización definida.

A partir de la *frame* 41, si no se aplica ningún umbral (Figura 3.7), se observa una mayor mezcla de clases en las agrupaciones. Se puede distinguir un tercer grupo que corresponde con el ruido de una lancha, al principio se observa con claridad, pero luego presenta más confusión porque es similar a los siguientes audios (Figura 3.10).

	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88
15	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	7	7	7	6	7	6	6	6	6	6	6	6	6	6	7	7	1	5	7	7	5	5	7	7	5	5	5	5
16	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	7	7	7	1	1	1	1	1	1	1	1	1	1	1	1	1	5	6	7	7	6	6	7	7	6	6	6	6
17	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	7	7	7	4	7	4	4	4	4	4	4	4	4	4	7	7	1	6	7	7	6	6	7	7	6	6	6	6

Figura 3.10. Índices asignados a cada trama

También aparece una cuarta agrupación que corresponde con las voces de niños, pero este sonido es más difícil de distinguir porque se solapa con ruidos de fondo.

3.2.3 Ampliación del *Dataset inicial*

Hago una nueva modificación en la base de datos, esta consiste en introducir una nueva clase de sonido: “Ruido de fondo”. Para llevar a cabo este ajuste, analizo nuevamente los audios grabados y selecciono un par de muestras que considero representativas del ruido de fondo. Estos audios seleccionados son luego incorporados al *dataset* inicial empleando para realizar las agrupaciones.

Contenido del audio	*Número de audios*	*Tiempo total*	*Número frames*
Ruido de aviones	2	2 min	40
Voces humanas	2	2 min	40
Ruido de lanchas	2	2 min	40
Ruido de fondo (pájaros)	2	2 min	40
Número total de frames: 160			

Tabla 3.3. Resumen de la base de datos ampliada

Después de aplicar el algoritmo de clasificación varias veces, compruebo que los clústeres obtenidos son consistentes, ya que las diferencias entre una clasificación y otra son mínimas. Incluso la posición de los centroides es bastante similar. Esto se observa en la Figura 3.11.

El resultado es relevante, ya que sugiere que el algoritmo está agrupando adecuadamente los audios procesados.

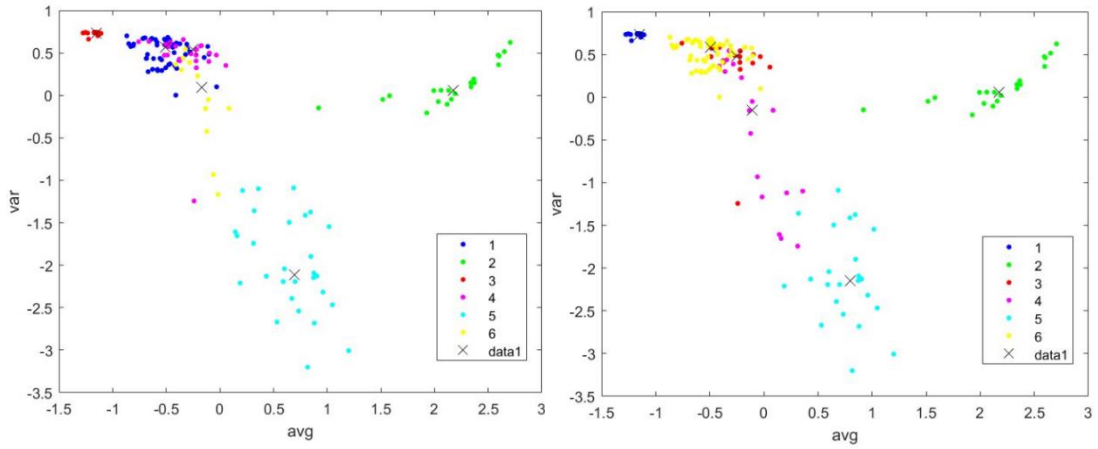

Figura 3.11. Media y varianza de la dispersión espectral en dos clasificaciones diferentes

Para comprobar la robustes del algoritmo *k-means* tras añadir esta nueva clase al *dataset* vuelvo a generar la matriz de similitud.

En esta ocasión, empleo 6 clases diferentes para la clasificación, ya que considero que son suficientes para realizar las agrupaciones. La diferencia se hace evidente en la clasificación de los audios de aviones. Con k=7, el algoritmo utiliza 2 o 3 clases para agrupar estos dos audios, mientras que con k=6 usa 2 clases. Lo que resulta en la formación de dos clústeres bastante

definidos y distinguibles incluso sin poner el umbral. Cada uno de ellos empela una clase diferente y estos valores no vuelven a repetirse en la iteración.

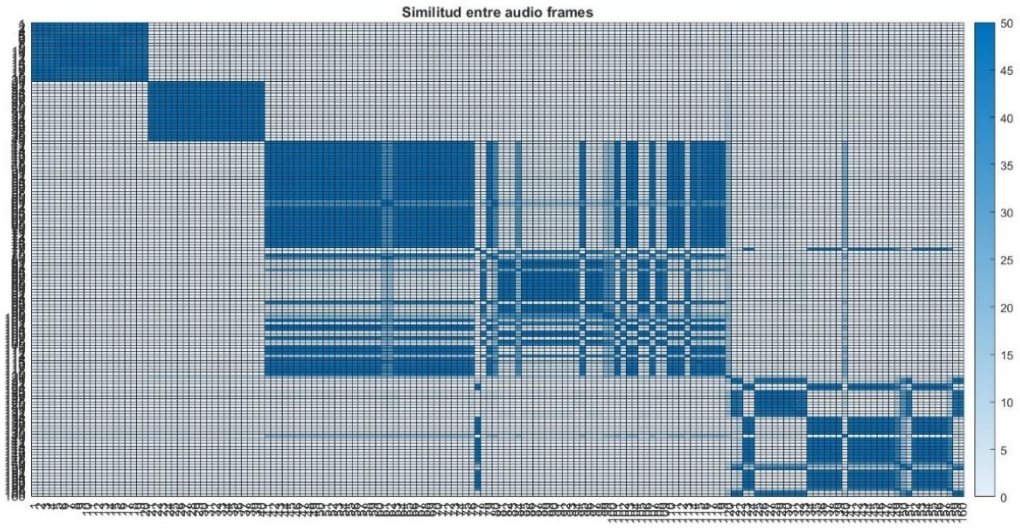

Figura 3.12. Matriz de similitud usando *k-means* y la base de datos ampliada

La Figura 3.12. Muestras los resultados obtenidos tras hacer 50 clasificaciones sin emplear el umbral. El resultado se asemeja al obtenido con el *dataset* inicial, pero parece que se consigue una mayor consistencia en la formación de clústeres, lo que sígnica que es capaz de distinguir los sonidos con mejor precisión. La figura 3.13. muestra las agrupaciones tras aplicar el umbral.

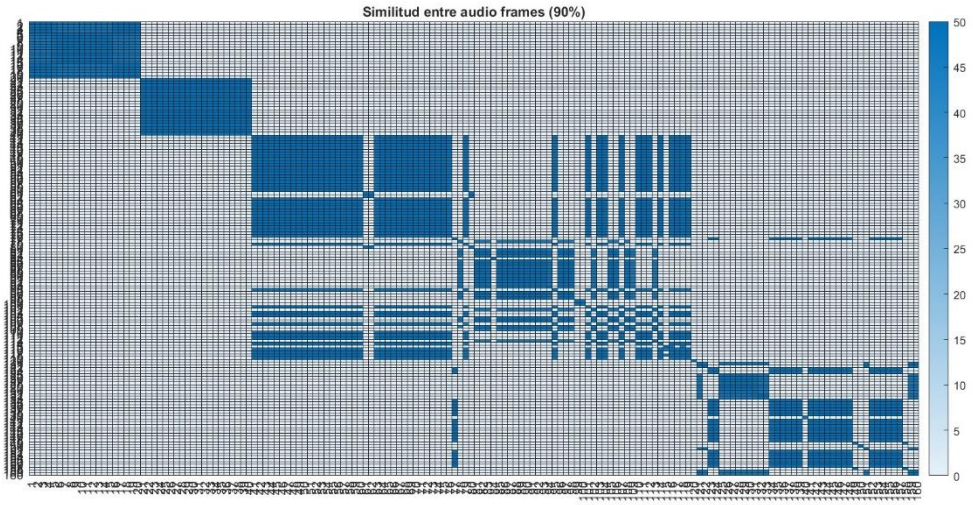

Figura 3.13. Matriz de similitud con umbral usando *k-means* y la base de datos ampliada

Posteriormente, nos dirigimos a examinar los audios de ruido de fondo, estos corresponden con las *frames* comprendidas entre 120-160, ubicadas en la parte inferior derecha de la matriz. Aquí, se puede diferenciar dos agrupaciones, un cuadrado más pequeño situado entre las *frames* 124 y 134, seguido de otro cuadrado más grande. Figura 3.14.

125	26	27	28	29	130	31	32	33	34	35	36	37	38	39	140	41	42	43	44	45	46	47	48	49	150	51	52	53	54	55	56	57	58	59	160
5	5	5	5	5	5	5	5	5	2	2	2	2	2	2	5	2	2	2	2	2	2	2	2	2	2	5	2	2	2	2	2	2	2	5	5
1	1	1	1	1	1	1	1	1	6	6	6	6	6	6	3	6	6	6	6	6	6	6	6	1	1	3	6	6	6	6	6	6	3	3	3
4	4	4	4	4	4	4	4	4	6	6	6	6	6	6	3	6	6	6	6	6	6	6	6	6	5	5	6	6	6	6	6	6	6	5	5

Figura 3.14. Índices asignados a cada trama de los audios de ruido de fondo

Para realizar la clasificación de estos dos audios emplea 3 clases distintas, son muchas teniendo en cuenta que solo hay 6 en total. Debido a esta diversidad en las clases es difícil que dos iteraciones coincidan en la misma agrupación. Se trata de sonidos cargados de matices y a pesar de su complejidad, el algoritmo es capaz de distinguirlos del resto de sonidos que conforman el *dataset*. El cuadrado más pequeño corresponde con la primera mitad del primer audio donde se escucha el sonido de pájaros en la distancia, difícil de percibir. Sin embargo, poco después de la mitad aparecen dos pájaros piando y graznando con mucha más intensidad de ahí la aparición del cuadrado más grande. El segundo audio presenta una mayor intensidad y presencia de sonidos de pájaros por ello continua la agrupación, con algunas partes en blanco que representan momentos de silencio.

A continuación, analizo la región central de la matriz. En esta zona se puede observar que los clústeres muestran una menor definición por tanto aparece una mayor dispersión de las agrupaciones. Sin embargo, se nota que entre las *frames* 40 y 80 se distingue un grupo que presenta una cruz blanca en el centro. Esta agrupación corresponde con los audios de ruido de lanchas y la cruz representa las diferencias que hay en el límite entre el primer y segundo audio. El algoritmo demuestra su capacidad al detectar y clasificar estas *frames* como idénticas, asignándoles la misma clase. Este fenómeno se muestra en la Figura 3.15.

	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
1	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	3	3	4	4	4	4	4	4	4	4	4	4	4	4	4	4	2	3	4	3
2	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	2	2	5	5	5	5	5	5	5	5	5	5	5	5	5	5	6	2	5	2
3	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	5	5	2	2	2	2	2	2	2	2	2	2	2	2	2	2	6	5	2	5

Figura 3.15. Índices asignados a cada trama de los audios de lanchas

A partir de la *frame* 80 están representados los audios de ruidos de niños, se puede apreciar que el algoritmo no logra formar una agrupación definida de estos.

También se observan la presencia de unas bandas verticales y horizontales alejadas de la diagonal. Estas bandas aparecen debido a las similitudes presentes entre audios.

Nos centramos en una de las bandas horizontales y procedo a explicar por qué surge. Esto se debe a la semejanza que presentan varias tramas pertenecientes a clases de audios diferentes. En la Figura 3.15 aparecen las clases asignadas a diversas tramas en 10 iteraciones.

En la Figura 3.16.a aparecen las *frames* de la 41 a la 50, estas pertenecen a un audio que contiene ruido de lanchas. La Figura 3.16.b contiene las *frames* de la 116 a la 118, pertenecientes a un audio de voces de niños. Lo que resulta notable en estas imágenes es la homogeneidad en la clasificación de todas las tramas, a pesar de pertenecer a grabaciones de audio con contenidos diferentes. El algoritmo detecta similitudes entre ellas y por ello las agrupa en el mismo clúster.

	41	42	43	44	45	46	47	48	49	50
1	4	4	4	4	4	4	4	4	4	4
2	5	5	5	5	5	5	5	5	5	5
3	2	2	2	2	2	2	2	2	2	2
4	5	5	5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5	5	5	5
6	5	5	5	5	5	5	5	5	5	5
7	1	1	1	1	1	1	1	1	1	1
8	3	3	3	3	3	3	3	3	3	3

116	117	118	119
4	4	4	4
5	5	5	5
2	2	2	2
5	5	5	5
5	5	5	5
5	5	5	5
1	1	1	1
3	3	3	3

Figura 3.16.a: Ìndices asignados a cada trama de los audios de lanchas y Figura 3.16.b: Índices asignados a cada trama de los audios de niños.

Para entender esto hay que recurrir a los audios. Al analizar los segundos correspondientes a las tramas 116-119 de los audios con sonidos de niños, se escucha como aparecen las voces de los niños, pero estos van montados en lanchas. Por lo tanto, este sonido de niños se mezcla con el sonido de fondo de las lanchas. Supongo que esta superposición de elementos es lo que da lugar a la formación de estas bandas. El algoritmo detecta que estos sonidos no pueden ser clasificados en ninguno de los clústeres principales (lanchas y niños) porque ambos están solapados y ninguno de los dos destaca en estas tramas.

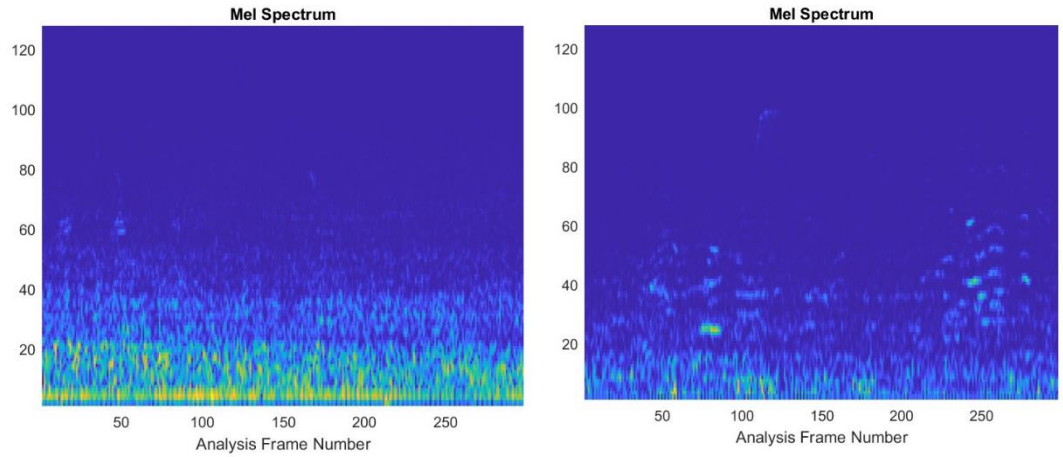

Figura 3.17. Espectro trama 43 de lanchas y espectro trama 119 de niños

Cuando observamos el espectrograma de dos tramas de cada uno de estos sonidos (Figura 3.17), podemos apreciar cierta similitud entre ellos. Esta similitud en las características del sonido es lo que lleva al algoritmo a formar un clúster diferente para ellos. En otras palabras, el algoritmo nota que las características extraídas de estas tramas son parecidas, lo que indica que estos sonidos comparten cualidades acústicas y por ello, los agrupa juntos en un clúster.

Este análisis establece que el algoritmo *k-means* es capaz de generar clasificaciones sólidas y confiables para los audios de aviones y sonidos de fondo. Cuando un audio es asignado a una de estas clases, su clasificación puede considerarse precisa, debido a que cada grupo presenta una serie de características que lo hacen inconfundible.

3.2.4 Evaluación mediante PCA

El análisis de componentes principales o PCA es una técnica estadística que se utiliza para reducir la dimensionalidad de un conjunto de datos, mientras se mantiene la mayor cantidad posible de información relevante. Esta técnica se explica más a fondo en el apartado 2.6. Empleo este método para tratar de deducir qué características son las más relevantes para realizar las agrupaciones ya que representan la mayor variabilidad de los datos y de esta manera reducir el conjunto de datos que entra al clasificador.

Tras aplicar este análisis a la matriz de características extraídas de los archivos de audio, se obtiene una matriz cuyas filas contiene las observaciones y las columnas las componentes principales. De los resultados obtenidos género dos gráficas y deduzco lo siguiente.

Me interesa determinar el número necesario de componentes principales para explicar al menos el 95% de la varianza en los datos. Para lograrlo, es necesario utilizar las 21 primeras componentes principales, tal como se muestra en la Figura 3.18. Esta reducción resulta beneficiosa, ya que indica que, empleando únicamente estas 21 componentes principales, en lugar de las 64 originales, se lograría realizar una clasificación que forme agrupaciones precisas y coherentes.

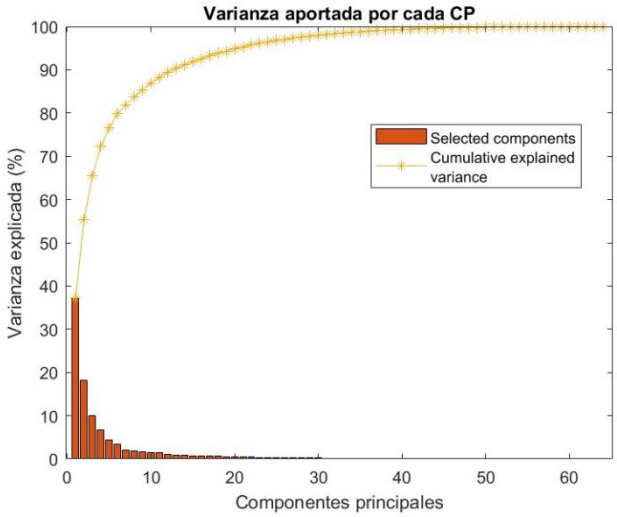

Figura 3.18. Varianza que aportan las componentes principales

La primera y segunda componente principal son las más interesantes pues explican el mayor porcentaje de varianza, un 55%. Estas componentes están formadas a partir de combinaciones lineales de las variables originales. Sería interesante averiguar qué características tiene un mayor peso en la formación de la 1º y 2º componente principal. Tras examinarlo, deduzco que son estas las características:

- Media y varianza del Delta MFCC
- Media y varianza del MFCC
- Media y varianza del Roll-off espectral
- Media de la dispersión espectral

Al analizar los resultados proporcionados por estas variables se deduce lo siguiente. La característica dispersión espectral presenta valores elevados cuando se trata de sonidos de aviones, lo que indica que la energía espectral está distribuida en un amplio rango de frecuencias. En contraste, los sonidos de lanchas, presenta valores bajos, lo que sugiere una concentración elevada de energía en un conjunto especifico de frecuencias. Esto se debe a que se trata de un sonido más

ruidoso, acompañado de un constante traqueteo. En los audios de voces de niños, los valores resultan ser muy bajos, lo que indica que las frecuencias dominantes se encuentran en una región muy estrecha. Este patrón es típico de la naturaleza de las voces infantiles, que suelen tener tonos más agudos y rangos de frecuencias más estrechos que los adultos.

La variable Roll-off espectral presenta valores altos que oscilan entre 1 y 2 en sonidos de aviones. Esto indica que la mayoría de la energía se encuentra concentrada en frecuencias más altas, es probable que esté relacionado con el ruido de los motores. En los sonidos de lanchas, los valores están alrededor de 0.5, indicando que la energía está concentrada en bajas frecuencias. Esto se debe al sonido del motor que se caracteriza por un contenido espectral bajo. Por otro lado, los sonidos de niños presentan valores inferiores, esto es típico en las voces humanas, ya que su contenido espectral es más bajo en comparación con los ruidos generados por maquinaria.

A continuación, procedo a realizar la clasificación de los audios utilizando el nuevo espacio de características formado. Para reducir el conjunto de datos escojo utilizar las 21 primeras componentes principales y observar sus resultados.

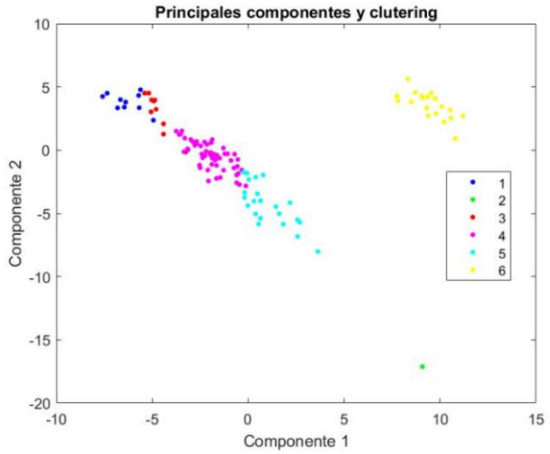

Figura 3.19. Agrupaciones formadas por la 1º y 2º componente principal

La Figura 3.19 muestras la distribución de los puntos que presentan las dos primeras componentes y se comprueba cómo son significativas para las agrupaciones. Los clústeres formados están bastante definidos y parecen clasificar correctamente los audios, a pesar de haber utilizado menos información que en las clasificaciones anteriores.

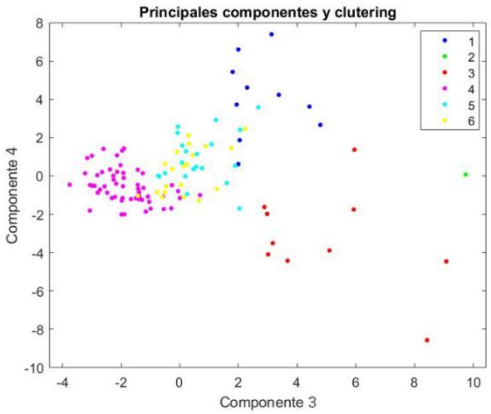

Figura 3.20. Agrupaciones formadas por la 3º y 4º componente principal

La Figura 3.20 muestra la distribución que se obtiene de la 3ª y 4ª componente. Sin embargo, estas componentes no resultan tan útiles en la formación de clústeres, ya que solo representan el 17% de la variabilidad, conteniendo información menos relevante si se usan solo ellas dos.

3.3 Clasificación *k-medoids*

A continuación, procedo a realizar la clasificación de la base de datos ampliada utilizado el algoritmo de clasificación no supervisado *k-medoids*. De esta manera, intento determinar si este algoritmo sería más apropiado para la detección y agrupación de estos sonidos de audio.

Para analizar los resultados, decido construir una matriz de similitud y extraer conclusiones tanto de esta matriz como de los vectores que indican los índices de las clases.

Al observar la Figura 3.21. Se aprecia como este método realiza peores agrupaciones en comparación con el anterior. Aunque se puede distinguir 4 agrupaciones.

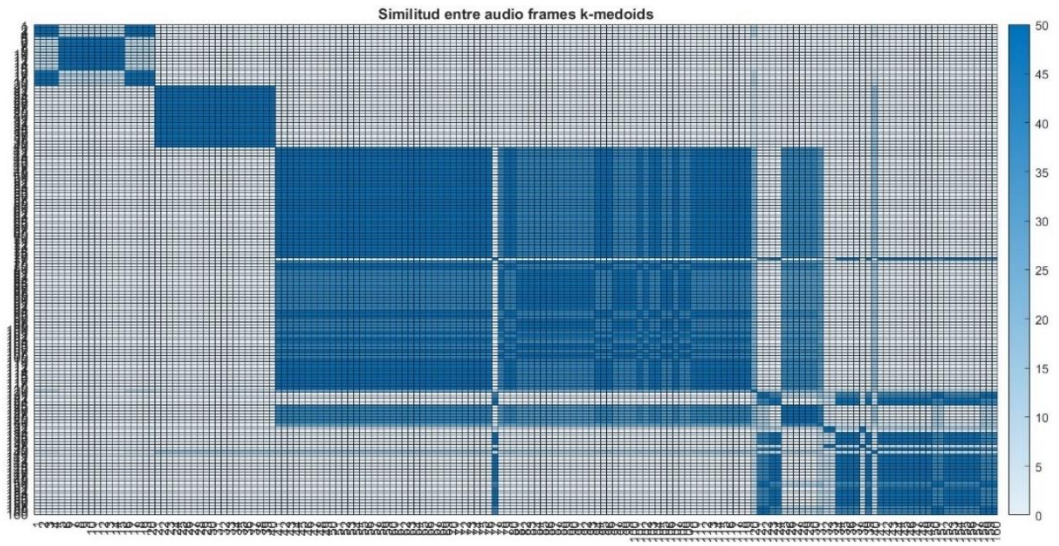

Figura 3.21. Matriz de similitud obtenida al realizar 50 clasificaciones empleando *k-medoids*.

Las dos primeras representan los sonidos de aviones y son clústeres bastante definidos y separables. Algo similar ocurre con los audios de ruido de fondo, pero con una agrupación menos delimitada y mucho más dispersa.

No obstante, en el centro de la matriz se forma una gran agrupación que engloba en la misma clase los sonidos de lanchas y de niños, lo cual es un error de clasificación. Este algoritmo no logra capturar los matices en los vectores de características que permiten la diferenciación entre estas dos clases. La Figura 3.22 muestra las tramas del 40 al 100 correspondientes a los sonidos mencionados.

	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70
1	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2

	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
1	3	3	3	3	3	3	5	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
2	1	1	1	1	1	1	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	2	2	2	2	2	2	4	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2

Figura 3.22. Contiene un fragmento de los índices asignados a cada trama empleando *k-medoids.*

El método *k-medoids* utiliza como centroides (los centros de los clústeres) uno de los vectores de características que se proporcionan como entrada, similar a cuando se calcula la mediana (percentil 50%) de una distribución. Algunas de estas características tienen histogramas con valores muy expandidos y es posible que escoja un de estos valores como centroide de algún clúster, reduciendo la precisión del algoritmo al realizar la clasificación.

3.4 Clasificación empleando *Gaussian Mixture Models*

En este apartado pruebo a agrupar los audios de la base de datos ampliada utilizando el algoritmo no supervisado de modelos de mezclas gaussianas (GMM). Para analizar los resultados genero una matriz de similitud. Figura 3.23.

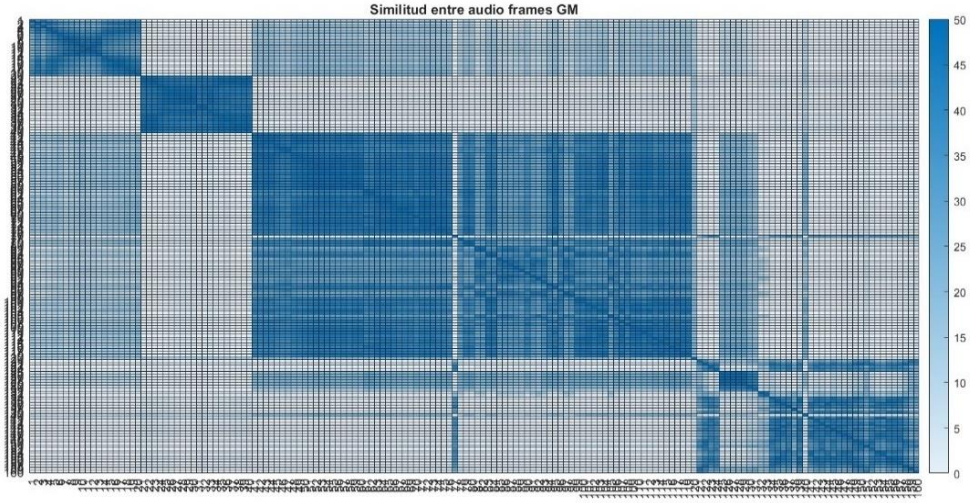

Figura 3.23. Matriz de similitud obtenida al realizar 50 clasificaciones empleando el modelo gaussiano.

Como se aprecia en la Figura 3.23, este algoritmo, logra formar agrupaciones que se identifican con las distintas clases de sonidos al igual que las anteriores, aunque en algunas de ellas aumenta la confusión con otras clases.

Se genera un clúster definido para el segundo audio, que no presenta ninguna relación con los demás. Esto también se refleja en la Figura 3.24 que representa el 90% de similitud entre tramas. Sin embargo, el resto de los clústeres presentan confusión. La primera agrupación, que corresponde al primer audio de aviones, contiene bandas azules que se extienden, englobando los

audios de lanchas y niños. Esto indica similitudes entre ellos, ya que comparten características que los hacen asociarse en la misma clase.

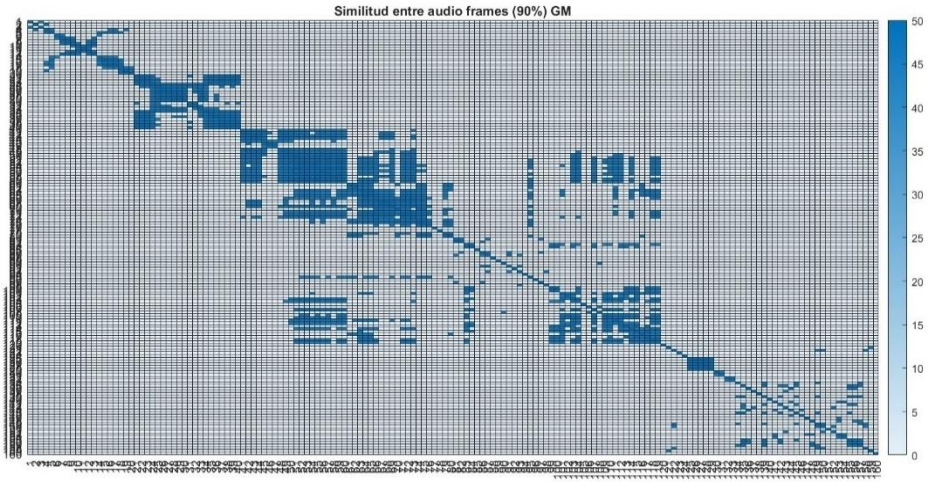

Figura 3.24. Matriz de similitud con umbral obtenida al realizar 50 clasificaciones empleando el modelo gaussiano.

Además, el algoritmo tampoco es capaz de identificar las características que permiten distinguir los sonidos de ruido de fondo, lo que resulta en la incapacidad de formar un clúster cuando se aplica un umbral del 90%.

Esto puede deberse a que el modelo gaussiano asigna a cada *frame* un grado de probabilidad para cada uno de los clústeres basándose en su afinidad. Por ello, plasma las similitudes que detecta entre las *frames.*

En resumen, aunque el algoritmo consigue identificar qué audios son separables, no lo hace con la suficiente precisión como para generar clústeres definidos que superen el umbral establecido. En consecuencia, este algoritmo es el que genera las peores agrupaciones.

Comparando los tres algoritmos de clasificación no supervisados empleados, se comprueba como *k-means* es el que obtiene los resultados más favorables y realiza agrupaciones más robustas.

Capítulo 4.

Conclusiones y propuestas de futuro

4.1 Conclusiones

Este trabajo se ha centrado en el estudio de unas grabaciones de sonido realizadas en el Parc de l'Albufera de Valencia con el objeto de desarrollar un clasificador no supervisado capaz de detectar sonidos de origen no humano. Para ello se han extraído y analizado una serie de características obtenidas de los audios y se han clasificados utilizando diferentes algoritmos de *machine learning* no supervisados.

La selección de una base de datos adecuada es un paso de suma importancia en el proceso. Dicha base debe estar compuesta por audios que sean altamente representativos de sus respectivas clases y que contengan calidad y claridad, dado que son estos los que permitirán al clasificador no supervisado generar y aprender patrones cruciales para la diferenciación entre clases. Además, la base de datos debe contener un número suficiente de audios, de manera que el clasificador disponga de un mayor conjunto de muestras para realizar comparaciones y poder generar patrones más robustos. Una vez el clasificador a aprendido a distinguir entre las clases estará en condiciones de clasificar con éxito cualquier audio nuevo que se le presente.

Las características seleccionadas para el análisis de los audios son un factor importante pues afectarán directamente a las agrupaciones que forme el clasificador. Se ha comprobado que características como Roll-off espectral y asimetría espectral han demostrado ser muy útiles para distinguir estas clases de sonidos y detectar matices que permiten su diferenciación.

En lo que respecta a los algoritmos de *machine learning* utilizados, el más efectivo resulta ser el algoritmo *k-means*, ya que es capaz de formar agrupaciones más robustas y definidas, lo que permite diferenciar de manera más precisa entre los diversos sonidos de la base de datos. Esto se observa especialmente en los audios de sonidos de aviones y de ruido de fondo, donde k-*means* muestra un rendimiento destacado, mientras que la clasificación de audios que contienen voces de niños resulta ser la más confusa y menos precisa. Esta dificultad podría indicar que, en ocasiones, el estudio de voces puede requerir enfoques de aprendizaje supervisado u otras técnicas más especializadas para obtener resultados óptimos.

Sin embargo, el algoritmo *k-medoids* no logra generar una percepción tan clara sobre los distintos sonidos, lo que da lugar a confusiones y clasificaciones erróneas en ciertos audios. Por último, el modelo gaussiano se muestra ineficaz en la creación de clústeres que sean representativos.

4.2 Líneas futuras

En cuanto a las mejoras que se podrían incluir en este estudio, se podrían centrar en tres aspectos principales.

En primer lugar, sería beneficioso aumentar la base de datos, lo que proporcionaría al algoritmo más muestras para identificar diferencias y matices en los sonidos. Esto podría llevar a una mayor capacidad para detectar la presencia de sonidos simultáneos en los audios, lo que, a su vez, mejoraría la precisión en la clasificación.

En segundo lugar, se podría profundizar en la búsqueda de características que proporcionen información adicional sobre los eventos sonoros, lo que permitiría una mejor distinción entre las diferentes clases de sonidos. En tercer lugar, se podría considerar la posibilidad de combinar algoritmos no supervisados con algoritmos supervisados para lograr un enfoque más efectivo en el estudio de las voces humanas.

Estas mejoras podrían contribuir significativamente a la calidad y precisión de la clasificación de sonidos en futuras investigaciones.

Bibliografía

[1] U. Qamar and M. S. Raza, *Data Science Concepts and Techniques with Applications*, vol. 2. Switzerland: Springer, 2023.

[2] Y. F. Phillips, M. Towsey, and P. Roe, "Revealing the ecological content of long-duration audio-recordings of the environment through clustering and visualisation," *PLoS One*, vol. 13, no. 3, Mar. 2018, doi: 10.1371/JOURNAL.PONE.0193345.

[3] V. Roman, "Aprendizaje No Supervisado en Machine Learning: Agrupación," *Ciencia y Datos*, 2019. https://medium.com/datos-y-ciencia/aprendizaje-no-supervisado-en-machine-learning-agrupaci%C3%B3n-bb8f25813edc (accessed Sep. 04, 2023).

[4] A. Jung, *Machine Learning The Basics*, 1st ed. Singapour: Springer, 2022. [Online]. Available: https://link.springer.com/bookseries/16715

[5] A. Mesaros, T. Heittola, T. Virtanen, and M. D. Plumbley, "Sound Event Detection: A tutorial," *IEEE Signal Process Mag*, vol. 38, no. 5, pp. 67–83, Aug. 2021, doi: 10.1109/MSP.2021.3090678.

[6] D. Stowell, "Computational bioacoustics with deep learning: a review and roadmap," *PeerJ*, vol. 10, p. e13152, Mar. 2022, doi: 10.7717/PEERJ.13152/SUPP-1.

[7] S. D. Tiwari and K. S. Subraya, "Exploring Regression-based Approach for Sound Event Detection in Noisy Environments," *IJACSA) International Journal of Advanced Computer Science and Applications*, vol. 13, no. 7, p. 2022, Accessed: Sep. 04, 2023. [Online]. Available: www.ijacsa.thesai.org

[8] A. Mesaros *et al.*, "Detection and Classification of Acoustic Scenes and Events: Outcome of the DCASE 2016 Challenge," *IEEE/ACM Trans Audio Speech Lang Process*, vol. 26, no. 2, p. 379, 2018, doi: 10.1109/TASLP.2017.2778423.

[9] A. Mesaros, T. Heittola, and T. Virtanen, "TUT database for acoustic scene classification and sound event detection," *European Signal Processing Conference*, vol. 2016-November, pp. 1128–1132, Nov. 2016, doi: 10.1109/EUSIPCO.2016.7760424.

[10] T. Heittola, A. Mesaros, A. Eronen, and T. Virtanen, "Context-dependent sound event detection," *EURASIP J Audio Speech Music Process*, no. 1, 2013, Accessed: Sep. 04, 2023. [Online]. Available: http://asmp.eurasipjournals.com/content/2013/1/1

[11] T. Heittola, A. Mesaros, T. Virtanen, and A. Eronen, "Sound Event Detection in Multisource Environments Using Source Separation," in *Workshop on Machine Listening Multisource Environments, CHiME*, Florence, Italy, 2011, pp. 36–40.

[12] Y. Raj Pandeya, B. Bhattarai, and J. Lee, "Visual Object Detector for Cow Sound Event Detection," *IEEE Access*, vol. 8, pp. 162625–162633, 2020, doi: 10.1109/ACCESS.2020.3022058.

[13] SED, "Sound event Detector." https://soundeventdetector.eu/ (accessed Jul. 04, 2023).

[14] X. Zheng *et al.*, "A CRNN System for Sound Event Detection Based on Gastrointestinal Sound Dataset Collected by Wearable Auscultation Devices," *IEEE Access*, vol. 8, pp. 157892–157905, Aug. 2020, doi: 10.1109/ACCESS.2020.3020099.

[15] T. Fernando *et al.*, "Robust and Interpretable Temporal Convolution Network for Event Detection in Lung Sound Recordings," *IEEE J Biomed Health Inform*, vol. 26, no. 7, 2022, doi: 10.1109/JBHI.2022.3144314.

[16] NTI audio, “Fast Fourier Transformation FFT - Basics,” *NTI audio*, 2020. https://www.nti-audio.com/en/support/know-how/fast-fourier-transform-fft (accessed Jul. 10, 2023).

[17] A. Albiol, V. Naranjo, and J. Prades´indice, “Tratamiento Digital de la Señal Teoría y Aplicaciones,” Valencia: UPV, pp. 49–51.

[18] L. Robets, “Understanding the Mel Spectrogram,” *Analytics Vidhya*, Mar. 06, 2020. https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53 (accessed Jul. 10, 2023).

[19] K. Doshi, “Audio Deep Learning Made Simple - Why Mel Spectrograms perform better,” *Ketan Doshi blog*, Feb. 2021. https://ketanhdoshi.github.io/Audio-Mel/ (accessed Sep. 04, 2023).

[20] MathWorks, “Mel spectrogram - MATLAB,” *España*. https://es.mathworks.com/help/audio/ref/melspectrogram.html (accessed Sep. 04, 2023).

[21] H. Subramanian, P. Rao, and S. D. Roy, “AUDIO SIGNAL CLASSIFICATION,” Bombay, 2004.

[22] MathWorks, “Compute delta features - MATLAB audioDelta,” *España*. https://es.mathworks.com/help/audio/ref/audiodelta.html (accessed Sep. 04, 2023).

[23] C. S. Jensen, R. T. Snodgrass, J. Chomicki, and D. Toman, “Audio Representation,” *Encyclopedia of Database Systems*, vol. 1399, pp. 160–167, 2009, doi: 10.1007/978-0-387-39940-9_1442.

[24] M. M. Rahman, J. Kabi, K. Nazrul, M. Al-Amin, and M. Al-Amin Bhuiyan, “Continuous Bangla Speech Segmentation using Short-term Speech Features Extraction Approaches,” 2012. [Online]. Available: www.ijacsa.thesai.org

[25] G. Peeters, “A large set of audio features for sound description ,” Apr. 2004.

[26] Biology For Life, “Skew.” https://www.biologyforlife.com/skew.html (accessed Jul. 13, 2023).

[27] “Kurtosis,” *SIEMENS*, Aug. 29, 2019. https://community.sw.siemens.com/s/article/kurtosis (accessed Jul. 13, 2023).

[28] “Spectral features ,” *Twoears*, 2016. https://docs.twoears.eu/en/latest/afe/available-processors/spectral-features/ (accessed Jul. 15, 2023).

[29] “K-Medoids clustering-Theoretical Explanation,” *JavaTpoint*. https://www.javatpoint.com/k-medoids-clustering-theoretical-explanation (accessed Aug. 02, 2023).

[30] A. Entezami, H. Sarmadi, and B. Saeedi Razavi, “An innovative hybrid strategy for structural health monitoring by modal flexibility and clustering methods,” *J Civ Struct Health Monit*, vol. 10, no. 5, pp. 845–859, Nov. 2020, doi: 10.1007/S13349-020-00421-4.

[31] A. Singh, “Build Better and Accurate Clusters with Gaussian Mixture Models,” *Analytics Vidhya*, Jul. 21, 2023. https://www.analyticsvidhya.com/blog/2019/10/gaussian-mixture-models-clustering/ (accessed Aug. 04, 2023).

[32] A. C. Müller, “Clustering and Mixture Models,” *amuller*, Jun. 04, 2020. https://amueller.github.io/aml/03-unsupervised-learning/02-clustering-mixture-models.html (accessed Aug. 05, 2023).

[33] F. Karabiber, “Hierarchical Clustering,” *Learn data sci*, 2023. https://www.learndatasci.com/glossary/hierarchical-clustering/ (accessed Aug. 06, 2023).

[34] H. Sharma, “Hierarchical Clustering,” *Medium*, Apr. 23, 2021. https://harshsharma1091996.medium.com/hierarchical-clustering-996745fe656b (accessed Aug. 07, 2023).

[35] I. M. Khater, I. R. Nabi, and G. Hamarneh, “A Review of Super-Resolution Single-Molecule Localization Microscopy Cluster Analysis and Quantification Methods,” *Patterns*, vol. 1, no. 3, Jun. 2020, doi: 10.1016/J.PATTER.2020.100038.

[36] Z. Jaadi and B. Whitfield, “A Step-by-Step Explanation of Principal Component Analysis (PCA),” *Built in*, Mar. 29, 2023. https://builtin.com/data-science/step-step-explanation-principal-component-analysis (accessed Aug. 10, 2023).

[37] Jun de Wu, “Análisis de Componentes Principales: Una breve introducción matemática,” *Damavis blog*, Oct. 28, 2021. https://blog.damavis.com/analisis-de-componentes-principales-una-breve-introduccion-matematica/ (accessed Aug. 10, 2023).

[38] Generalitat Valenciana, “L’ALBUFERA,” *Conselleria de Agricultura, Desarrollo Rural, Emergencia Climática y Transición Ecológica*, Valencia. Accessed: Aug. 11, 2023. [Online]. Available: https://parquesnaturales.gva.es/es/web/pn-l-albufera/conocenos

[39] N. P. García-de-la-Puente, F. Fuentes-Hurtado, L. Fuster, V. Naranjo, and G. Pinero, “Deep Learning Models for Gunshot Detection in the Albufera Natural Park,” *ITEAM, KNODIS Research Group, Dep. Sistemas Informaticos, Universidad Politécnica de Madrid, I3B, Universitat Politécnica de Valencia*, 2022, doi: 10.13039/501100011033.

Parte II

Anexo

Código implementado en Matlab

En este anexo se presenta el código más importante. En el se lleva a cabo la extracción de características de los audios y su clasificación empleando el algoritmo de *k-means.* Además, se explica al procese de generación de la matriz de similitud.

```
%% Definir los parámetros para el audio
% Se definene los parametros para el estudio de cada audio: la frecuencia %
% de muestreo, el número de muestras por segmento (3sg de un audio de     %
% 1min), la duración de la frame(25ms) y el solape entre frames (10ms)    %
%                                                                         %

fs = 24000;
samplesPerCapture = 3*fs;

frameDuration = 0.025;
frameSamples = round(frameDuration*fs);

hopDuration = 0.010;
hopSamples = round(hopDuration*fs);

%-------------------------------------------------------------------------%

%% Definir los parámetros y las características                          %
% La función audioFeatureExtractor permite extraer las características de %
% los audios. Hay que indicar las características que nos interesan.      %
% Además, de una serie de parámetros para el análisis del audio, como el  %
% tamaño de la ventana empleada (25ms) para dividir el segmento del audio %
% y extraer las características.                                          %
% La función setExtractorParams permite ajustar otro parámetro de la      %
% función anterior. Concretamente el número de bandas del espectrograma de%
% Mel y la normalización de ventana.                                      %
%                                                                         %

afe = audioFeatureExtractor( ...
    'SampleRate',fs, ...
    'FFTLength',1024, ...
    'Window',hann(frameSamples,'periodic'), ...
    'OverlapLength',frameSamples - hopSamples, ...
    'melSpectrum',true, ...
     "mfcc",true,...
     "mfccDelta",true,...
     "spectralFlux",true,...
     "spectralCentroid",true,...
     "spectralSpread",true,...
     "spectralSkewness",true,...
     "spectralKurtosis",true,...
     "spectralRolloffPoint",true);

numBands = 128;
setExtractorParams(afe,
'melSpectrum','NumBands',numBands,'WindowNormalization',true);

informacion = info(afe);

%-------------------------------------------------------------------------%
```

```
%% Crear un buffer para almacenar el segmento de audio leído              %
%                                                                         %
audioBuffer = dsp.AsyncBuffer(samplesPerCapture);

%-------------------------------------------------------------------------%


%% Leer los audios, extraer las características y realizar la clasificación
% Este apartado consta de varias secciones que iré explicando            %
% progresivamente                                                         %
%                                                                         %
%                                                                         %
% Dirección de la carpeta que contiene los archivos de audios y nombres   %
c=0;
Ab=dir('audios fondo');
a = Ab(~[Ab.isdir]);
file_names = {a.name};
numAudios = numel(file_names);
%-------------------------------------------------------------------------%

% Crear un buffer para las características extraídas de los audios        %
buffdescriptor = zeros(20,64);
numFilas = numAudios*20;
buffconvetores = zeros(numFilas,64);
buffvectoresNor = zeros(numFilas,64);
%-------------------------------------------------------------------------%

% se inicia el proceso de análisis y extracción de características y      %
% posteriormente la clasificación. Se realizan 50 lecturas de los audios  %
% y su respectiva clasificación                                           %

for j=1:50

    % Obtener un orden aleatorio para leer los nombres de los archivos
    rng('shuffle');
    random_order = randperm(numel(file_names));
    ordenaudiosnum(j,1:8)= random_order;

    % Acceder a los audios
    cd('audios fondo')
    ordenaudios(j,:) = {a(random_order).name};


    % Este for sirve para analizar todos los audios de la carpeta 1 vez
    for i=1:numel(a)

        % Carga todo el audio y realiza una lectura del audio por segmentos
        afr = dsp.AudioFileReader(a(random_order(i)).name, 'SamplesPerFrame',
       samplesPerCapture);

        c=c+1;
        % Audios de duracion 1min, cogo trozos de 3sg = 20 tramas en total
        infor = audioinfo(a(i).name);
        lengthaudio = infor.TotalSamples;
        nameaudio = a(random_order(i)).name;

        t=0;
```

```
% A cada ciclo carga y analiza individualmente un segmento de      %
% audio (trama)                                                    %
 while ~isDone(afr)

    [x,eof] = afr();

    if eof == 1
        break;
    end

    t =t+1;
    write(audioBuffer,x);
    % Extraer las características del segmento
    features = extract(afe,x);
    % Pasar lo valores a dB
    auditoryFeatures = 10*log10(features + 1e-6);

    % Acumular la matriz auditoryFeatures que se calculan en cada
    % ronda en una matriz 3D
    if t==1
        matrixaudio = auditoryFeatures;
    else
        matrixaudio = cat(3,matrixaudio,auditoryFeatures);
    end


    % Calcular la media y varianza de las características extraídas
    % MFCC media y varianza
    avg_mfcc=mean(features(:,129:141));
    var_mfcc=avg_mfcc.^2-mean(features(:,129:141).^2);
    % MFCCdelta media y varianza
    avg_mfccdelta= mean(features(:,142:154));
    var_mfccdelta=avg_mfccdelta.^2-mean(features(:,142:154).^2);
    %Centroid media y varianza
    avg_centroid=mean(features(:,155));
    var_centroid=avg_centroid.^2-mean(features(:,155).^2);
    %Flux media y varianza
    avg_flux=mean(features(:,156));
    var_flux=avg_flux.^2-mean(features(:,156).^2);
    %kurtosis media y varianza
    avg_kurtosis=mean(features(:,157));
    var_kurtosis=avg_kurtosis.^2-mean(features(:,157).^2);
    %rolloffPoint media y varianza
    avg_rolloffpoint=mean(features(:,158));
    var_rolloffpoint=avg_rolloffpoint.^2-mean(features(:,158).^2);
    %skewness media y varianza
    avg_skewness=mean(features(:,159));
    var_skewness=avg_skewness.^2-mean(features(:,159).^2);
    %Spread media y varianza
    avg_spread=mean(features(:,160));
    var_spread=avg_spread.^2-mean(features(:,160).^2);

    % Escoger las características para formar el vector de datos
    % He creado un vector de 3sg de audio que contiene estas
     características
    descriptor= [avg_mfcc,var_mfcc,avg_mfccdelta,var_mfccdelta,
     avg_centroid,var_centroid,avg_flux,var_flux,avg_kurtosis,
     var_kurtosis, avg_rolloffpoint,var_rolloffpoint,
```

```
            avg_skewness,var_skewness,avg_spread,var_spread];

            buffdescriptor(t,:) = descriptor;

          end

          % Acaba de leer 1 audio, va a leer el siguiente
          %-----------------------------------------------------------------%


          % Guarda la matriz 3D de características y nombre del audio
          % correspondiente
          FeaturesFile(i,:) = {nameaudio, [matrixaudio]};

          % Matriz formada por el vector de índices generado para cada
          % segmento
          buffconvetores((c-1)*20+1:c*20,:) = buffdescriptor;
    end

    %-----------------------------------------------------------------------%
    % Normalizar dicha matriz para igualar la influencia de todas las
    % variables
       for k=1:size(buffconvetores,2)
                media = mean(buffconvetores(:,k));
                desv = std(buffconvetores(:,k));
                normalizar = (buffconvetores(:,k)-media) / desv;
                buffvectoresNor(:,k) = normalizar;
          end


     % Entrada al clasificador k-means
     k=6;
     vectores = buffvectoresNor(1:160,:);
     [ind, centros] = kmeans(vectores,k,'Replicates',8,
      'Distance','cityblock');

      c=0;

     % Volvemos a la carpeta anterior para que guarde allí el cvs
     oldFolder = cd('C:\Users\Alba\Desktop\POLIMI\TFG\codigo');

     % Almacenar el resultado de la clasificación en un Excel
     writematrix(ind','Resultados_fondo_kmeans.xls','WriteMode','append');

end
%-------------------------------------------------------------------------%


%% Plot de las muestras de datos mediante las características

 for i=53:2:64

    figure
    gscatter(vectores(:,i), vectores(:,i+1), ind, 'bgrmcyk', '.', 10);
    xlabel('avg'); ylabel('var')
    hold on;
```

```
    plot(centros(:,i), centros(:,i+1), 'kx', 'MarkerSize', 10);

    pause(2)

 end

%-------------------------------------------------------------------------%

%% Reordenar los vectores de índices de las tramas

Resultados=zeros(50,160);
Resultados=table2array(readtable('Resultados_fondo_kmeans.xls'));
Resultados_ordenados_fondo= zeros(size(Resultados));

for col=1:50       %para cada columna
    for fil=1:8    %para una fila
        if ordenaudiosnum(col,fil) == 1
            Resultados_ordenados_fondo(col,1:20) = Resultados(col,(fil-
             1)*20+1:fil*20);
        elseif ordenaudiosnum(col,fil) == 2
            Resultados_ordenados_fondo(col,21:40) = Resultados(col,(fil-
             1)*20+1:fil*20);
         elseif ordenaudiosnum(col,fil) == 3
            Resultados_ordenados_fondo(col,41:60) = Resultados(col,(fil-
             1)*20+1:fil*20);
         elseif ordenaudiosnum(col,fil) == 4
            Resultados_ordenados_fondo(col,61:80) = Resultados(col,(fil-
             1)*20+1:fil*20);
         elseif ordenaudiosnum(col,fil) == 5
            Resultados_ordenados_fondo(col,81:100) = Resultados(col,(fil-
             1)*20+1:fil*20);
         elseif ordenaudiosnum(col,fil) == 6
            Resultados_ordenados_fondo(col,101:120) = Resultados(col,(fil-
             1)*20+1:fil*20);
         elseif ordenaudiosnum(col,fil) == 7
            Resultados_ordenados_fondo(col,121:140) = Resultados(col,(fil-
             1)*20+1:fil*20);
         elseif ordenaudiosnum(col,fil) == 8
            Resultados_ordenados_fondo(col,141:160) = Resultados(col,(fil-
             1)*20+1:fil*20);
        end
    end
end

%-------------------------------------------------------------------------%

%% Crear la matriz de similitud

[nf, nc] = size(Resultados_ordenados_fondo);
% matriz de similitud
R = zeros(nc);

Rumbral = R;
% umbral que corresponde 90% de las iteraciones (0.9*30)
Ncoincide = 45;

for i = 1:nc

    for j = i:nc
```

```
        % Cuenta cuántas veces ha clasificado igual
         Cuenta = (Resultados_ordenados_fondo(:,i) ==
         Resultados_ordenados_fondo(:,j));
         R(i,j) = sum(Cuenta);

         R(j,i) = R(i,j);

         % solo se queda con los que han coincidido en N de las 50 iteraciones
         if R(i,j) >= Ncoincide
             Rumbral(i,j) = R(i,j);
             Rumbral(j,i) = Rumbral(i,j);
         end
    end
end
figure, heatmap(R,'Title','Similitud entre audio frames')
figure, heatmap(Rumbral,'Title','Similitud entre audio frames (90%)')
```