Universidad de Valladolid

Escuela Técnica Superior de Ingenieros de Telecomunicación

Grado en Ingeniería de Tecnologías Específicas de Telecomunicación
Mención Sistemas de Telecomunicación

Aplicando algoritmos de Machine Learning en un Sistema acústico de detección de larvas de insectos xilófagos

Trabajo Fin de Grado

Autora: Ana Benito Sáez
Tutora: Lara del Val Puente
Convocatoria: Valladolid, Junio de 2025

A mi familia, por confiar en mí y darme la oportunidad de llegar hasta aquí.

Y, en especial, a mis abuelos, por la ilusión con la que siempre han seguido este camino.

Resumen

Este Trabajo Fin de Grado se centra en la detección de insectos xilófagos mediante técnicas acústicas, con el objetivo de profundizar en los retos actuales asociados a esta línea de investigación y proponer mejoras en la interpretación y clasificación de las señales. Para ello, se han empleado imágenes acústicas generadas a partir de un array de micrófonos MEMS, permitiendo estimar la localización de larvas de Hylotrupes bajulus en función de la energía acústica emitida durante su actividad alimentaria sobre madera. La captura de las señales se ha desarrollado en una cámara anecoica, utilizando cuatro vigas de madera separadas 60 cm del array y una base de datos de 148.000 señales acústicas registradas a partir de seis larvas.

A partir de esta base de datos, se han implementado esquemas en LabVIEW para la extracción sistemática de características acústicas relevantes. Previamente al entrenamiento de los modelos, se evalúan distintas técnicas de escalado con el objetivo de optimizar la calidad de los datos. Se entrenan diversos algoritmos de aprendizaje automático para identificar patrones discriminativos, discriminando entre todas las clases en unos casos y únicamente entre nudo y larva en otros, y evaluar su capacidad de generalización, así como para comparar entre distintas configuraciones de hiperparámetros. Los resultados pretenden contribuir a futuros estudios orientados a la detección en entornos distintos a los aquí evaluados, favoreciendo la aplicación práctica de los modelos desarrollados.

Palabras clave

Característica acústica, normalización, estandarización, algoritmos de clasificación, insectos xilófagos, larvas, nudos, validación cruzada.

Abstract

This Final Degree Project focuses on the detection of wood-boring insects using acoustic techniques, with the objective of deepening the current challenges associated with this line of research and proposing improvements in the interpretation and classification of the signals. For this purpose, acoustic images generated from a MEMS microphone array have been used, allowing the estimation of the location of Hylotrupes bajulus larvae based on the acoustic energy emitted during their feeding activity on wood. The signal capture was carried out in an anechoic chamber, using four wooden beams separated 60 cm from the array and a database of 148,000 acoustic signals recorded from six larvae.

From this database, schemes have been implemented in LabVIEW for the systematic extraction of relevant acoustic features. Prior to model training, different scaling techniques are evaluated with the goal of optimizing data quality. Various machine learning algorithms are trained to identify discriminative patterns, discriminating between all classes in some cases and only between knot and larva in others, and to evaluate their generalization capacity, as well as to compare different hyperparameter configurations. The results aim to contribute to future studies oriented to detection in environments different from those evaluated here, favoring the practical application of the developed models.

Keywords

Acoustic feature, normalization, standardization, classification algorithms, wood-boring insects, larvae, knots, cross-validation.

Índice

Capítulo 1

Introducción

La detección de plagas en estructuras de madera sigue siendo un reto importante en sectores como la conservación del patrimonio, la construcción o el comercio internacional. En concreto, los ataques de insectos xilófagos, como las larvas del Hylotrupes bajulus, pueden provocar daños graves sin dejar señales visibles hasta que ya es demasiado tarde. Esto complica tanto la identificación del problema como la intervención a tiempo, especialmente cuando se busca evitar métodos invasivos que puedan deteriorar aún más la pieza afectada.

Ante esta necesidad, en los últimos años han surgido nuevas líneas de investigación centradas en la detección acústica de actividad biológica en el interior de la madera. Esta técnica permite registrar el sonido que generan las larvas al alimentarse y moverse, lo cual da pie al desarrollo de sistemas de diagnóstico más precisos, seguros y respetuosos con el material.

En este contexto se enmarca este Trabajo de Fin de Grado. Partiendo de un sistema de adquisición acústica basado en micrófonos MEMS y técnicas avanzadas de procesado de señal, se plantea un enfoque de clasificación automática que permita distinguir los sonidos generados por larvas y captados en el punto de origen (la zona donde se produce la actividad) de los que, debido a la propagación del sonido a través del material, especialmente por los nudos de la madera, son registrados en posiciones distintas a su origen real. El objetivo final es contribuir a la mejora de estos sistemas, haciéndolos más eficaces y accesibles en escenarios reales.

1.1. Objetivos

El objetivo principal de este trabajo es explorar qué algoritmos de machine learning (Random Forest, K-Nearest Neighbors y Support Vector Machine), junto con qué combinaciones de hiperparámetros, ofrecen mejores resultados en distintas métricas de evaluación, con el fin de identificar qué tipo de modelos son más eficaces para clasificar las distintas detecciones de larvas y nudos. Estos

sonidos han sido generados por larvas del escarabajo de la carcoma grande (Hylotrupes bajulus), que se alimentan en el interior de vigas de madera.

Para alcanzar este objetivo, se han seguido una serie de pasos encaminados a la preparación, procesamiento y análisis de los datos, que se detallan a continuación:

- Analizar la estructura de las capturas de sonido y, en función de la información contenida en ellas, determinar qué características acústicas pueden resultar relevantes para el estudio, tanto en el dominio del tiempo como en el de la frecuencia.
- Implementar en LabVIEW los esquemas necesarios para procesar dichas capturas, almacenadas en una base de datos, y extraer de ellas los valores correspondientes a las diferentes características seleccionadas.
- Aplicar un proceso de filtrado para identificar tanto las características que aportan información significativa al análisis como aquellas que no contribuyen de forma útil.
- Escalar los datos seleccionados mediante dos métodos diferentes, normalización y estandarización, con el objetivo de comparar el impacto que tiene cada forma de escalado en el rendimiento de los algoritmos de machine learning.
- Entrenar y evaluar los tres algoritmos seleccionados (Random Forest, K-Nearest Neighbors y Support Vector Machine) utilizando ambos tipos de escalado, tanto en una clasificación multiclase (entre distintas larvas) como en una binaria (entre larva y ruido).
- Comparar el rendimiento de los distintos modelos mediante métricas que permitan representar gráficamente los resultados, incluyendo curvas ROC y curvas de aprendizaje, facilitando de esta manera la interpretación visual de los comportamientos observados.

Para estructurar adecuadamente la información obtenida a lo largo del estudio, esta memoria se inicia con un marco teórico, en el que se explican los conceptos fundamentales que serán desarrollados y evaluados de forma práctica en las secciones posteriores. Seguidamente, se detallan los pasos llevados a cabo en el proceso de preprocesamiento de los datos, así como la implementación realizada en LabVIEW, poniendo especial énfasis en el desarrollo de los esquemas empleados para la extracción de características. Finalmente, se presentan los resultados obtenidos en las distintas pruebas realizadas con los algoritmos seleccionados, junto con las comparativas entre ellos y las conclusiones extraídas a partir del análisis de los mismos.

Capítulo 2

Motivación y primeros pasos del estudio

La motivación que inició esta línea de investigación y con la que este trabajo continúa es la dificultad con la que se lidia actualmente en el ámbito de la conservación y restauración de obras de arte realizadas en madera, así como de controlar las infecciones no intencionales en el comercio internacional de madera, en el montaje y uso de madera infectada, entre otras posibles aplicaciones. Este estudio se basa en las ocasiones en que esto se produce por ataques a la madera de larvas de insectos xilófagos que se alimentan de ella. De cara a poder identificar la posición de los insectos en el interior de las obras de arte sin la necesidad de su manipulación física y por tanto aumento de su deterioro, se propone desarrollar a partir del sonido que emiten dichos insectos, un algoritmo que sea capaz de identificarlos y posicionarlos. Los primeros pasos que se han dado en este estudio han sido los siguientes, recogidos en [1].

Se ha implementado un sistema avanzado para identificar larvas de insectos que dañan la vigas estructurales de madera. Este sistema consiste en una red amplia de micrófonos MEMS capacitados para capturar los sonidos emitidos por las larvas de dichos insectos. Mediante técnicas de filtrado de frecuencia y análisis de energía, se pueden detectar las larvas activas. Posteriormente, el análisis de las señales segmentadas permite localizar espacialmente a las larvas usando beamforming. Las pruebas realizadas han demostrado la efectividad del sistema para detectar y rastrear la trayectoria de las larvas dentro de madera de pino.

El sistema de adquisición que se ha utilizado en el estudio consiste en tres elementos principales:

1. Un conjunto de micrófonos MEMS (Micro-Electro-Mechanical Systems) que conforman un array acústico.
2. Un sistema de adquisición y preprocesamiento basado en FPGA/Procesa-

dor.

3. Una aplicación de análisis, detección y visualización basada en PC.

El sistema de adquisición utiliza una plataforma con micrófonos MEMS para capturar datos de actividad de larvas de forma sincronizada. El software desarrollado controla la captura, detecta la actividad de las larvas, almacena las señales para su segmentación, localiza los sonidos mediante algoritmos de conformación de haces, y muestra una imagen 2D con las posiciones de las larvas detectadas. Para esto se ha desarrollado un software basado en LabVIEW 2021 que controla la captura sincronizada de datos, detecta y almacena la actividad de las larvas, localiza los sonidos mediante algoritmos de conformación de haz, y muestra una imagen 2D con las posiciones de las larvas.

Para la configuración de la prueba se precisó de una cámara anecoica y un sistema de medición basado en una matriz de micrófonos MEMS. Además, se implantaron 6 larvas Hylotrupes bajulus L en una estructura de pino silvestre con ciertas condiciones ambientales. En la Fig. 2.1 puede verse un esquema con las posiciones de las larvas en la estructura de madera indicada. Se cerraron los agujeros desde los que se insertaron las larvas en la madera y se marcaron sus posiciones. Se mantuvieron las larvas durante 30 días para simular condiciones realistas y posteriormente se tomaron medidas durante 2 meses. Se caracterizaron tanto la duración de las detecciones, de aproximadamente 1 ms, como el espectro de las mismas. Se obtuvieron un total de 50000 detecciones.

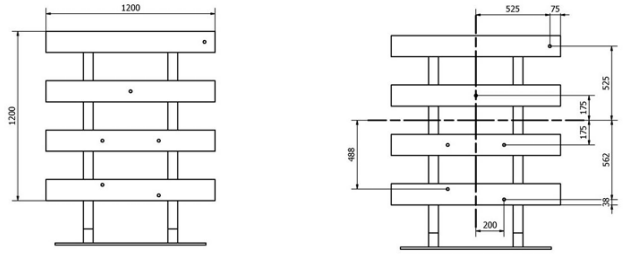


Figura 2.1: Estructura de pino para la captura y distribución de las larvas a lo largo de la viga.

Tras la adquisición de las señales acústicas generadas por las larvas, se empleó un algoritmo especializado para procesar y analizar los patrones característicos de su actividad. Este algoritmo utiliza técnicas avanzadas de procesamiento de señales para detectar de manera fiable las señales acústicas producidas por las larvas al alimentarse dentro de las vigas de madera. Posteriormente, se generaron segmentos cortos de señales que permitieron la localización espacial de las larvas mediante un algoritmo de conformación de haz basado en la suma de retardos.

La implementación de este algoritmo demostró la capacidad del sistema para

detectar y localizar múltiples larvas de Hylotrupes bajulus L. dentro de las piezas de madera, así como para seguir su trayectoria interna. La combinación de técnicas de procesamiento de señales y algoritmos de conformación de haces permitió una detección precisa y una localización espacial efectiva de las larvas, lo cual es crucial para el éxito de la aplicación en la detección de plagas de insectos xilófagos en entornos reales.

Tras un proceso de análisis de dos meses se obtuvieron 50,000 capturas de la actividad de las larvas, lo que representa el 2 % del total de las capturas que se realizarón, que fue 2,5 millones. Mediante un histograma 2D se mostraron las posiciones de las larvas basadas en imágenes acústicas, identificando 7 zonas de mayor actividad. Se tuvo ern cuenta que, la complejidad de la propagación del sonido en la madera, debido a su variabilidad, afecta la precisión de la localización. Todas las larvas sobrevivieron y crearon túneles de entre 110 y 290 mm de longitud. A pesar de algunas desviaciones en la fibra que causaron errores de localización, el sistema logró identificar correctamente las posiciones de la mayoría de las larvas. Como conclusión del estudio se desarrolló un sistema acústico económico y no invasivo con micrófonos MEMS para detectar larvas de cerambícidos en madera estructural, permitiendo identificar la localización de múltiples individuos con precisión.

En cuanto a las grietas y nudos que presenta la madera, se identificó que pueden llegar a desviar el punto de salida del sonido, afectando la localización precisa de las larvas dentro de la madera. Aun así, los errores de localización se mantienen dentro del área afectada mínima.

Otras líneas de investigación ajenas a la que se tratará en este trabajo podrían explorar cómo el tamaño de las larvas influye en el umbral de detección del sistema. También se podrían estudiar los patrones de alimentación y galerías de otros xilófagos, como anóbidos y termitas, evaluando su desempeño en condiciones de campo.

Capítulo 3

Marco teórico

3.1. LabVIEW

Para la extracción de características acústicas se ha empleado LabVIEW.[2] LabVIEW o 'Laboratory Virtual Instrument Workbench' es un entorno de desarrollo creado por la empresa National Instruments que se dedica a la implementación de sistemas de pruebas y control. A día de hoy, LabVIEW ofrece una solución muy completa para aplicaciones en tecnología e ingeniería, y es compatible con Windows, UNIX y GNU/Linux.

Una característica de LabVIEW a resaltar es su capacidad para conectarse con una amplia gama de hardware, siendo capaz de integrar diferentes instrumentos y dispositivos en un solo entorno de desarrollo. Además de esto, la interfaz que proporciona LabVIEW permite al usuario monitorizar pruebas en tiempo real.

Otra de sus funcionalidades es su amplia oferta de funciones de análisis de ingeniería, las cuales están integradas en el entorno de desarrollo y permiten la realización de cálculos complejos sin recurrir a herramientas externas.

El lenguaje de programación que se emplea en LabVIEW es gráfico, de manera que mediante bloques el usuario es capaz de crear scripts, en vez de emplear código escrito. Por esto muchas veces se considera más intuitivo, sobretodo a la hora de crear sistemas avanzados sin necesidad de tener un conocimiento avanzado en código escrito. Otra de las ventajas de LabVIEW es que permite el manejo de diversas interfaces de comunicación. Esto permite conectar y controlar múltiples dispositivos o sistemas. También dispone de herramientas destinadas a la adquisición y tratamiento de imágenes, el control de movimiento y la programación de FPGAs.

Otras herramientas presentes son las gráficas para el procesamiento digital de señales, su visualización y el manejo de datos dinámicos. La depuración gráfica integrada y el control de código fuente facilitan la identificación y resolución de problemas durante el desarrollo.

Como conclusión, LabVIEW se considera una herramienta poderosa y versátil

que combina un entorno de desarrollo gráfico con una amplia gama de funciones y compatibilidad con diversos lenguajes y hardware. Pretende simplificar el desarrollo de aplicaciones complejas e integrarse con otros sistemas y tecnologías en múltiples aplicaciones de ingeniería.

3.2. Python

Una vez extraídas las características acústicas, se ha empleado Python para la normalización de los datos y para el desarrollo de los scripts que permitirán probar algoritmos de 'machine learning'.[3][4]

Python es un lenguaje de programación de alto nivel creado en 1991. La prioridad de su diseño es tanto la legibilidad como la claridad del código, haciendo de este uno de los lenguajes de programación considerados más intuitivos actualmente. Uno de los cambios empleados por Python para mejorar la organización del código es el uso de la indentación en lugar de llaves.

Una de las ventajas de este lenguaje es que es un lenguaje interpretado, es decir, no necesita una etapa de compilación previa para ejecutar el código, aligerando así el tiempo que requiere correr el programa. Python es además, compatible tanto con Windows como con macOS y Linux.

Otra de las ventajas de este lenguaje es que Python ofrece datos de tipo dinámico, es decir, que permite que las variables cambien de tipo durante la ejecución del código. Ademas, es un lenguaje provisto de una gran cantidad de librerías y módulos. De esta manera los usuarios pueden acceder a funciones muy complejas ya implementadas. Algunas de sus librerias están destinadas al análisis de datos son Pandas y NumPy.

A día de hoy Python es utilizado en una gran cantidad de campos, así como el de desarrollo Web, análisis de datos para poder manejar grandes cantidades de los mismos, inteligencia artificial, automatización, seguridad, desarrollo de software... Gracias a esto y a su simplicidad e intuitividad, es ampliamente utilizado para aplicaciones en ingeniería y se ha convertido en una herramienta esencial en el mundo de la programación.

3.2.1. Scikit-Learn

Scikit-Learn es una librería de código abierto empleada en apredizaje automático en Python.[5] Las herramientas que contiene se destinan a la modelización predictiva y el análisis de datos, y contiene algoritmos de clasificación, regresión, clustering y reducción de dimensionalidad. Algunas de las ventajas que ofrece es su simplicidad y facilidad de uso, su integración con otros ecosistemas de Python y su amplia gama de algoritmos.

Las funciones más importantes de Scikit-Learn son:

- Procesamiento de datos
- Selección de características
- Algoritmos de aprendizaje automático
- Validación de modelos
- Ajuste de modelos
- Interfaz con otras librerías

3.3. Características acústicas

Para poder obtener los datos finales requeridos para entrenar los algoritmos de machine learning que se van a probar, es necesario saber qué información es interesante y cómo extraerla de las capturas de audio de las larvas.[6] Existen una serie de características acústicas que son propias de cada señal de audio, que describen su estructura, contenido y comportamiento y que pueden ayudar a diferenciar cada una de las capturas ya obtenidas.

Las características acústicas pueden clasificarse según el dominio en el que se extraen, siendo algunos de los más relevantes el temporal, el frecuencial, el cepstral, el modulado o el de transformada de Fourier de tiempo corto (STFT). A pesar de esto, para este estudio concreto, se han considerado relevantes características que se extraen tanto en el dominio temporal como en el dominio frecuencial, sin ser todas las pertenecientes a estos dominios las extraídas, puesto que, por ejemplo, características que son interesantes para señales moduladas, no van a proporcionar una información relevante en este caso.

Las características obtenidas mediante su implementación en LabVIEW son las siguientes, cuya explicación detallada se encuentra en el apartado 4.1 de esta memoria:

Dominio frecuencial:

- Centroide espectral
- Envergadura espectral
- Asimetría espectral
- Curtosis espectral
- Roll-of espectral
- Pendiente espectral
- Decaimiento espectral
- Centroide del espectro de audio
- Dispersión del espectro de audio
- Planicidad del espectro de audio

Dominio temporal:

$$\begin{cases} \text{Número de cruces por cero} \\ \text{Pitch} \\ \text{Tiempo de ataque logarítmico} \end{cases}$$

Capítulo 4

Parte experimental

4.1. Extracción de características

4.1.1. Centroide espectral

El centroide espectral se emplea para caracterizar de manera numérica la forma del espectro de una señal. Define el punto central del espectro, calculado mediante la ponderación de las frecuencias presentes en el espectro en función de sus respectivas amplitudes o energías. Proporciona lo que se puede determinar como "punto medio"del espectro, reflejando dónde se concentra la mayor parte de la energía de la señal en términos de frecuencia.

El centroide se calcula según la expresión:

$$C = \frac{\sum_i A(f_i)}{\sum_i f_i \cdot A(f_i)} \tag{4.1}$$

Donde:

- $\sum_i$ representa la suma sobre todos los índices i.
- f_i representa cada índice frecuencial.
- $A(f_i)$ representa la amplitud de cada f_i.

Y se ha implementado en LabVIEW siguiendo la ecuación 4.1 mediante el esquema visto en la Fig. 4.1.

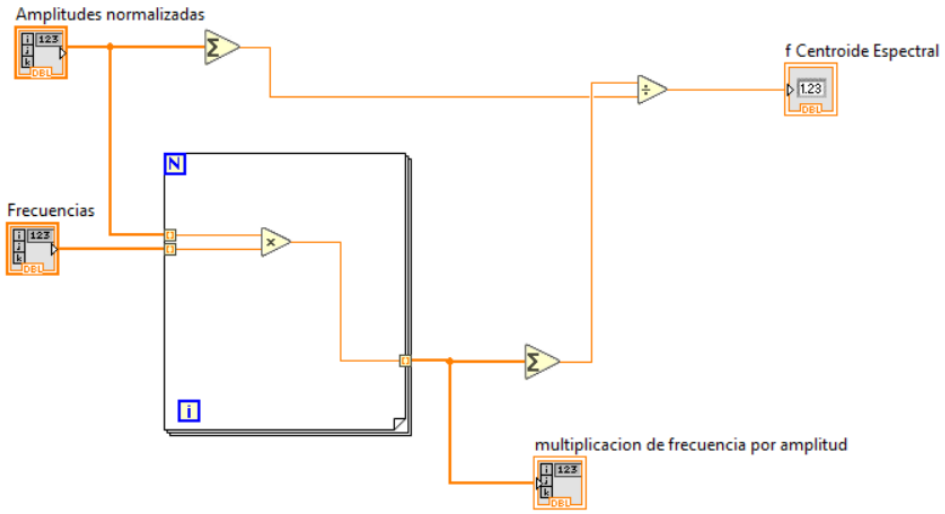


Figura 4.1: Implementación en LabVIEW del centroide espectral.

4.1.2. Envergadura espectral

La envergadura espectral se refiere a lo que llamamos en términos estadísticos 'varianza', y depende del centroide espectral. Define la dispersión y concentración de la energía del espectro de la señal en torno a la frecuencia media, es decir, del centroide. Valores más altos de dicho parámetro indican que la energía está más dispersa en torno al centroide espectral y viceversa.

La envergadura espectral se calcula según la expresión:

$$\sigma^2 = \frac{\sum_i (f_i - \bar{f})^2 \cdot A(f_i)}{\sum_i A(f_i)} \tag{4.2}$$

Donde:

- $\sum_i$ representa la suma sobre todos los índices i.
- f_i representa cada índice frecuencial.
- $\bar{f}$ representa el centroide de la frecuencia.
- $A(f_i)$ representa la amplitud de cada f_i.

y se ha implementado en LabVIEW siguiendo la ecuación 4.2 mediante el esquema visto en la Fig. 4.2.

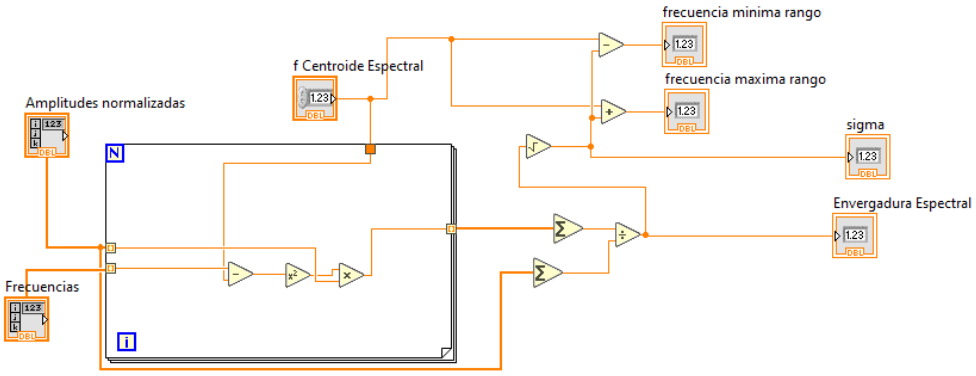


Figura 4.2: Implementación en LabVIEW de la envergadura espectral.

4.1.3. Asimetría espectral

La asimetría espectral, que se relaciona con el centroide espectral o media, proporciona una medida numérica de cómo se distribuye la energía del espectro de una señal alrededor de su centroide. Esta medida permite interpretar la simetría o asimetría del espectro de la siguiente manera:

$$\begin{cases} \gamma_1 = 0 & \rightarrow \text{Distribución simétrica} \\ \gamma_1 < 0 & \rightarrow \text{Distribución asimétrica hacia la derecha} \\ \gamma_1 > 0 & \rightarrow \text{Distribución asimétrica hacia la izquierda} \end{cases}$$

La asimetría espectral se calcula según las expresiones:

$$m_3 = \frac{\sum_i (f_i - \bar{f})^3 \cdot A(f_i)}{\sum_i A(f_i)} \tag{4.3}$$

$$\gamma_1 = \frac{m_3}{\sigma^3} \tag{4.4}$$

Donde:

- m_3 representa el tercer momento centrado de la distribución.
- $\bar{f}$ representa el centroide de la frecuencia.
- $A(f_i)$ representa la amplitud de cada f_i.
- σ es la desviación estándar.

y se ha implementado en LabVIEW siguiendo las ecuaciones 4.3 y 4.4 mediante en esquema visto en la Fig. 4.3.

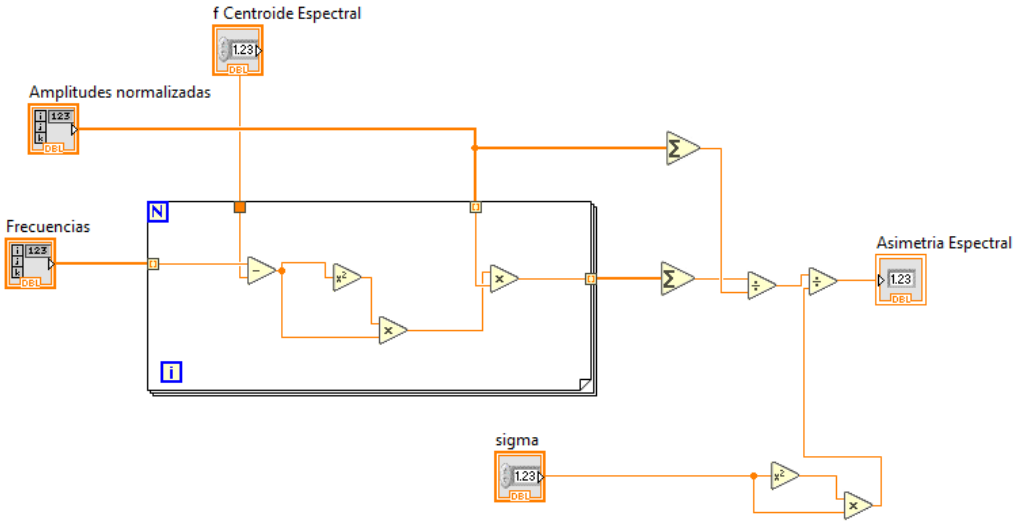


Figura 4.3: Implementación en LabVIEW de la asimetría espectral.

4.1.4. Curtosis espectral

La curtosis espectral indica numéricamente el grado de concentración de energía alrededor del centroide espectral o media de una señal. Un valor de curtosis espectral alto indica una concentración mayor de energía alrededor de la media. La curtosis espectral se interpreta numéricamente de la siguiente manera:

$$\begin{cases} \gamma_2 = 3 & \rightarrow \text{Distribución normal} \\ \gamma_2 < 3 & \rightarrow \text{Distribución más plana} \\ \gamma_2 > 3 & \rightarrow \text{Distribución más cónica} \end{cases}$$

y visualmente, como se observa en la Fig. 4.4.

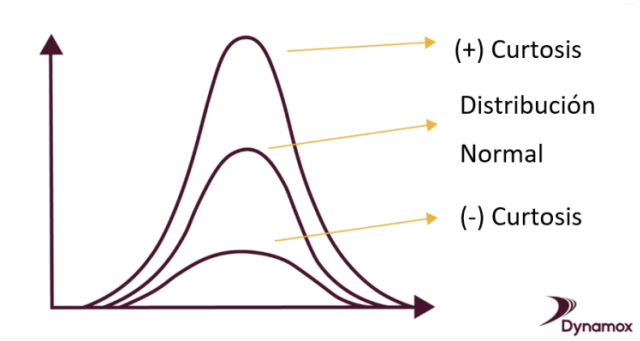


Figura 4.4: Esquema de la interpretación de la curtosis espectral [7].

La curtosis espectral se calcula según las expresiones:

$$m_4 = \frac{\sum_i (f_i - \bar{f})^4 \cdot A(f_i)}{\sum_i A(f_i)} \tag{4.5}$$

$$\gamma_2 = \frac{m_4}{\sigma^4} \tag{4.6}$$

Donde:

- m_4 representa el cuarto momento centrado de la distribución.
- $\bar{f}$ representa el centroide de la frecuencia.
- $A(f_i)$ representa la amplitud de cada f_i.
- σ es la desviación estándar.

y se ha implementado en LabVIEW siguiendo las ecuaciones 4.5 y 4.6 mediante el esquema visto en la Fig. 4.5.

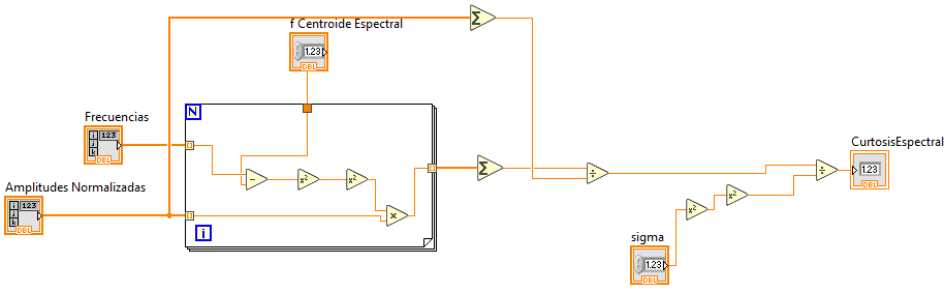


Figura 4.5: Implementación en LabVIEW de la curtosis espectral.

4.1.5. Roll-off espectral

El roll-of espectral es una característica acústica que considera el espectro de la señal como una función de densidad de probabilidad. De esta manera, el valor de roll-of es aquella frecuencia para la cual la función de probabilidad acumulada supera cierto umbral, es decir, la frecuencia en la cual se alcanza cierto porcentaje acumulado de la energía de la señal. Esta característica proporciona información sobre la rapidez o la lentitud con la que decae la energía de una señal, y por tanto también sobre la concentración de la misma.

El roll-of espectral se calcula según la expresión:

$$\sum_{f=0}^{f_c}(A(fi))^2 = \gamma \sum_{f=0}^{\frac{f_s}{2}}(A(fi))^2 \tag{4.7}$$

Donde:

- f_c es la frecuencia de roll-off.
- f_s es la frecuencia de muestreo.
- $(A(fi))^2$ es la amplitud de la señal a la frecuencia f.
- γ es la probabilidad acumulada deseada, típicamente 0.85 o 0.95, que indica el porcentaje de energía acumulada.

En este caso y teniendo en cuenta nuestras señales, la γ elegida ha sido 0.85, y la implementación práctica en LabVIEW difiere ligeramente de la original, puesto que ha sido particularizada para nuestro caso concreto.

Para poder implementarlo, se han seguido los siguientes pasos:

1. Cálculo de la densidad espectral de potencia de la señal.

$$DEP = \sum_i (A(fi))^2 \tag{4.8}$$

2. Cálculo del porcentaje de energía que representa cada muestra frecuencial.

$$\%\, Energia\, por\, muestra = \frac{(A(fi))^2}{\sum_i (A(fi))^2} \cdot 100 \tag{4.9}$$

3. Cálculo iterativo de la suma del porcentaje de energía que representa cada muestra frecuencial hasta que alcanza el γ seleccionado.

$$Amplitud\, umbral = \sum_i \frac{(A(fi))^2}{\sum_i (A(fi))^2} \cdot 100 \tag{4.10}$$

Cuando

$$x < 0{,}85 \cdot 100 \tag{4.11}$$

4. A partir de la amplitud umbral, determinar con qué componente frecuencial se corresponde la muestra anterior, puesto que se trata de obtener la última muestra frecuencial antes de alcanzar el umbral de energía.

Se ha implementado en LabVIEW siguiendo las ecuaciones 4.8, 4.9 y 4.10 mediante el esquema visto en la Fig. 4.6.

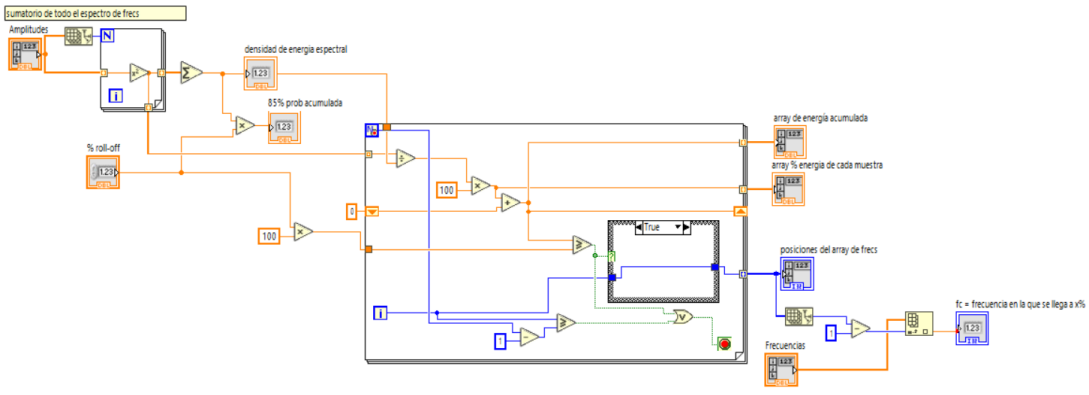


Figura 4.6: Implementación en LabVIEW del Roll-of espectral.

4.1.6. Pendiente espectral

La pendiente espectral se emplea para identificar cómo disminuye la amplitud de cada frecuencia a medida que van aumentando las frecuencias en el espectro.

De esta manera proporciona información sobre la forma del espectro y sobre la distribución de la energía de la señal. Se realiza una regresión lineal a las amplitudes asociadas a cada frecuencia del espectro, de manera que se obtiene una recta, obteniendo tanto su pendiente como su ordenada en el origen.

La pendiente espectral se calcula según la expresión:

$$\hat{a}(f) = \text{slope} \cdot f + \text{constant} \tag{4.12}$$

$$\text{slope} = \frac{1}{\sum_i A(fi)} \cdot \frac{N \sum_i f(i) \cdot A(fi) - \sum_i f(i) \cdot \sum_i A(fi)}{N \sum_i f(i)^2 - (\sum_i f(i))^2} \tag{4.13}$$

Donde:

- *slope* representa la pendiente de la recta.
- *constant* representa la ordenada en el origen.
- fi cada índice frecuencial.
- $A(f_i)$ representa la amplitud de cada f_i.
- N es representa el número total de términos sobre los que hace la suma.

y se ha implementado en LabVIEW siguiendo las ecuaciones 4.12 y 4.13 mediante el esquema visto en la Fig. 4.7.

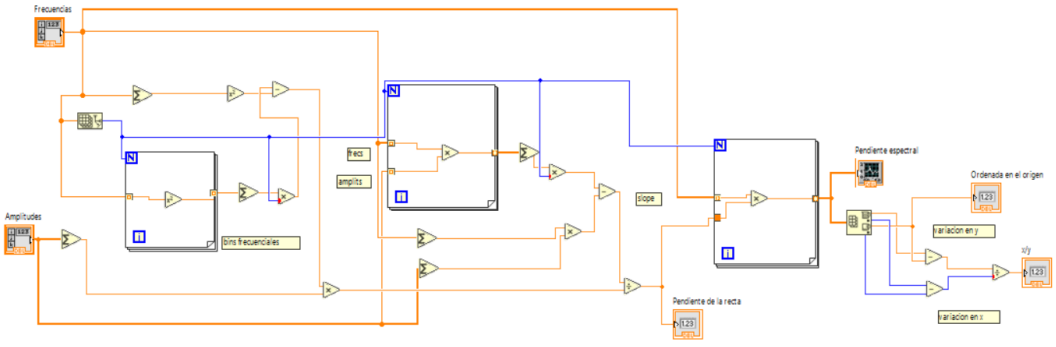


Figura 4.7: Implementación en LabVIEW de la pendiente espectral.

4.1.7. Decaimiento espectral

El decaimiento espectral está relacionado con la pendiente espectral, puesto que ambos tratan de caracterizar la manera en la que la energía del espectro se distribuye a través de sus frecuencias. Concretamente el decaimiento espectral, en función de sus valores permite saber si la distribución de la energía a lo largo del espectro se encuentra en mayores o en menores frecuencias.

El decaimiento espectral se calcula según la expresión:

$$Decaimiento = \frac{1}{\sum_{i=2}^{N} A(fi)} \cdot \sum_{i=2}^{N} \frac{A(fi) - A(1)}{i-1} \tag{4.14}$$

Donde:

- i representa el contador de iteraciones
- fi cada índice frecuencial.
- $A(f_i)$ representa la amplitud de cada f_i.
- N es representa el número total de términos sobre los que hace la suma.

y se ha implementado en LabVIEW siguiendo la ecuación 4.14 mediante el esquema visto en la Fig. 4.8.

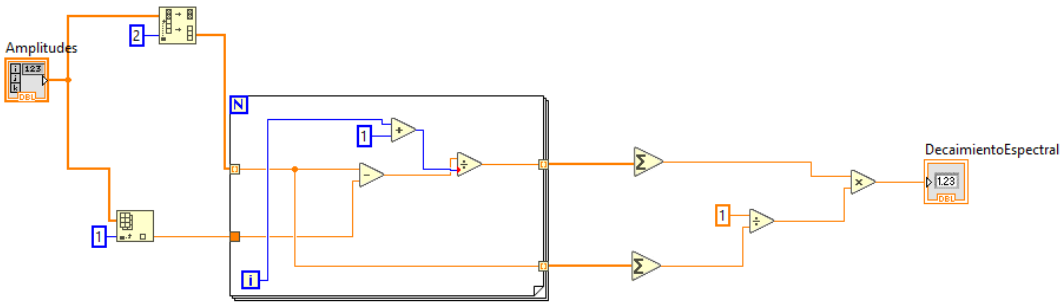


Figura 4.8: Implementación en LabVIEW del decaimiento espectral.

4.1.8. Centroide del espectro de audio

El centroide del espectro de audio es una adaptación del centroide espectral. En este caso se emplea una escala logarítmica de frecuencia para poder obtener resultados más precisos y describir la media de la señal de una manera más detallada. A pesar de que su empleo se suele dar en señales de audio más compatibles con la percepción humana, se ha implementado.

Para este caso concreto, se ha adaptado el cálculo de esta característica de manera que se tiene en cuenta todo el espectro de frecuencias de las señales. El centroide del espectro de audio se calcula según la expresión:

$$\text{ASC} = \frac{\sum_i \log_2(\frac{f(i)}{1000}) \cdot (A(fi))^2}{\sum_i (A(fi))^2} \tag{4.15}$$

Donde:

- fi representa cada índice frecuencial.
- $A(f_i)$ representa la amplitud de cada f_i.

y se ha implementado en LabVIEW siguiendo la expresión 4.15 mediante el esquema visto en la Fig. 4.9.

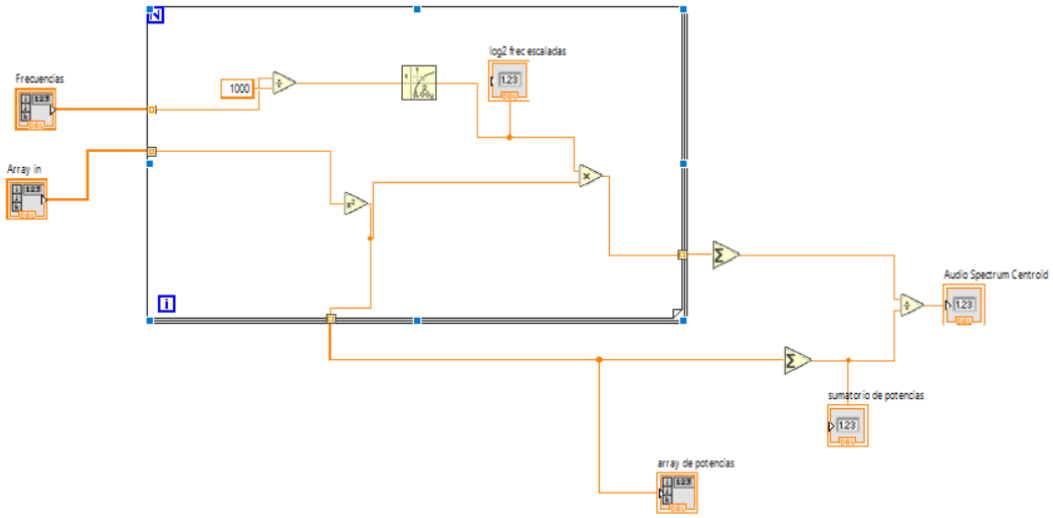


Figura 4.9: Implementación en LabVIEW del centroide del espectro de audio.

4.1.9. Dispersión del espectro de audio

La dispersión del espectro de audio, al igual que en el caso anterior, trata de dar una descripción más detallada, en este caso de la envergadura espectral. En este caso también se consigue empleando la escala de frecuencias logarítmicas. La disperisón del espectro de audio se calcula según la ecuación:

$$\text{ASC} = \sqrt{\frac{\sum_i [\log_2(\frac{f(i)}{1000}) - ASC]^2 \cdot (A(fi))^2}{\sum_i (A(fi))^2}} \tag{4.16}$$

Donde:

- fi representa cada índice frecuencial.
- $A(f_i)$ representa la amplitud de cada f_i.
- ASC representa el centroide espectral de audio.

y se ha implementado en LabVIEW siguiendo la ecuación 4.16 mediante el esquema visto en la Fig. 4.10.

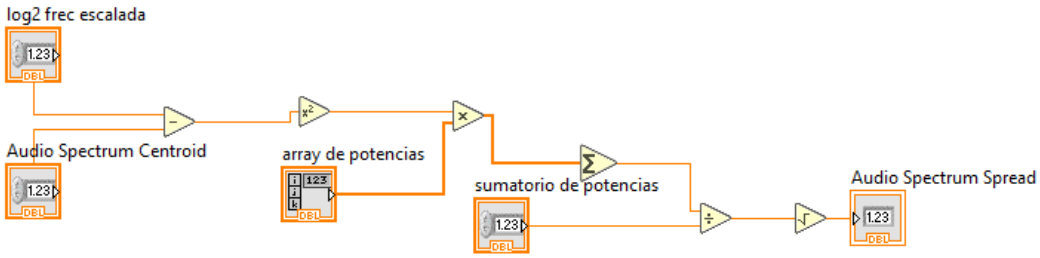


Figura 4.10: Implementación en LabVIEW de la dispersión del espectro de audio.

4.1.10. Planicidad del espectro de audio

La planicidad del espectro de audio proporciona información sobre la similitud de un determinado espectro con otro totalmente plano. Se ha tratado el espectro completo como una única banda a la que se le ha aplicado el ratio entre la media geométrica y la media aritmética de la potencia de sus coeficientes. La planicidad del espectro de audio se calcula según la expresión:

$$\mathrm{ASF} = \frac{\sqrt[n]{\prod_i (A(fi))^2}}{\frac{1}{n} \cdot \sum_i (A(fi))^2} \tag{4.17}$$

Donde:

- fi representa cada índice frecuencial.
- $(A(f_i))^2$ representa la potencia asociada a cada f_i.
- n representa el número total de muestras del array.

y se ha implementado en LabVIEW siguiendo la ecuación 4.17 mediante el esquema visto en la Fig. 4.11.

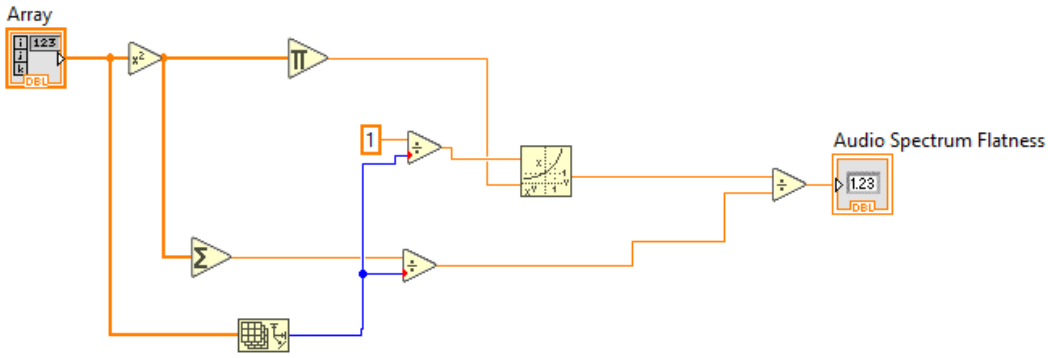


Figura 4.11: Implementación en LabVIEW de la planicidad del espectro de audio.

4.1.11. Nº de cruces por cero

El número de cruces por cero es una característica acústica calculada en el dominio del tiempo. La información que proporciona es el número de veces que la señal cruza el eje Y a lo largo del tiempo en el que esta se extienda. Esto ayuda a caracterizar la señal, puesto que las frecuencias altas presentan un mayor número de cruces por cero que las bajas. A su vez, el ruido en las señales se presenta con un número elevado de estos cruces, mientras que en un período de silencio este será considerablemente menor. La fórmula general para el cálculo en una señal discreta de audio es la siguiente:

$$ZCR \cdot f_s = \frac{f_s}{2N} \sum_{n=1}^{N-1} |\text{sign}(x[n]) - \text{sign}(x[n-1])| \tag{4.18}$$

donde:

- f_s representa la frecuencia de muestreo.
- N representa la longitud de la señal $x[n]$.
- $\text{sign}(x[n])$ representa la función signo de la muestra $x[n]$.

Para este caso concreto, se ha implementado en LabVIEW mediante el esquema visto en la Fig. 4.12 y en la Fig. 4.13 siguiendo los pasos:

1. La señal se hace pasar por el bloque 'Zero Crossing PtByPt'. Este detecta cruces por cero de los puntos de datos de entrada. El cruce se vuelve VERDADERO inmediatamente después de que ocurra la transición.
2. En caso de que ocurra el cruce, se suma al contador

3. En caso de que no se produzca un cruce, se sigue evaluando el resto de la señal.

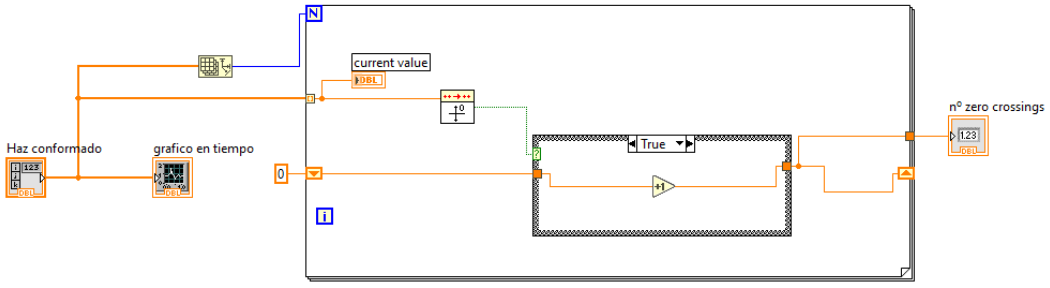


Figura 4.12: Implementación en LabVIEW del número de cruces por zero.

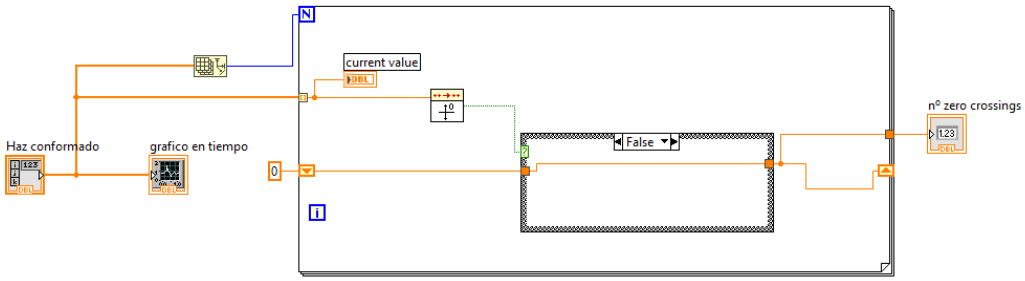


Figura 4.13: Implementación en LabVIEW del número de cruces por zero.

4.1.12. Pitch

El pitch es una característica para el análisis de señales de audio que describe la percepción de la frecuencia fundamental de un sonido, representando la altura tonal en una escala de grave a agudo. Se puede decir que el pitch se refiere a la 'altura' percibida de un sonido, que está directamente relacionada con la frecuencia fundamental predominante en la señal. El cálculo del pitch se emplea en muchas aplicaciones de procesamiento de señales de audio y reconocimiento de patrones, ya que proporciona información fundamental sobre el contenido tonal de la señal. Uno de los métodos más empleados, y el elegido en este caso es la autocorrelación.
Normalmente, los métodos utilizados para estimar el pitch implican analizar patrones temporales de la señal para identificar la periodicidad o la frecuencia

fundamental de interés. A la hora de emplear el método de la autocorrelación para hallar el pitch, es necesario que esta no sea aleatoria, pues son necesarios patrones repetitivos para su utilización. En el caso de una señal repetitiva, la autocorrelación alcanza un máximo cuando un determinado desplazamiento coincide con el período de la señal. Así pues, el valor máximo de la autocorrelación indica que el desplazamiento efectuado coincide con el período fundamental de la señal. La autocorrelación se calcula según la expresión:

$$R_x(m) = E\{x[n]x[n-m]\} \tag{4.19}$$

Donde:

- $R_x(m)$: Representa la función de autocorrelación de la señal x en el desplazamiento m.
- $E\{\cdot\}$: Representa el operador de valor esperado, que calcula el promedio tras multiplicar dos señales.
- $x[n]$: Representa la señal x en el tiempo discreto n.
- $x[n-m]$: Representa la señal x en el tiempo discreto $n-m$, es decir, la señal x desplazada m muestras en el tiempo.

y se ha implementado en LabVIEW siguiendo la ecuación 4.19 mediante el esquema visto en la Fig. 4.14.

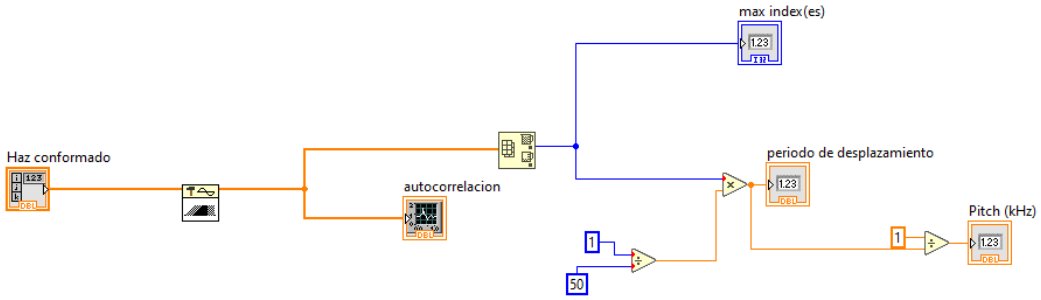


Figura 4.14: Implementación en LabVIEW del pitch.

1. Cálculo de la correlación mediante el bloque 'AutoCorrelation'.
2. Cálculo del período de desplazamiento a partir de la multiplicación del máximo valor de la autocorrelación con el período de la señal, teniendo en cuenta que la frecuencia de muestreo empleada ha sido de 50kHz.
3. Obtención del pitch (kHz) como inverso del período.

4.1.13. Tiempo de ataque logarítmico

El tiempo de ataque logarítmico es una característica temporal de la señal que se obtiene a partir de su envolvente. Se emplea para medir el tiempo que tarda la señal en alcanzar cierto nivel (normalmente el máximo), es decir, cuando su amplitud es mayor. Para el cálculo de esta característica es necesario determinar un punto de inicio a partir del cual empezar a evaluar, y normalmente este es el 25 % del valor RMS de la señal. El tiempo de ataque logarítmico se calcula según la expresión:

$$\text{LogAttackTime} = \log_{10}(\text{stop}_{\text{attack}} - \text{start}_{\text{attack}}) \tag{4.20}$$

Donde:

- $\text{start}_{\text{attack}}$ es el instante de tiempo en el que comienza el ataque.
- $\text{stop}_{\text{attack}}$ es el instante de tiempo en el que el ataque termina.

y se ha implementado en LabVIEW siguiendo la ecuación 4.20 mediante el esquema visto en la Fig. 4.15 y en la Fig. 4.16.

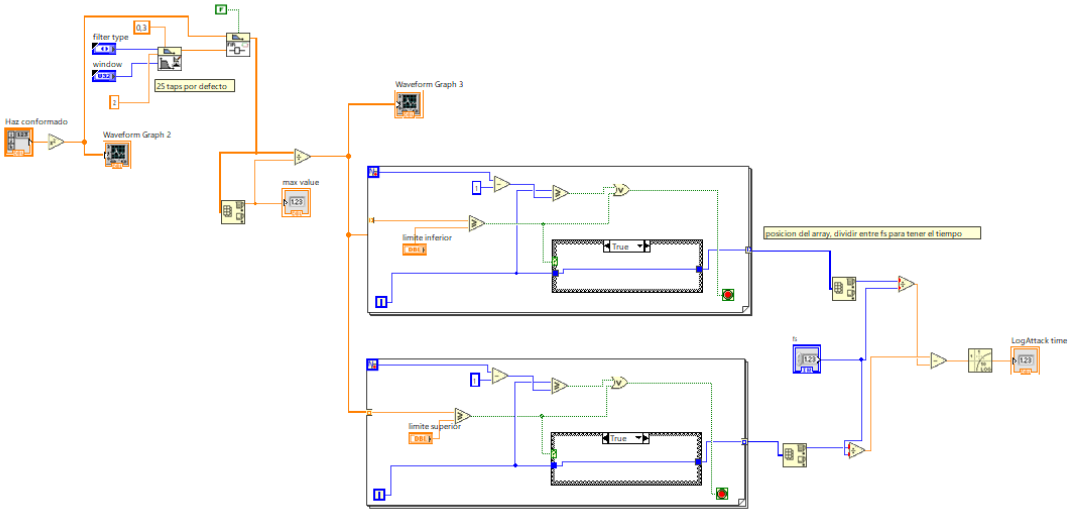


Figura 4.15: Implementación en LabVIEW del tiempo de ataque logarítmico.

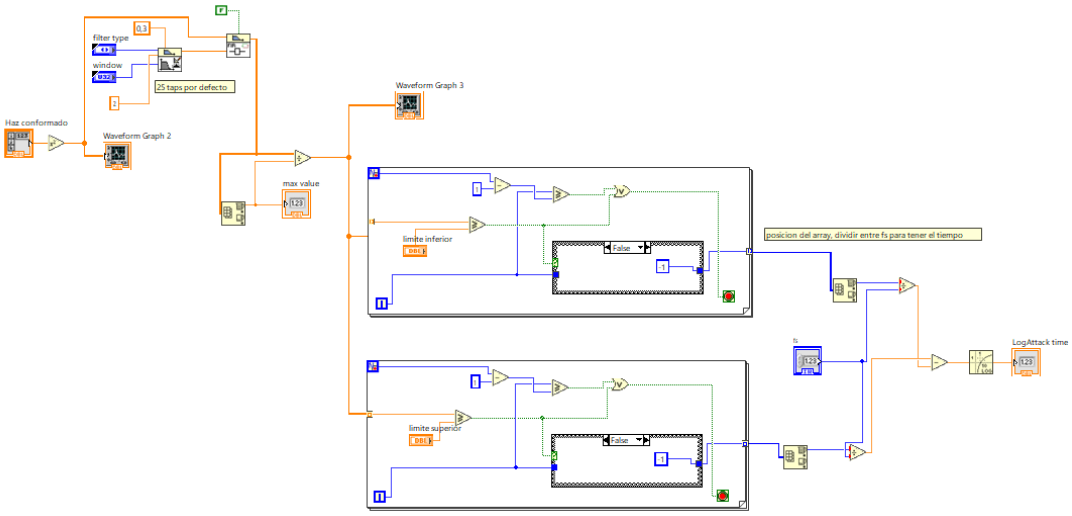


Figura 4.16: Implementación en LabVIEW del tiempo de ataque logarítmico.

Para el caso concreto de este estudio se han seguido los siguientes pasos:

1. Cálculo de la envolvente de la señal y normalización de la misma.
2. Obtención de dos arrays. El primero con las posiciones del array inicial mayores al límite inferior y el segundo con las posiciones del array inicial mayores al límite superior.
3. Obtención a partir de los puntos registrados en dichos arrays del tiempo transcurrido hasta dicho límite (tanto en el inferior como en el superior).
4. Resta del segundo menos el primer array de manera que se obtiene el tiempo entre ambos límites.
5. Obtención del resultado en escala logarítmica.

4.2. Etiquetado de los datos

De cara al tratamiento de los datos, se ha añadido al archivo .csv en el que se recogen los valores de las características acústicas organizadas y la etiqueta a la que pertenece cada una de las capturas. De esta manera, posteriormente se puede verificar el correcto funcionamiento del modelo implementado. Este proceso, llevado a cabo sin intervención directa por mi parte, se basó en la ubicación conocida de las larvas y nudos, determinando su posición 2D (azimut y elevación) para agrupar las señales de manera precisa. Las posiciones de interés fueron registradas en el archivo de datos, de manera que la posterior asociación entre las características acústicas medidas y su etiqueta correspondiente fue

más sencilla. Este etiquetado se realizó de manera semiautomática, permitiendo obtener las etiquetas de cada muestra sin una verificación constante, aunque no se aplicaron técnicas adicionales para validar su precisión [8].

Los resultados obtenidos y, por tanto, las etiquetas asignadas a cada detección, se visualizan en la Tabla 4.1.

Blanco	**ID**	**Posición (x, y)**
Ruido	0	
Larva	1	(525, 525)
Larva	2	(0, 175)
Larva	3	(-200, -175)
Larva	4	(200, -175)
Larva	5	(-200, -488)
Larva	6	(200, -562)
Nudo	1	(0, -563)
Nudo	2	(525, 175)

Tabla 4.1: Posiciones de interés dadas en milímetros.

La etiqueta asignada como '0' y designada como ruidoïmplica que la detección no se corresponde ni con una larva ni con un nudo.

Las posiciones anteriormente descritas se pueden ver reflejadas sobre la propia estructura mostrada en la Fig. 4.17, en la que cada listón se identifica con la letra 'B', cada punto en el que se ha insertado la larva aparece representado con un rombo naranja indicando su ubicación y cada nudo representado por un círculo rosa en los dos primeros listones. El designado como nudo 1 es el ubicado en el listón B1, mientras que es nudo 2 es el ubicado en el listón 2.

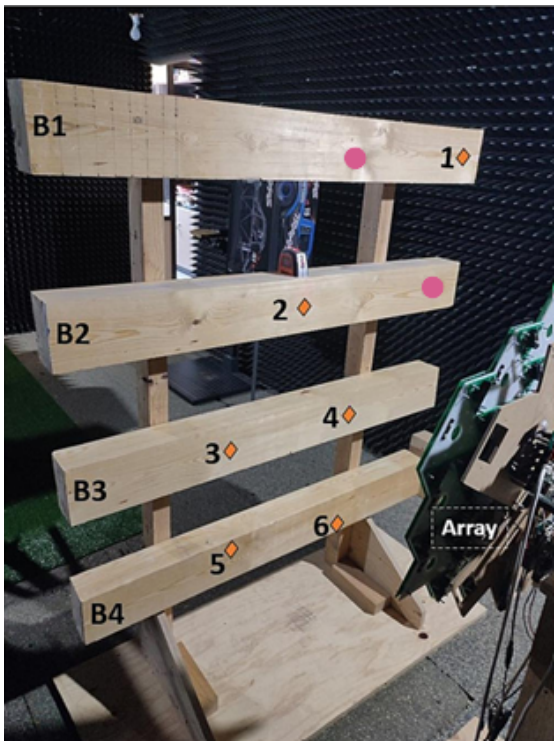
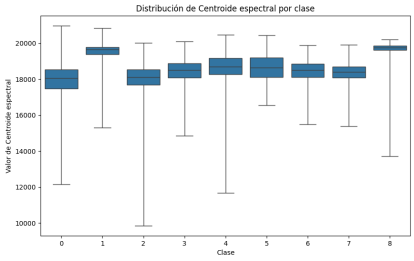


Figura 4.17: Estructura con ubicaciones de larvas y nudos.

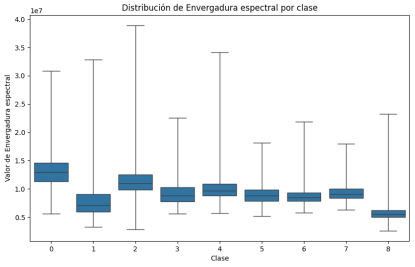
4.3. Filtrado de las características acústicas

De cara a aligerar la carga computacional que requerirá el entreno de los algoritmos de clasificación, se han representado mediante boxplots, cada una de las características acústicas extraídas anteriormente en cada una de las distintas clases o etiquetas.

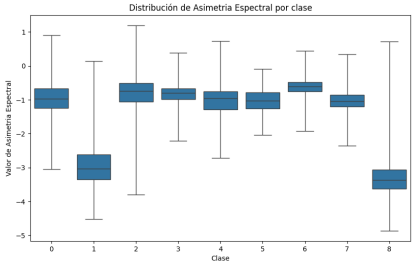
Los resultados obtenidos se visualizan en la figura Fig. 4.18.



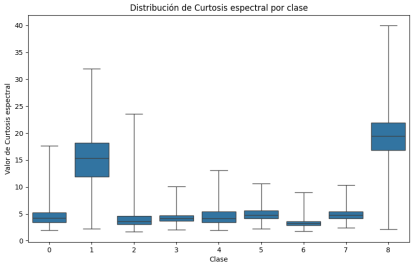
(a) Centroide espectral



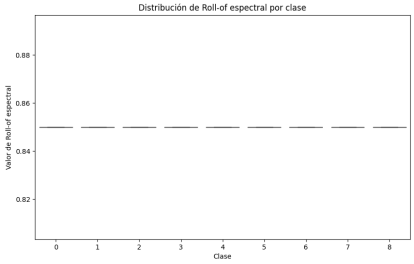
(b) Envergadura espectral



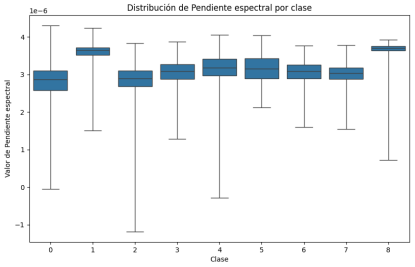
(c) Asimetría espectral



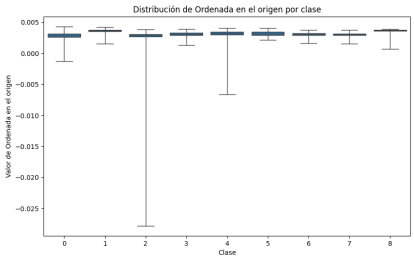
(d) Curtosis espectral



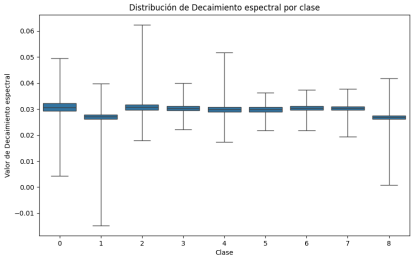
(e) Roll-off espectral



(f) Pendiente espectral

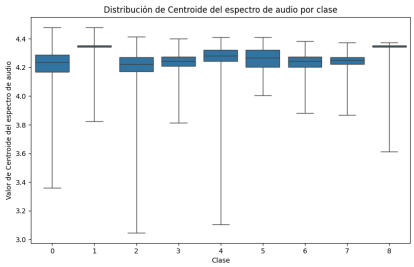


(g) Ordenada en el origen

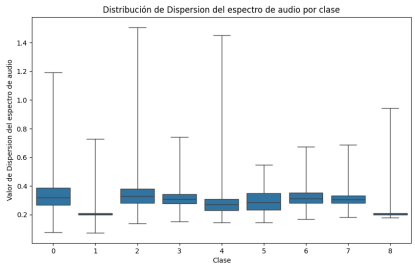


(h) Decaimiento espectral

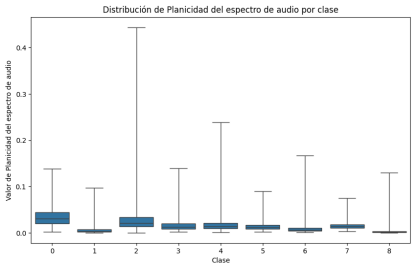
Figura 4.18: Boxplots de las características acústicas (1/2).



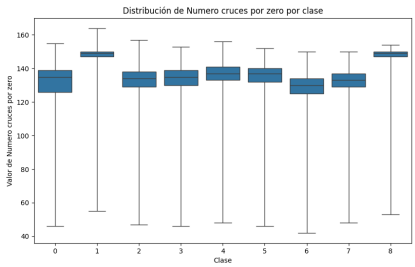
(i) Centroide del espectro de audio



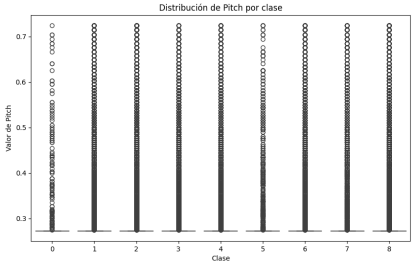
(j) Dispersión del espectro de audio



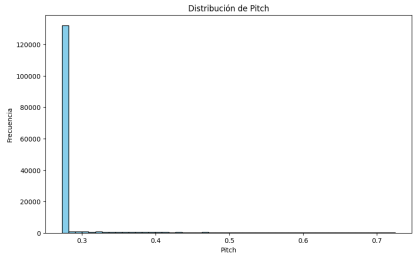
(k) Planicidad del espectro de audio



(l) Número de cruces por cero



(m) Pitch



(n) Gráfico de barras del Pitch

Figura 4.18: Boxplots de las características acústicas (2/2).

En la Figura 4.18e se observa claramente que todos los valores en la distribución de roll-of espectral por clase se mantienen constantes en todas las detecciones. Visualizar esta distribución nos permite eliminar dicha característica del conjunto de datos, puesto que no van a proporcionar información relevante a la hora de la clasificación. En cuanto al resto de características, a excepción de la representada en la Figura 4.18m, aportarán información muy útil para la clasificación, ya que los valores presentan una variabilidad significativa entre clases, lo que impide establecer un patrón común o generalizado aplicable a todas ellas y por tanto su eliminación.

Por otra parte, en el caso de la Figura 4.18m en concreto, se observa que los boxplots son muy similares, pero los outliers o valores atípicos, sobre todo en la clase 0, sí que varían, por lo que se ha decidido mantener dicha característica

en la clasificación. La razón por la cual la representación de la Figura 4.18m es peculiar es debido al concepto del valor intercuartílico (IQR) que utilizan las representaciones box-plot. Para este caso en concreto, la gran mayoría de valores del pitch son iguales, como se puede observar en el gráfico de barras, lo cual hace que el valor del primer cuartil Q1 y del tercer cuartil Q3 sea el mismo, generando una caja en el boxplot inexistente. Esto hace que el valor intercuartílico, medido como IQR $= Q_3 - Q_1$, sea 0. A la hora de generar los bigotes de un boxplot, que engloban los valores de la distribución considerados normales, se extienden hasta los valores mínimo y máximo de la distribución, dentro de un rango razonable que suele ser 1,5 veces el IQR. En este caso, no hay valores que se consideren normales, y todos los que no sean el valor más representativo se han clasificado como outliers o valores atípicos, como se refleja en la Figura 4.18m.

4.4. Implementación de algoritmos de machine learning y resultados

De cara al análisis de los datos, se ha tratado de implementar ciertos algoritmos que, mediante entrenamiento, sean capaces de clasificar cada detección con la etiqueta que le corresponde, sea esta nudo, larva o ninguna de las anteriores, pero que esto se ejecute con la mayor precisión posible.

Se han preparado los datos tanto mediante normalización como mediante estandarización, de cara a comparar cuál de las dos técnicas de escalado de datos proporciona mejores resultados.

Como primer paso después del escalado y separación de los datos, se han implementado en Python los algoritmos de clasificación 'Random Forest', 'K-Nearest neighbors' y 'Support vector machine' de cara a comprobar cuál de ellos proporciona un parámetro AUC o área bajo la curva ROC mayor y clasifica mejor el conjunto de test. Además de este parámetro, se han extraído cuatro características principales. Antes de describirlas, es necesario explicar ciertos conceptos que tienen relación con ellas.

- **Verdaderos positivos o True positive** (**TP**): Representa los casos en los que el modelo clasifica correctamente una clase específica.
- **Verdaderos negativos o True negative** (**TN**): Representa los casos en los que el modelo, para una clase en específico, se ha clasificado como otra clase de manera correcta.
- **Falsos positivos o False positive** (**FP**): Representan los casos en los que el modelo predice incorrectamente una clase específica como si perteneciera a esa clase, cuando en realidad su etiqueta corresponde a otra clase.
- **Falsos negativos o False negative** (**FN**): Representan las veces en las que el modelo detecta incorrectamente la clase de la detección, asignando otra clase en su lugar.

Una vez definidos estos conceptos, se definen ciertas características, extraídas durante el análisis de cada modelo:

- **Accuracy o exactitud**: Representa el porcentaje de valores clasificados correctamente, ya sean positivos o negativos. Se puede calcular de la siguiente forma:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.21}$$

- **Precisión**: Representa el porcentaje de valores positivos, es decir, clasificados correctamente como pertenecientes a una clase concreta. Se puede calcular de la siguiente forma:

$$\text{Precisión} = \frac{TP}{TP + TN} \tag{4.22}$$

- **Recall o sensibilidad**: Representa el porcentaje de valores positivos que han sido correctamente clasificados como tal. Se puede calcular de la siguiente forma:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4.23}$$

- **F1 score**: Representa un balance entre las medidas de precisión y recall, siendo muy útil cuando se trabaja con datos no balanceados. Se puede calcular de la siguiente forma:

$$F_1 = 2 \cdot \frac{\text{Recall} \cdot \text{Precisión}}{\text{Recall} + \text{Precisión}} \tag{4.24}$$

- **Support o soporte**: El soporte es una métrica que indica cuántas instancias reales hay de cada clase en el conjunto de datos, es decir, nos muestra cuántos ejemplos tenemos de cada categoría que el modelo está tratando de clasificar. Esta métrica ayuda a entender si el modelo tiene suficiente información para aprender a clasificar cada clase de manera adecuada. Si el soporte es bajo en alguna clase, eso puede significar que el modelo no ha tenido suficientes ejemplos para aprender bien, lo que podría afectar su rendimiento al clasificar nuevos datos.[9]

Es importante tener en cuenta de cara a la clasificación en el contexto de nuestro análisis, el parámetro 'support'. Es fundamental, ya que indica la cantidad de instancias de cada clase en el conjunto de datos.

Este desbalance entre las diferentes clases estudiadas puede influir significativamente en el desempeño del modelo. Las clases con mayor soporte tenderán a tener mejores métricas de precisión y recall, lo que es esperable, puesto que el modelo tiene más ejemplos para aprender de ellas. Sin embargo, las clases con bajo soporte podrán no ser clasificadas correctamente, o no con la misma precisión que las anteriores, lo que las proporcionará puntuaciones de precisión

muy bajas o incluso nulas. Este fenómeno debido al parámetro recall destaca la importancia de tener un conjunto de datos equilibrado, ya que un soporte bajo puede llevar a que el modelo no generalice bien y tenga dificultades para identificar correctamente esas clases. Por tanto a la hora de trabajar con un conjunto de clases desbalanceado, habrá de ser tenido en cuenta.

En relación a estas ideas, se describe y se aplicará posteriormente el concepto de Curva ROC (Receiver Operating Characteristic). Se trata de una herramienta gráfica utilizada para evaluar el rendimiento de un algoritmo de clasificación, y representa la relación entre la tasa de verdaderos positivos (Sensibilidad, recall o TP) en el eje Y y la tasa de falsos positivos (1-especificidad o FP) en el eje X, tal y como se observa en la Fig. 4.19. Esta curva se construye ajustando el umbral de decisión del modelo, lo que permite generar diferentes combinaciones de TP y FP. Esto facilita la evaluación de la capacidad del modelo para discriminar entre clases a medida que se modifica dicho umbral.

Idealmente se alcanzaría el punto (0, 1) en la curva ROC, lo cual implicaría que el modelo es capaz de identificar todos los casos positivos sin clasificar ninguún falso positivo, dando como resultado una sensibilidad y una especificidad del 100 %. La calidad del modelo puede medirse por el área bajo la curva ROC (AUC). Esto es posible porque un valor AUC cercano a 1 indica un muy buen rendimiento, mientras que un valor cercano a 0.5 indica un desempeño considerado del 50 %, es decir, aleatorio. Un valor menor de 0.5 indica un desempeño peor que el propio azar. La AUC da una visión general de la capacidad del modelo para maximizar los verdaderos positivos y minimizar los falsos positivos, lo cual es especialmente útil cuando estamos trabajando con datos desbalanceados, como es nuestro caso.

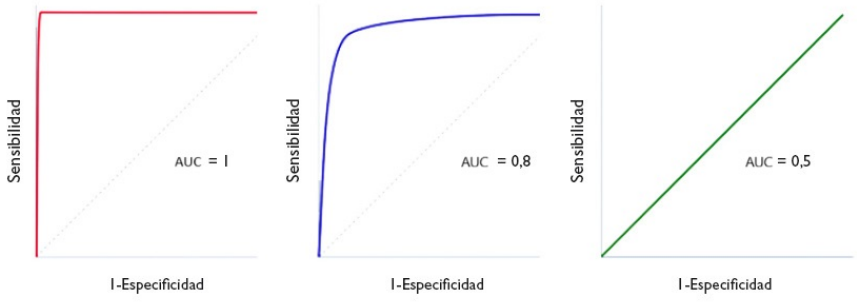


Figura 4.19: Ejemplo de curvas ROC con distintos valores de AUC.

4.4.1. Normalización de los datos sin implementación de validación cruzada

Una vez que las capturas obtenidas del sonido emitido por los insectos xilófagos han pasado por el script de LabVIEW que permite el cálculo de las características espectrales, se genera un archivo .csv también desde el mismo

script, en el que se organizan en columnas cada una de las características de cada captura.

Para poder trabajar con los datos en los algoritmos de 'machine learning' con los que se quiere experimentar, es necesario normalizar los datos, de manera que todos oscilen entre 0 y 1, siendo 0 el valor más bajo de cada característica y 1 el valor mayor. Para ello se ha empleado un script de Python en el que se normalizan los datos mediante la función 'MinMaxScaler' y se genera otro archivo .csv con ellos. Es importante tener en cuenta que la normalización de los datos de entrenamiento y los de test han de ser normalizados independientemente, puesto que de lo contrario se ocasionaría una filtración de datos o 'data leakage' [10], dando lugar al uso de datos del conjunto de prueba en el conjunto de test de manera indirecta. Esto puede derivar en 'overfitting', de manera que el algoritmo se ajuste muy bien a los datos proporcionados, pero no sea capaz de generalizar para nuevos conjuntos de datos no vistos a lo largo del entrenamiento, y por tanto, proporcione indicadores erróneos sobre el rendimiento del propio algoritmo.

4.4.1.1. Random Forest

Se han variado para un mismo modelo el número de árboles de decisión para comprobar su influencia en la capacidad de clasificación del modelo. Se han obtenido los resultados presentados en la Tabla 4.2.

Nº de árboles de decisión	**Exactitud**	**AUC**
10	0.5052	0.83
50	0.5526	0.90
100	0.5519	0.90
200	0.5577	0.91
500	0.5586	0.91
1000	0.559	0.91

Tabla 4.2: Resultados del modelo Random Forest

Se puede observar que la exactitud aumenta al incrementar el número de árboles de decisión, pero no de manera significativa, a excepción de con 10 árboles de decisión y con 1000 árboles, cifra en la cual empieza a descender. En cuanto a la AUC, a partir de 100 árboles de decisión y hasta los 1000 se mantiene estable. Los mejores resultados se obtienen con 500 árboles.

Para el número de árboles de decisión con el que mejores resultados se han obtenido con este algoritmo, 500, se obtienen las curvas ROC de cada una de las clases y la curva ROC promedio, reflejadas en la Fig. 4.20.

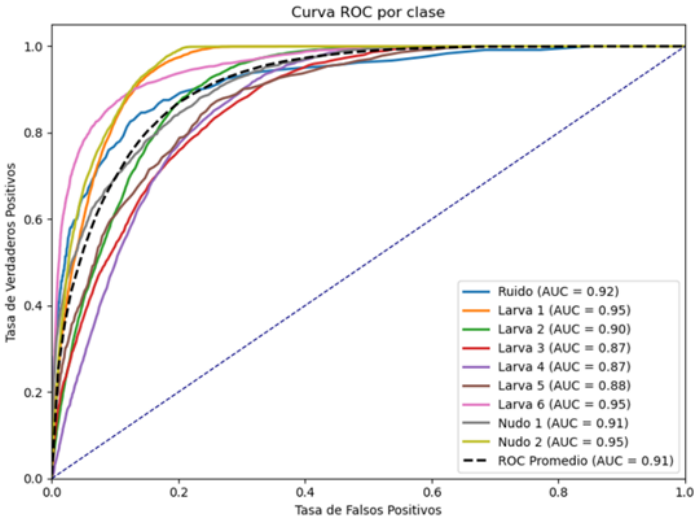


Figura 4.20: Curvas ROC por clase y promediada en Random Forest para 500 árboles.

Y las métricas obtenidas son las indicadas en la Tabla. 4.3.

Clase	**Precisión**	**Recall**	**F1-Score**	**Soporte**
Ruido	0.58	0.17	0.27	241
Larva 1	0.81	0.15	0.25	4988
Larva 2	0.55	0.60	0.57	5137
Larva 3	0.56	0.28	0.37	3554
Larva 4	0.37	0.50	0.43	3161
Larva 5	0.50	0.12	0.19	505
Larva 6	0.76	0.71	0.74	4044
Nudo 1	0.37	0.60	0.46	1714
Nudo 2	0.59	0.98	0.74	6256

Tabla 4.3: Métricas de Exactitud, Recall, F1-Score y Soporte por clase

De estos resultados se obtienen las siguientes conclusiones:

- La larva 6 y la larva 8 presentan un buen equilibrio entre precisión y recall, ambos parámetros siendo altos. La clasificación de estas clases es correcta en la mayoría de los casos, lo que indica que el modelo las identifica y clasifica adecuadamente.
- Las larvas 2 y 4 y el nudo 1 muestran un rendimiento intermedio, con valores de precisión y recall moderados. A pesar de que el modelo es aceptable para dichas clases, no destacan en ninguna de las métricas obtenidas en

particular, lo que da pie a que exista un margen de mejora en la clasificación de estas clases.

- El ruido y las larvas 1, 3 y 5 presentan valores bajos de precisión y recall, lo que indica dificultades significativas en la clasificación de dichas clases. Esto implica que el modelo no las está reconociendo adecuadamente, ya sea por falta de características distintivas o por insuficiencia de support en dichas clases.

En cuanto a las curvas de aprendizaje obtenidas para el caso de óptimo funcionamiento del modelo, 500 árboles, se obtienen las reflejadas en la Fig. 4.21.

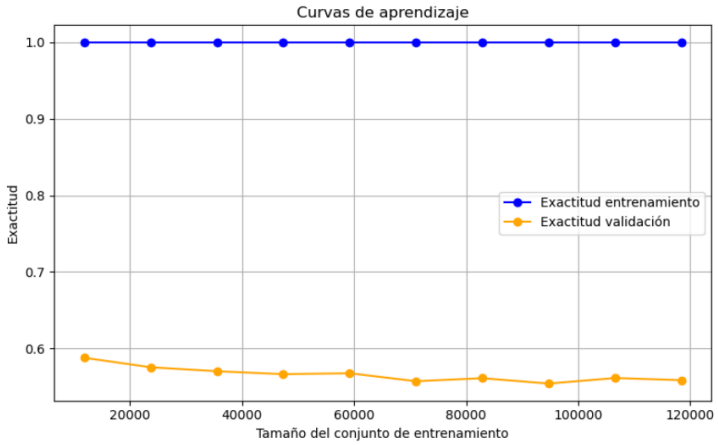


Figura 4.21: Curvas de aprendizaje para Random Forest implementando 500 árboles.

En la Fig. 4.21 se tienen enfrentados en el eje X el tamaño del conjunto de entrenamiento frente a la exactitud obtenida en el eje Y, es decir, lo bien que predice el modelo.

A lo largo del entrenamiento se observa que la exactitud se mantiene constante en el 100 %, indicando que el modelo aprende a la perfección los datos con los que se está entrenando. Sin embargo, a la hora de evaluar la evolución de la exactitud en la validación, esta parte de un valor relativamente bajo comparándolo con la curva de entrenamiento (en torno al 59 %) y se estabiliza cuando baja hasta el 55 %.

Esto indica que el modelo no está aprendiendo a predecir nuevos datos. Durante el entrenamiento se está memorizando los datos, en vez de aprendiendo a generalizar. Se produce overfitting, observable por la gran brecha existente entre ambas curvas, solucionable mediante validación cruzada o mediante el empleo de un menor número de árboles entre otras opciones.

4.4.1.2. K-Nearest Neighbors

Se ha variado para el mismo modelo el número de vecinos a tener en cuenta a la hora de que el algoritmo clasifique cada captura obtenida. Se han obtenido los resultados presentados en la Tabla. 4.4:

Nº de Vecinos	**Exactitud**	**AUC**
2	0.4359	0.69
5	0.5015	0.78
10	0.5507	0.83
30	0.5765	0.87
60	0.5819	0.89
100	0.5835	0.90
200	0.5856	0.91
500	0.5847	0.92
1000	0.5759	0.92

Tabla 4.4: Resultados del modelo KNN

Se observa que a medida que se incrementa el número de vecinos el aumento de la exactitud y de la AUC es muy significativo, sobre todo al principio. Sin embargo, al determinar un número de 200 vecinos se alcanza el mayor valor de exactitud, incluso más alto que cuando se colocan números de vecinos mayores. Además, con este incremento la AUC no sufre una mejora reseñable.

Esto puede deberse al 'over-smoothing'. Esto explica la situación en la que se considera un número excesivo de vecinos en el modelo KNN, lo que hace que la variación local de las clases no se tenga en cuenta adecuadamente. Como resultado, se experimenta una menor exactitud, ya que el modelo promedia las clases sobre un número elevado de puntos que pueden pertenecer a diferentes categorías. A su vez, se suavizan los límites de decisión, por lo que se reduce la capacidad del modelo para diferenciar correctamente entre las distintas clases. [11]

Para el número de vecinos con el que mejor resultados se han obtenido del algoritmo aplicado, 200, se grafican las curvas ROC de cada una de las clases y la curva promedio, reflejadas en la Fig. 4.22.

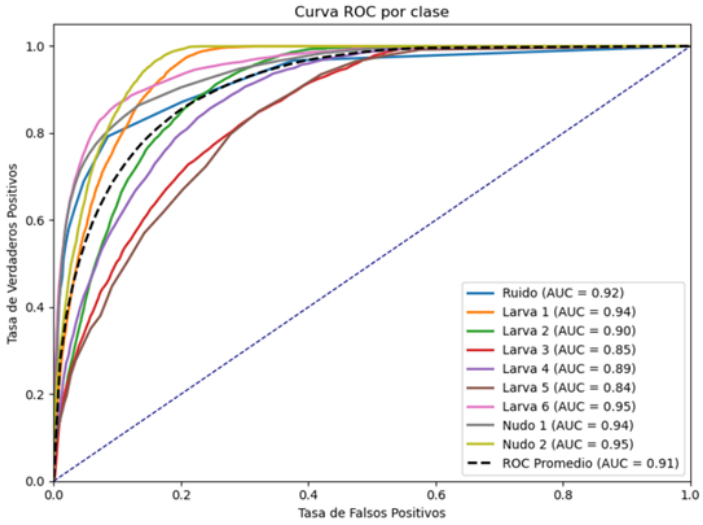


Figura 4.22: Curvas ROC por clase y promediada en KNN para 200 vecinos.

Y las métricas obtenidas son las indicadas en la Tab. 4.5.

Clase	**Precisión**	**Recall**	**F1-Score**	**Soporte**
Ruido	0.0	0.0	0.0	241
Larva 1	0.68	0.66	0.67	4988
Larva 2	0.45	0.87	0.60	5137
Larva 3	0.42	0.48	0.45	3554
Larva 4	0.52	0.37	0.43	3161
Larva 5	0.75	0.01	0.02	505
Larva 6	0.89	0.41	0.56	4044
Nudo 1	0.81	0.20	0.32	1714
Nudo 2	0.75	0.75	0.75	6256

Tabla 4.5: Métricas de Precisión, Recall, F1-Score y Soporte por clase

De estos resultados se obtienen las siguientes conclusiones:

- La clase que representa la larva 8 presenta el mejor equilibrio entre precisión y recall, con valores altos en ambas métricas, lo que indica que el modelo la clasifica correctamente en la mayoría de los casos.
- La clase que representa la larva 6 muestra una precisión elevada, pero su recall es relativamente bajo, lo que indica que el modelo deja sin detectar muchas de las instancias presentes, a pesar de que las que si que reconoce sean bien clasificadas. Esto implica que no se detectan suficientes instancias de dicha clase. De igual manera ocurre con la clase correspondiente al nudo 1.

- Las larvas 1, 2, 3 y 4 tienen valores de precisión y de recall moderados, pero no destacan en ellos.
- La clase que representa al ruido y la larva 5 presentan un recall extremadamente bajo. Son clases no reconocidas por el algoritmo, es decir, no las identifica, ya sea por falta de support o porque no encuentre parámetros distintivos.

Al igual que sucede en el caso del modelo anterior, Random Forest, K-Nearest Neighbors no es un modelo que se entrene por épocas, por lo que se sigue la misma estrategia empleada anteriormente.

La gráfica obtenida es la reflejada en la Fig. 4.23.

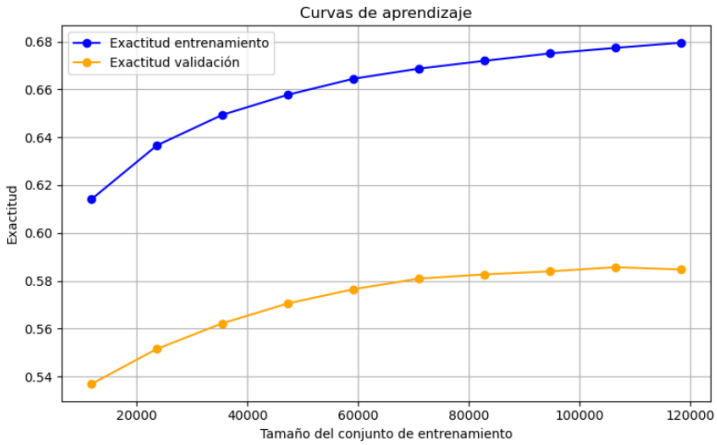


Figura 4.23: Curvas de aprendizaje para KNN implementando 200 vecinos.

La curva de aprendizaje muestra el comportamiento esperado del modelo. En cuanto a la curva correspondiente a la exactitud del entrenamiento , mejora a medida que se incrementa el número de datos, es decir, no se estanca y cada vez aprende más patrones útiles.

A pesar de que se trata de un caso de overfitting, dado que existe una gran brecha entre ambas curvas, el modelo es capaz de aprender patrones, puesto que la curva de entrenamiento va incrementando a medida que se aumenta el tamaño del conjunto de datos. El modelo generaliza, aunque no lo haga de manera muy eficiente.

Se observa también que, en el caso de la curva de validación, en las iteraciones finales la curva se estanca e incluso desciende levemente, dando a entender que con este modelo no se alcanzarían exactitudes mayores a pesar de incrementar el tamaño del conjunto de datos empleado.

4.4.1.3. Support Vector Machine

En este caso, para un mismo algoritmo SVM y manteniendo el kernel lineal, ya que es el más adecuado para el tipo de datos con el que se está trabajando, se ha variado el parámetro C para obtener resultados que indiquen en qué punto el clasificador alcanza una mayor exactitud.
El parámetro C en los algoritmos SVM actúa como un balanceador entre la exactitud del modelo y la complejidad de la función de decisión, de manera que cuanto mayor es C, más se adapta el modelo a los pequeños detalles y ruidos de los datos. Por tanto, ajustando el parámetro C podemos balancear la simplicidad de la clasificación y su correcta ejecución. Con valores altos de C obtendremos una exactitud mayor al clasificar los datos, mientras que con valores bajos la frontera de decisión adoptará una función más simple. [12] Se han obtenido los resultados presentados en la Tabla. 4.6.

Valor de C	**Exactitud**	**AUC**
0.001	0.390	0.77
0.01	0.5414	0.89
0.1	0.5493	0.89
1	0.5251	0.88
10	0.4354	0.86
100	0.4205	0.80

Tabla 4.6: Resultados del modelo SVM

Se observa que el parámetro de exactitud alcanza su valor máximo cuando $C = 0.01$. En dicho punto, además, el parámetro AUC ha terminado de crecer. A partir de 0.1, tanto los valores de la exactitud como los de AUC van disminuyendo. Considerando el valor de C que optimiza mejor ambos parámetros y que, al mismo tiempo, resulta computacionalmente viable, se decide establecer $C = 0.1$.

Para el valor de C con el que mejores resultados se han obtenido con este algoritmo, 0.1, se obtienen las curvas ROC de cada una de las clases y la curva ROC promediada, reflejadas en la Fig. 4.24

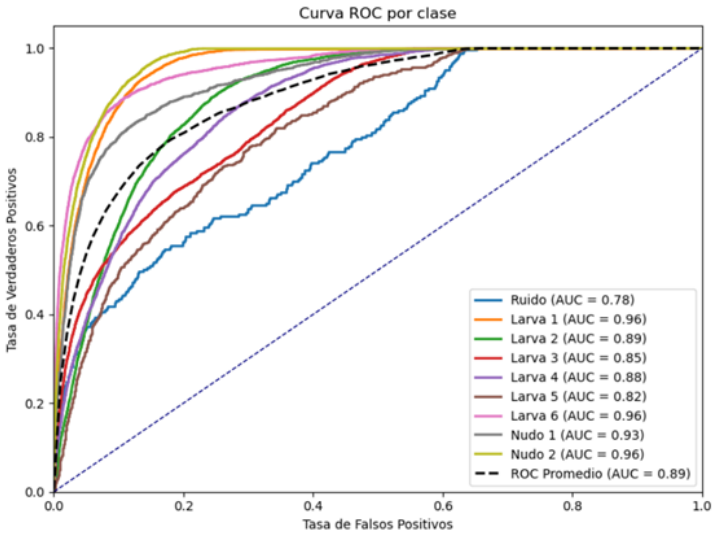


Figura 4.24: Curvas ROC por clase y promediada en Support Vector Machine para $C = 0{,}1$.

Y las métricas obtenidas son las indicadas en la Tabla. 4.7.

Clase	**Precisión**	**Recall**	**F1-Score**	**Soporte**
Ruido	0.00	0.00	0.00	241
Larva 1	0.87	0.29	0.43	4988
Larva 2	0.43	0.90	0.58	5137
Larva 3	0.42	0.44	0.46	3554
Larva 4	0.49	0.35	0.40	3161
Larva 5	0.00	0.00	0.00	505
Larva 6	0.93	0.38	0.53	4044
Nudo 1	0.79	0.01	0.02	1714
Nudo 2	0.65	0.96	0.77	6256

Tabla 4.7: Métricas de Exactitud, Recall, F1-Score y Soporte por clase

De estos resultados se obtienen las siguientes conclusiones:

- El modelo no identifica ningún caso de las larvas 0 y 5. Puede deberse a muy pocos datos en el entrenamiento o a que las características de estas clases están muy solapadas con otras.
- En cuanto a la larva 2 y el nudo 2, tienen el recall alto pero la precisión relativamente baja. El modelo clasifica en dichas clases más datos de los que pertenecen a ellas realmente.

- Lo contrario ocurre para las larvas 1 y 6, el algoritmo deja fuera de ellas muchos de los datos. Tiene alta precisión pero recall bajo.
- Las larvas 3 y 4 tienen unos valores de precisión y un recall equilibrados. A pesar de no ser valores muy elevados, el modelo las predice con suficiente fiabilidad.
- En cuanto al nudo 1, el modelo prácticamente no tiene en cuenta dicha clase.

Es posible que el problema se deba a un desequilibrio en los datos. Podrían emplearse técnicas de balanceo de datos para tratar de que no haya tantas diferencias entre clases.

Al igual que sucede en el caso del modelo anterior, Random Forest, SVM no es un modelo que se entrene por épocas, por lo que se sigue la misma estrategia empleada anteriormente. Las curvas de aprendizaje son las reflejadas en la Fig. 4.25.

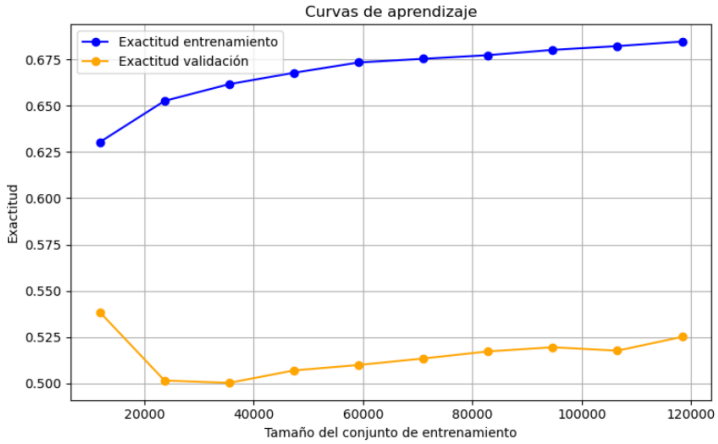


Figura 4.25: Curvas de aprendizaje para SVM con $C = 0{,}1$.

En cuanto a la curva correspondiente a la exactitud del entrenamiento , mejora a medida que se incrementa el número de datos dedicados a ello, es decir, va aprendiendo más patrones útiles. El modelo no únicamente aprende los datos con los que entrena, sino que a medida que se incrementa el número de datos, va aprendiendo a generalizar.

En cuanto a la curva de validación, empieza en valores cercanos al 54 %, tras lo cual tiene un descenso muy pronunciado. Posteriormente va recuperando y mejorando el valor de la exactitud y que con un tamaño mayor del conjunto de datos de entrenamiento se podría conseguir incrementarlo más.

En cuanto al overfitting, se sigue produciendo en este modelo también. Esto se refleja en la brecha entre ambas curvas.

4.4.1.4. Conclusiones para datos normalizados sin CV

De los modelos aplicados sin validación cruzada, Random Forest muestra un rendimiento con una AUC de 0,91 y una exactitud del 55,86 %. Su comportamiento mejora al aumentar el número de árboles, estabilizándose a partir de 100. K-Nearest Neighbors también alcanza una AUC de 0,91, con una exactitud ligeramente superior, del 58,56 %, al usar 200 vecinos. No obstante, su rendimiento decrece si se incrementa este parámetro en exceso. SVM, por su parte, logra una AUC de 0.89 y una exactitud del 55,14 % con $C = 0{,}1$, mostrando un rendimiento competitivo y una buena capacidad de generalización sin requerir ajustes complejos.

En este escenario sin validación cruzada, los tres modelos se comportan de forma estable, pero hay que reseñar que, Random Forest, a pesar de presentar mayor robustez ante el desequilibrio de clases y ser menos sensible a la elección de parámetros, es el modelo que mayor brecha presenta en la curva de aprendizaje y el modelo más sobreajustado de los tres. Además, presenta la única curva de validación que disminuye su exactitud con el incremento del tamaño del conjunto de datos.

En este conjunto de resultados sin validación cruzada, se mantiene el desbalance en la distribución de clases, lo cual afecta al resto de métricas obtenidas y comparadas. Las clases correspondientes a Larva 6 y Nudo 2 dominan en cuanto a rendimiento, mientras que clases como Larva 3, Larva 5 o Ruido resultan más difíciles de clasificar, lo cual limita la precisión global del modelo.

4.4.2. Estandarización de los datos sin implementación de validación cruzada

Para trabajar en este caso con los datos en los algoritmos de machine learning, se ha decidido estandarizar las características espectrales, ajustándolas para que tengan una media de 0 y una desviación estándar de 1. Este proceso se ha llevado a cabo mediante un script en Python utilizando la función 'StandardScaler', lo que puede ser crucial para algoritmos sensibles a la escala de las características. Al igual que con la normalización, es fundamental realizar la estandarización de los datos de entrenamiento y de prueba de manera independiente para evitar problemas de data leakage que podrían afectar la generalización del modelo.

4.4.2.1. Random Forest

Se han variado para un mismo modelo el número de árboles de decisión para comprobar su influencia en la capacidad de clasificación del modelo. Se han obtenido los resultados presentados en la Tabla. 4.8.

Nº de árboles de decisión	Exactitud	AUC
10	0.7366	0.92
50	0.7577	0.95
100	0.76	0.96
200	0.7618	0.96
500	0.7623	0.96

Tabla 4.8: Resultados del modelo Random Forest sin CV.

Se puede observar que la exactitud aumenta al incrementar el número de árboles de decisión, lo hace de manera más significativa entre las primeras ejecuciones que entre las últimas. De igual manera ocurre con la AUC, puesto que a partir de 100 árboles de decisión y hasta los 500 se mantiene estable. Por la ligera mejora en la exactitud y por la carga computacional que supone para muy poca mejora en la AUC, con un número de árboles de decisión igual a 200 se obtienen muy buenos resultados con una carga computacional moderada.

Para el número de árboles de decisión con el que mejores resultados se han obtenido con este algoritmo, 200, se obtienen las curvas ROC de cada una de las clases y la curva ROC promediada, reflejadas en la Fig. 4.26.

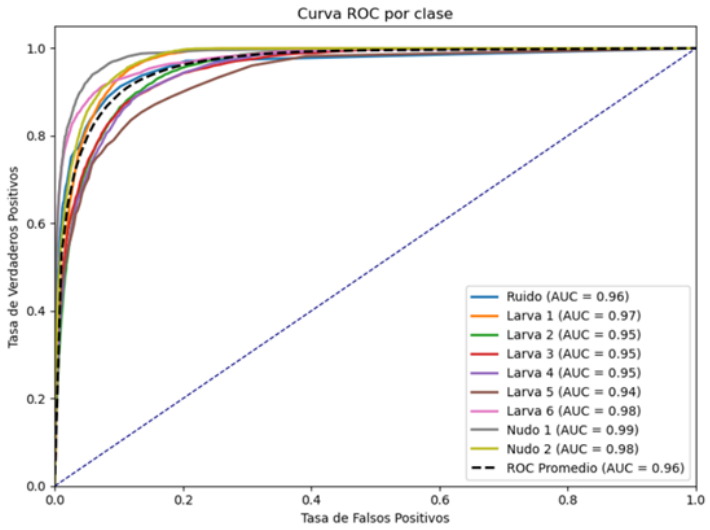


Figura 4.26: Curvas ROC por clase y promediada en Random Forest para 200 árboles.

Y las métricas obtenidas son las indicadas en la Tabla. 4.9.

Clase	Precisión	Recall	F1-Score	Soporte
Ruido	0.69	0.33	0.44	241
Larva 1	0.79	0.79	0.79	4988
Larva 2	0.68	0.80	0.74	5137
Larva 3	0.69	0.70	0.70	3554
Larva 4	0.68	0.64	0.66	3161
Larva 5	0.74	0.25	0.37	505
Larva 6	0.86	0.80	0.83	4044
Nudo 1	0.78	0.77	0.77	1714
Nudo 2	0.83	0.83	0.83	6256

Tabla 4.9: Métricas de Exactitud, Recall, F1-Score y Soporte por clase

De estos resultados se obtienen diversas conclusiones. Las clases correspondientes a las larvas 1, 2 y 6 y a los nudos 1 y 2 presentan los mejores resultados en cuanto a precisión, recall y F1-score, especialmente la larva 6 y el nudo 2, que muestran un equilibrio óptimo. Para entender este comportamiento, observamos que:

- El buen rendimiento de las detecciones papra las larvas 1, 2 y 6 y los nudos 1 y 2 puede deberse a que las detecciones están suficientemente separadas espacialmente.
- Las larvas 3 y 4 muestran un comportamiento intermedio, posiblemente porque las larvas están próximas entre sí y situadas en la misma viga, dificultando la discriminación.
- La clase correspondiente al ruido presenta un mal desempeño, probablemente por corresponder a ruido, sin un patrón definido.
- La larva 5 también muestra bajo rendimiento, lo cual puede explicarse por la gran dispersión de sus medidas y por la cercanía de la larva a la superficie de la viga, provocando detecciones fuera del patrón y confusión en los algoritmos.

De cara a poder comparar la exactitud del entrenamiento frente a la de la fase de validación no es posible generar un gráfico por épocas, puesto que Random Forest no es un modelo que entrene por épocas como podría serlo otro tipo de redes neuronales. Por esto, en este caso no obtenemos como parámetro la evolución de la exactitud. A pesar de esto, se puede realizar una aproximación para visualizar la 'exactitud de entrenamiento vs. exactitud de validación por época' entrenando el modelo con distintos tamaños del conjunto de entrenamiento y midiendo la exactitud en cada uno de los casos.

En este caso, la gráfica de las curvas de aprendizaje queda reflejada en la Fig. 4.27.

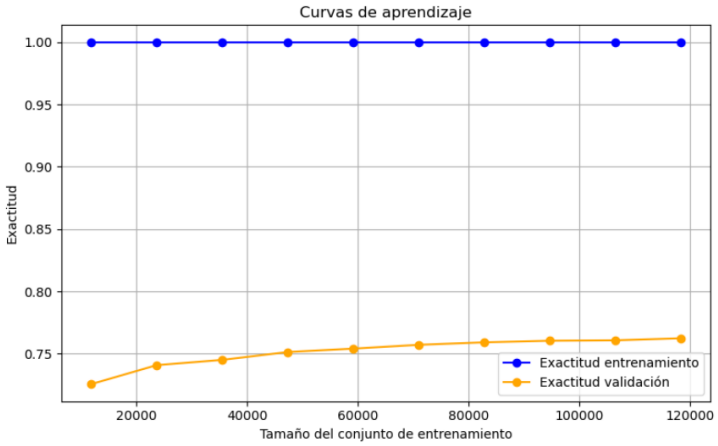


Figura 4.27: Curvas de aprendizaje para Random Forest con 200 árboles.

En la Fig. 4.27 ocurre algo similar a lo visto en el caso del análisis con los datos normalizados. El modelo aprende a la perfección los datos de los que dispone, pero a la hora de la validación, no generaliza bien y no alcanza valores de exactitud interesantes. Además, las curvas presentan una brecha bastante amplia entre ellas. Se trata de un caso de overfitting.

4.4.2.2. K-Nearest Neighbors

Se ha variado para el mismo modelo el número de vecinos a tener en cuenta a la hora de que el algoritmo clasifique cada captura obtenida. Se han obtenido los resultados presentados en la Tabla. 4.10.

Nº de vecinos	**Exactitud**	**AUC**
2	0.6596	0.83
5	0.7129	0.88
10	0.7224	0.91
30	0.7241	0.93
60	0.7156	0.94
100	0.7076	0.95
200	0.6955	0.95

Tabla 4.10: Resultados del modelo K-Nearest Neighbors

Se observa que a medida que se incrementa el número de vecinos, la exactitud va incrementando, hasta llegar a 30, a partir del cual este valor va disminuyendo. En cuanto al parámetro AUC, este va aumentando de manera progresiva, hasta que entre las últimas ejecuciones se ve que el incremento es bastante más lento, es decir, la mejora no es reseñable.

La explicación de la disminución de la exactitud puede ser el 'over-smoothing'. Lo que ocurre es que el modelo promedia las clases sobre un número elevado de puntos que pueden pertenecer a diferentes categorías. Esto suaviza los límites de decisión, haciéndolos menos precisos y reduciendo la capacidad del modelo para diferenciar correctamente entre clases. Haciendo un equilibrio entre resultados y complejidad del algoritmo, el número de vecinos a considerar óptimo en este caso es 30.

Para el número de vecinos con el que mejores resultados se han obtenido con este algoritmo, 30, se obtienen las curvas ROC de cada una de las clases y la curva ROC promediada, reflejadas en la Fig. 4.28.

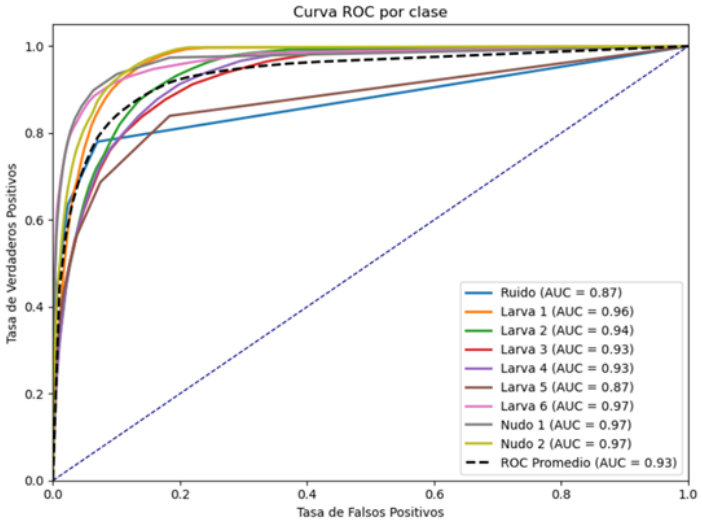


Figura 4.28: Curvas ROC por clase y promediada en K-Nearest Neighbors para 30 vecinos.

Y las métricas obtenidas son las indicadas en la Tabla. 4.11.

Clase	Precisión	Recall	F1-Score	Soporte
Ruido	0.56	0.22	0.32	241
Larva 1	0.74	0.79	0.77	4988
Larva 2	0.62	0.80	0.70	5137
Larva 3	0.66	0.59	0.62	3554
Larva 4	0.63	0.58	0.60	3161
Larva 5	0.73	0.11	0.20	505
Larva 6	0.81	0.80	0.80	4044
Nudo 1	0.79	0.66	0.72	1714
Nudo 2	0.83	0.78	0.81	6256

Tabla 4.11: Métricas de Exactitud, Recall, F1-Score y Soporte por clase

De estos resultados se obtienen las siguientes conclusiones:

- Las clases que presentan un mejor rendimiento son la de las larvas 1, 2 y 6 y las de los nudos 1 y 2. Su equilibrio entre recall y precisión es muy bueno, y destacan la larva 6 y el nudo 2 por tener mejores valores de sus métricas.
- las larvas 3 y 4 presentan un rendimiento moderado. A pesar de que presentan valores buenos, son optimizables.
- Las clases que peor rendimiento presentan son la del ruido y la de la larva 5. Son las dos clases con menor valor de support, y por tanto, puede que esto se deba al desbalanceo de las clases.

Al igual que sucede en el caso del modelo anterior, Random Forest, K-Nearest Neighbors no es un modelo que se entrene por épocas, por lo que se sigue la misma estrategia empleada anteriormente.

La gráfica de las curvas de aprendizaje se refleja en la Fig. 4.29.

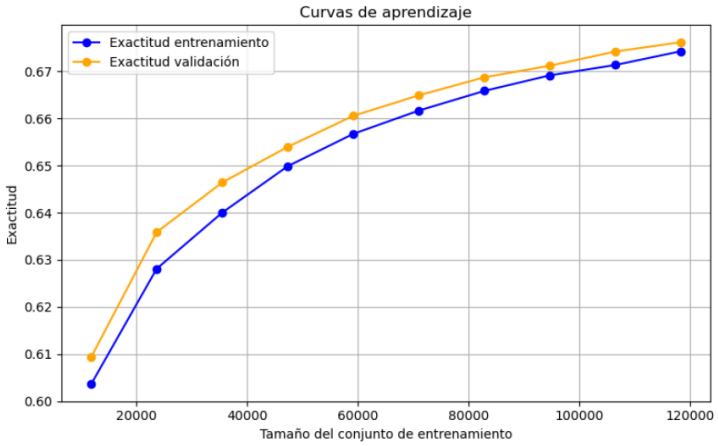


Figura 4.29: Curvas de aprendizaje para KNN con 30 vecinos.

En la Fig. 4.29 se visualiza que la curva de entrenamiento presenta una evolución progresiva, de manera que, a medida que se incrementa el tamaño del conjunto de entrenamiento, el modelo aprende más patrones útiles de los datos.

En cuanto a la curva que representa la exactitud en la validación, muestra una pendiente igual de pronunciada que la de entrenamiento, pero comienza y finaliza en valores algo inferiores. El sobreajuste se ve muy reducido en comparación con los datos obtenidos en las curvas de aprendizaje de los modelos entrenados con los datos normalizados. La brecha entre ambas curvas en este caso es casi inexistente.

Se observa también que, según la tendencia que siguen ambas curvas, existe margen de mejora en caso de incrementar el volumen de datos del que se dispone.

4.4.2.3. Support Vector Machine

En este caso, para un mismo algoritmo SVM y manteniendo el kernel lineal, ya que es el más adecuado para el tipo de datos con el que se está trabajando, se ha variado el parámetro C para obtener resultados que indiquen en qué punto el clasificador alcanza una mayor exactitud.

El parámetro C en los algoritmos SVM actúa como un balanceador entre la exactitud del modelo y la complejidad de la función de decisión, de manera que cuanto mayor es C, más se adapta el modelo a los pequeños detalles y ruidos de los datos. Por tanto, ajustando el parámetro C podemos balancear la simplicidad de la clasificación y su correcta ejecución. Con valores altos de C obtendremos una exactitud mayor al clasificar los datos, mientras que con valores bajos la frontera de decisión adoptará una función más simple. [12] Se han obtenido los resultados presentados es la Tabla. 4.12.

Valor de C	**Exactitud**	**AUC**
0.001	0.5389	0.92
0.01	0.6310	0.93
0.1	0.7014	0.95
1	0.7349	0.95
10	0.7547	0.96
100	0.7690	0.96

Tabla 4.12: Resultados del modelo SVM

Al incrementar el valor del parámetro C, los valores de exactitud aumentan progresivamente, inicialmente de manera más notable que en las últimas ejecuciones. Sin embargo, el valor no llega a estabilizarse ni a descender. En cambio, si observamos el AUC o área bajo la curva ROC, este se mantiene constante a partir de $C = 10$. Aunque la mayor exactitud se obtiene con $C = 100$, la mejora es mínima en comparación con el aumento en los recursos necesarios para

ejecutar el algoritmo. Por ello, el mejor equilibrio entre resultados y eficiencia se encuentra en $C = 10$.

Para el valor de C con el que mejores resultados se han obtenido con este algoritmo, 10, se obtienen las curvas ROC de cada una de las clases y la curva ROC promediada, reflejadas en la Fig. 4.30.

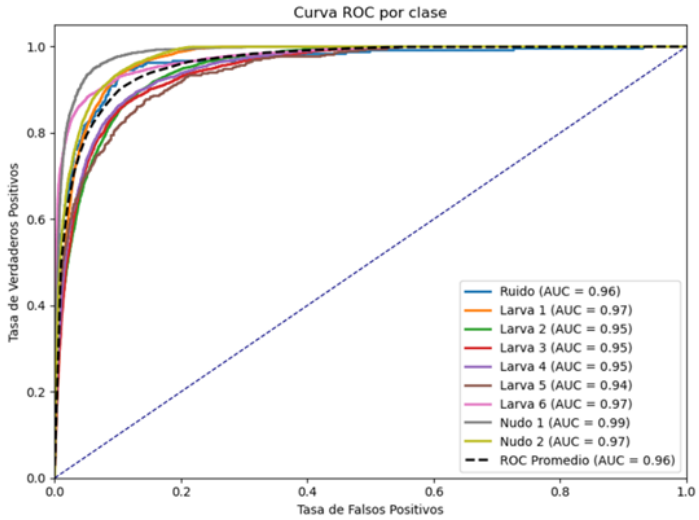


Figura 4.30: Curvas ROC por clase y promediada en SVM con $C = 10$.

Y las métricas obtenidas son las indicadas en la Tabla. 4.13.

Clase	**Precisión**	**Recall**	**F1-Score**	**Soporte**
Ruido	0.64	0.31	0.41	241
Larva 1	0.78	0.80	0.79	4988
Larva 2	0.67	0.79	0.73	5137
Larva 3	0.68	0.66	0.67	3554
Larva 4	0.67	0.65	0.66	3161
Larva 5	0.78	0.24	0.36	505
Larva 6	0.86	0.80	0.83	4044
Nudo 1	0.76	0.78	0.77	1714
Nudo 2	0.84	0.82	0.83	6256

Tabla 4.13: Métricas de Exactitud, Recall, F1-Score y Soporte por clase

De estos resultados se obtienen las siguientes conclusiones:

- En general el rendimiento del calsificador es bueno para la mayoría de clases de larvas.

- Destacan los F1-score de las larvas 1, 2 y 6, por lo que el modelo está detectando dichas clases de manera muy eficaz.
- En las clases de la larva 5 y el ruido se observa que el F1-score es más bajo que en el resto de clases. Esto está relacionado de nuevo con la falta de muestras de dichas clases, lo que limita la capacidad del modelo de generalizar para esas clases y genera más diferencia de resultados en las métricas entre clases.
- Las clases que presentan un número de datos (Soporte) mayor tienen un rendimiento bastante mayor que las que menos número de datos presentan.
- En concreto, la larva 6 y el nudo 2 destacan por su alta precisión, es decir, el modelo acierta la mayor parte de las veces que las predice. Sin embargo, por el bajo recall que presenta la larva 5, deducimos que el modelo detecta en muy pocas ocasiones de manera correcta dicha clase.

Al estar implementando el algoritmo SVM para machine learning, el cual no entrena por épocas como pueden hacerlo otros algoritmos enfocados a deep learning, se realiza la misma aproximación ya implementada anteriormente para visualizar la evolución tanto de la exactitud de la curva de entrenamiento como de la curva de validación.

En este caso, resulta interesante comparar las curvas de aprendizaje obtenidas para el caso analizado y decidido como óptimo para este clasificador, $C = 10$, reflejadas en la Fig. 4.31.

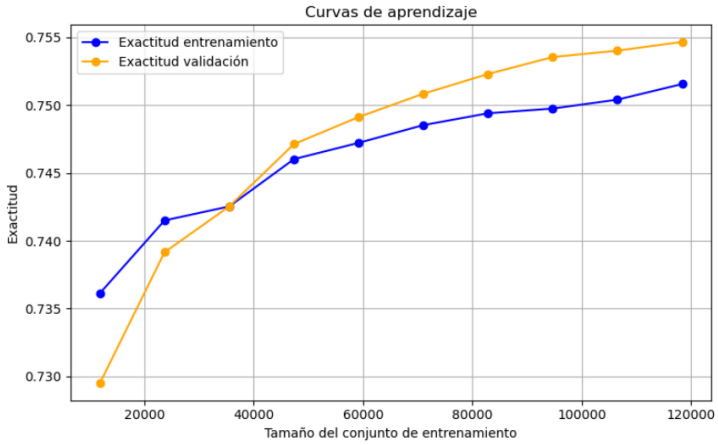


Figura 4.31: Curvas de aprendizaje para SVM con el parámetro $C = 10$.

con las curvas de aprendizaje obtenidas en el caso en el que $C = 100$ y la exactitud del modelo es ligeramente mayor a la obtenida con $C = 10$, reflejadas en la Fig. 4.32.

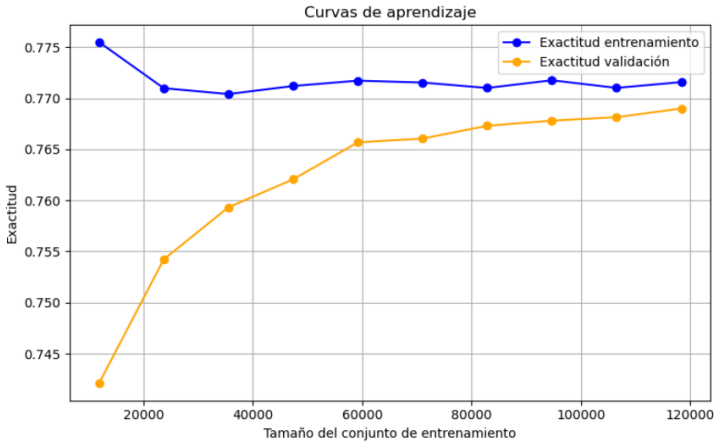


Figura 4.32: Curvas de aprendizaje para SVM con el parámetro $C = 100$.

Se observa que para ambos casos la curva de aprendizaje de la exactitud en la validación sigue una tendencia parecida, incrementando progresivamente a medida que se aumenta el tamaño del conjunto de datos. Además, se observa que en la Fig. 4.32 se alcanza la exactitud que predijo el modelo con anterioridad, siendo esta mayor que en el caso de la Fig. 4.31.

En cuanto a las curvas correspondientes al entrenamiento, sigue una tendencia mejor de aprendizaje la perteneciente a la Fig. 4.31, a pesar de que el las últimas épocas se observa que alcanza una exactitud menor que la alcanzada por la curva de validación. Esto podría deberse a que el modelo esté subajustando.

En el caso de la Fig. 4.32 la curva de entrenamiento sigue una tendencia mucho más lineal, acercándose a la curva de validación en los puntos en los que el tamaño del conjunto de entrenamiento es mayor. A pesar de esto, la brecha que existe en momentos iniciales de la ejecución del algoritmo, indica que puede existir cierto sobreajuste.

4.4.2.4. Conclusiones para datos estandarizados sin CV

De los modelos aplicados sin validación cruzada, Random Forest muestra el mejor rendimiento general, con una AUC de 0,96 y una exactitud del 76,18 %. Este modelo destaca por su robustez frente al desbalance de clases. Su desempeño se estabiliza en torno a 200.

Mediante el algoritmo K-Nearest Neighbors y utilizando 30 vecinos se obtiene una AUC de 0,93 y una exactitud del 72,41 %. Aunque el resultado es competitivo comparandolo con el resto de algoritmos aplicados, hay que tener cuidado con el número de vecinos que se elige. Si se aumenta demasiado, sobre todo cuando hay ruido o las clases están mezcladas, el rendimiento puede empeorar.

Por su parte, SVM, con un valor de $C = 0{,}1$, logra una AUC de 0,91 y

una precisión del 70,14 %. Si bien su rendimiento es inferior al de los modelos anteriores, mantiene una buena capacidad de generalización y es menos propenso al sobreajuste que Random Forest, al igual que ocurre en el caso de KNN, siendo una opción fiable cuando se requiere interpretar márgenes de separación más claros.

En este escenario sin validación cruzada, sigue presente el problema del desequilibrio en la distribución de datos en las distintas clases, afectando al rendimiento global. Las clases correspondientes a Larva 6 y Nudo 2 siguen destacando en términos de rendimiento, mientras que otras como Larva 3, Larva 5 y Ruido presentan mayores dificultades a la hora de ser clasificadas, lo cual afecta a la evaluación del rendimiento del modelo completo. Además, se puede observar que, siguiendo la línea del apartado anterior, en el que los datos fueron normalizados, la mayor brecha entre las curvas de aprendizaje se presenta en el caso de Random Forest.

Como conclusión, se observa que la estandarización de los datos proporciona mejores resultados que la normalización. Por ejemplo, en el caso de KNN, la precisión mejora del 58,56 % con datos normalizados al 61,43 % con estandarizados, y la AUC sube de 0,91 a 0,93. Esta mejora se repite en SVM y Random Forest, lo que indica que la estandarización permite una mejor separación entre clases. Esto puede deberse a que, al centrar los datos en media cero y desviación típica 1, se facilita el aprendizaje en algoritmos sensibles a las distancias o a la dispersión de los datos.

4.5. Implementación de los algoritmos con validación cruzada

De cara a tratar de mejorar la capacidad de generalización de los algoritmos anteriormente descritos, se ha implementado en ellos la validación cruzada.

La validación cruzada (CV) es un método de remuestreo muy utilizado en Machine Learning para evaluar el rendimiento de un modelo en datos nuevos, sin necesidad de entrenar y validar siempre con los mismos conjuntos preestablecidos. Consiste en dividir los datos en varios subconjuntos o folders, usando algunos para entrenar (todos menos 1) y el restante para validar, alternando estos roles en varias iteraciones para obtener métricas más precisas y una mejor clasificación de las distintas clases. CV es el nombre que se le ha asignado al número de folds o número de partes en que dividimos los datos a la hora de programar los algoritmos en Python. El empleo de la validación cruzada permite obtener métricas de rendimiento fiables sin depender de una única partición de datos. Además, ayuda a comparar modelos y detectar sobreajuste, siendo una técnica fácil de implementar que maximiza el uso de los datos disponibles.

En este caso, no es relevante la diferencia de instancias entre unas clases y otras a la hora de cambiar el número de folds, puesto que no afecta a la variación

de las mismas. [13]

4.5.1. Normalización de los datos con implementación de validación cruzada

4.5.1.1. Random Forest

En este caso se mantiene constante el número de árboles mientras se varía el valor del hiperparámetro CV. El número de árboles con el que se han obtenido mejores resultados en las pruebas sin validación cruzada es 500. Con ese valor constante se ha ido variando el valor de CV, obteniendo los resultados presentados en la Tabla. 4.14.

CV	**Exactitud**	**AUC**
2	0.7505	0.96
5	0.7561	0.96
10	0.7571	0.96
25	0.7586	0.96

Tabla 4.14: Resultados para diferentes números de validación cruzada (CV) en Random Forest con nº de árboles = 500.

A pesar de que computacionalmente sí requiere una cantidad de recursos considerablemente mayor, observamos una clara mejora a medida que incrementamos el valor de CV. En este caso, el valor de la exactitud se eleva lo suficiente como para decantarse por los resultados obtenidos con CV = 25 y no rechazar este caso por la carga computacional.

Se visualizan los resultados de la clasificación, mostrando la curva ROC promedio de todas las clases y cada una de ellas por separado, reflejadas en la Fig. 4.33.

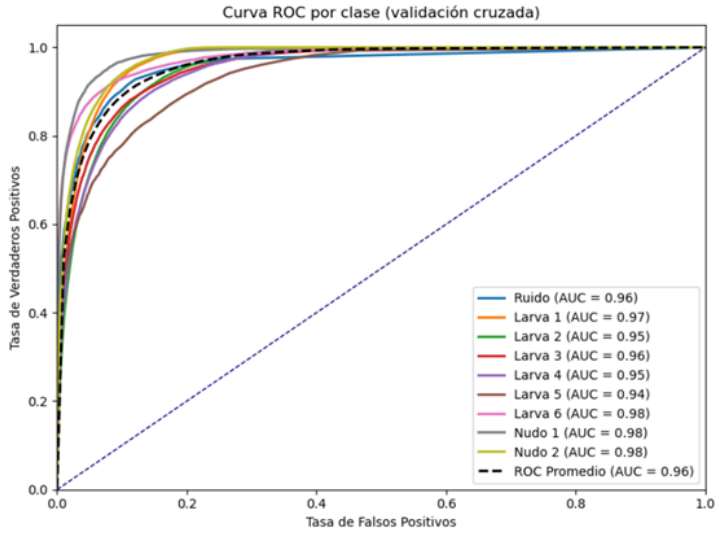


Figura 4.33: Curvas ROC por clase y promediada de Random Forest con CV=25 y 500 árboles.

Y las métricas obtenidas se indican en la Tabla. 4.15.

Clase	**Precisión**	**Recall**	**F1-Score**	**Soporte**
Ruido	0.65	0.32	0.42	1200
Larva 1	0.79	0.78	0.79	25091
Larva 2	0.68	0.80	0.74	26166
Larva 3	0.70	0.71	0.70	17850
Larva 4	0.67	0.64	0.66	15809
Larva 5	0.77	0.26	0.39	2697
Larva 6	0.85	0.80	0.82	19834
Nudo 1	0.77	0.76	0.77	8542
Nudo 2	0.83	0.83	0.83	30812

Tabla 4.15: Métricas de Exactitud, Recall, F1-Score y Soporte por clase

Si comparamos los resultados obtenidos con este modelo frente a los obtenidos sin aplicar validación cruzada, observamos diferencias significativas en las métricas de precisión, recall y F1-score. Estas diferencias nos permiten analizar si la validación cruzada mejora el rendimiento del modelo o si, por el contrario, introduce una pérdida de rendimiento notable.

De manera general, el modelo que implementa validación cruzada tiene mayor precisión en la mayoría de larvas, siendo esta diferencia más pronunciada en clases como la larva 4 y el nudo 2. Esto indica que los modelos que implementan

validación cruzada tienden a ser más conservadores y a clasificar menos falsos positivos.

En cuanto al recall, vemos que el modelo proporciona valores mucho más altos en la mayoría de las clases, a excepción del caso del nudo 2, en el que baja ligeramente. Esto indica que el modelo que implementa la validación cruzada detecta más verdaderos positivos que el que no la implementa. Además, el F1-score mejora en todas las clases, lo que sugiere que el modelo con validación cruzada logra un mejor equilibrio entre precisión y recall de manera general en todas ellas.

En el caso del nudo 2, el recall baja en este modelo con CV. Podemos sacar en claro que el modelo original tenía una mayor tendencia a clasificar los ejemplos en esta categoría, posiblemente generando más falsos positivos.

En cuanto a las curvas de aprendizaje para este modelo, obtenidas a medida que se añaden datos tanto a la hora de entrenar como a la hora de validar, se reflejan en la Fig. 4.34.

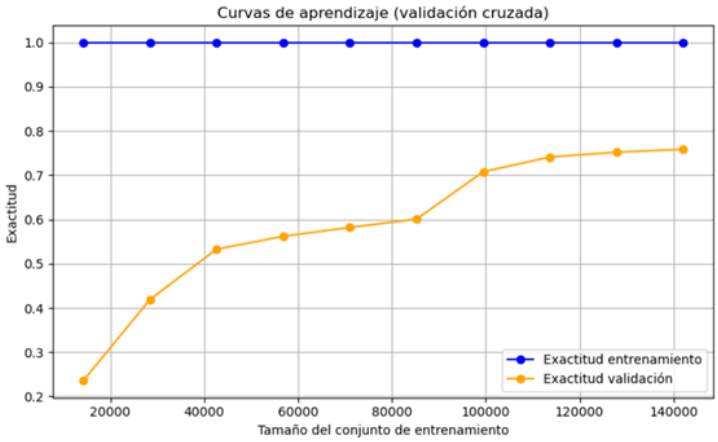


Figura 4.34: Curvas de aprendizaje para Random Forest con 500 árboles y CV = 25.

En la Fig. 4.34, comparándola con la Fig. 4.21, se observan distintas cosas. La curva de aprendizaje de la exactitud del entrenamiento es igual en ambos casos, y se mantiene constante. Esto quiere decir que el modelo, sin importar la cantidad de datos con los que entrene, los está memorizando, lo cual puede dar lugar a que se produzca overfitting.

Sin embargo, al comparar las curvas de aprendizaje de la validación, se encuentran diferencias. En el caso de la Fig. 4.34, el modelo sobreajusta, puesto que el rendimiento de la validación es mucho menor que el del entrenamiento, es decir, hay una brecha considerable entre ambas trayectorias. Aún así, se ve que gracias a la implementación de la validación cruzada, hay margen para que el modelo generalice con los datos y la validación va mejorando.

Como conclusión en este caso, se visualiza que ambos modelos sobreajustan, pero que el modelo que implementa la validación cruzada se ajusta mejor a la realidad.

4.5.1.2. K-Nearest neighbors

Para este mismo modelo sin emplear la validación cruzada se ha obtenido que el número de vecinos con el que mejores métricas se obtienen es 200. Con ese dato invariable, se varía el número de folds empleados en la validación cruzada, obteniendo los resultados presentados en la Tabla. 4.16.

CV	**Exactitud**	**AUC**
2	0.6865	0.94
5	0.6954	0.94
10	0.6983	0.94
25	0.6999	0.94
50	0.7004	0.95
100	0.7005	0.95

Tabla 4.16: Resultados para diferentes números de validación cruzada (CV) en KNN con 200 vecinos.

En este caso concreto, a pesar de que el incremento del número de folds no repercute de manera significativa en el incremento de los valores de la exactitud y la AUC, como el tiempo empleado en ejecutar el código en las distintas iteraciones ha sido prácticamente el mismo, se emplea para el resto de resultados sobre las distintas clases la ejecución con CV = 100.

Se muestran las curvas ROC tanto por clases como promediada. Se reflejan en la Fig. 4.35.

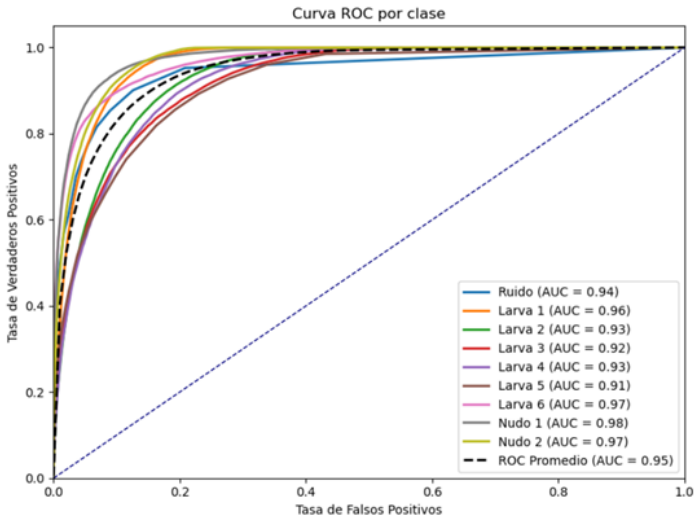


Figura 4.35: Curvas ROC por clase y promediada de KNN con CV=100 y 200 vecinos.

Y las métricas obtenidas se indican en la Tabla. 4.17.

Clase	**Precisión**	**Recall**	**F1-Score**	**Soporte**
Ruido	0.55	0.19	0.29	1200
Larva 1	0.75	0.77	0.76	25091
Larva 2	0.62	0.76	0.68	26166
Larva 3	0.61	0.54	0.57	17850
Larva 4	0.57	0.54	0.55	15809
Larva 5	0.91	0.05	0.09	2697
Larva 6	0.76	0.79	0.77	19834
Nudo 1	0.73	0.68	0.71	8542
Nudo 2	0.81	0.80	0.80	30812

Tabla 4.17: Métricas de Exactitud, Recall, F1-Score y Soporte por clase

Al comparar los resultados obtenidos con el algoritmo KNN con 200 vecinos y validación cruzada con 100 folds frente a los resultados del mismo algoritmo sin validación cruzada, observamos que:

La clase correspondiente al ruido presenta una mejora muy significativa. Pasa de valores de precisión, recall y F1-score de 0 a valores relativamente bajos, pero a ser una clase identificada por el algoritmo. Aun así, el modelo hace muchas predicciones erróneas sobre esa clase y clasifica muchos casos como falsos negativos.

La validación cruzada mejora la estabilidad del modelo, destacando espe-

cialmente en las larvas 1, 3, 4, 5, 6 y en el nudo 2. Se observa un aumento en precisión y recall en la larva 1. En la larva 4, el recall aumenta significativamente, mejorando la detección. La larva 5 presenta un gran salto en precisión, aunque sigue teniendo un recall muy bajo. La larva 6 equilibra mejor sus métricas al incrementar el recall de forma considerable. Finalmente, en el nudo 2, la validación cruzada mejora el F1-score, reduciendo el sesgo en la clasificación.

En cuanto a las curvas de aprendizaje obtenidas para el caso de óptimo funcionamiento del modelo, 200 vecinos, se obtienen las reflejadas en la FIg. 4.36.

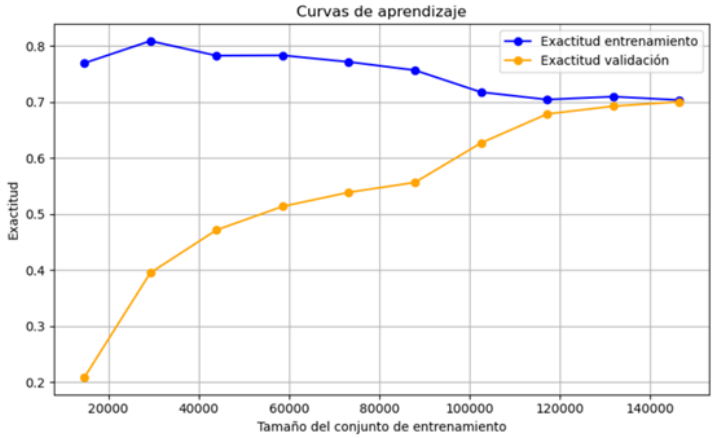


Figura 4.36: Curvas de aprendizaje para KNN con 200 vecinos y CV = 100.

En la Fig. 4.36 se observa que al principio existe una brecha muy notoria, pero a medida que aumenta el número de datos del conjunto, la exactitud de la validación se acerca cada vez más a la de entrenamiento. Con conjuntos pequeños de datos se produce sobreajuste.

Comparándola con la Fig. 4.23 se observa que al aplicar validación cruzada se obtiene una visión más realista del rendimiento del modelo. Conforme aumenta el tamaño del conjunto de entrenamiento, en la Fig. 4.36, la validación cruzada ayuda a estabilizar el rendimiento del modelo, cerrando la brecha entre entrenamiento y validación. Esto sugiere que disponer de más datos podría seguir mejorando la capacidad del modelo para generalizar, lo que no es tan evidente en el caso sin validación cruzada. Esto es muy útil para ajustar el proceso de ajuste.

4.5.1.3. Support Vector Machine

El valor de C para el que se obtienen los mejores resultados con este modelo de clasificación sin implementar la validación cruzada es 0,1. Dicho dato se mantiene constante en las pruebas realizadas tras la implementación de validación cruzada. Tras dichas simulaciones, se obtienen los resultados presentados en la Tabla. 4.18.

CV	Exactitud	AUC
2	0.6175	0.92
5	0.6349	0.93
10	0.6389	0.93

Tabla 4.18: Resultados para diferentes números de validación cruzada (CV) en SVM con $C = 0{,}1$.

Se puede ver claramente que las métricas de exactitud y AUC no mejoran apenas entre las ejecuciones de 5 y 10 folds. Por esto, y porque el tiempo de ejecución es mucho mayor en la segunda, la ejecución escogida para evaluar sus métricas es la de CV = 5.

Se representan en la Fig. 4.37 las curvas ROC de cada una de las clases y la curva ROC promediada.

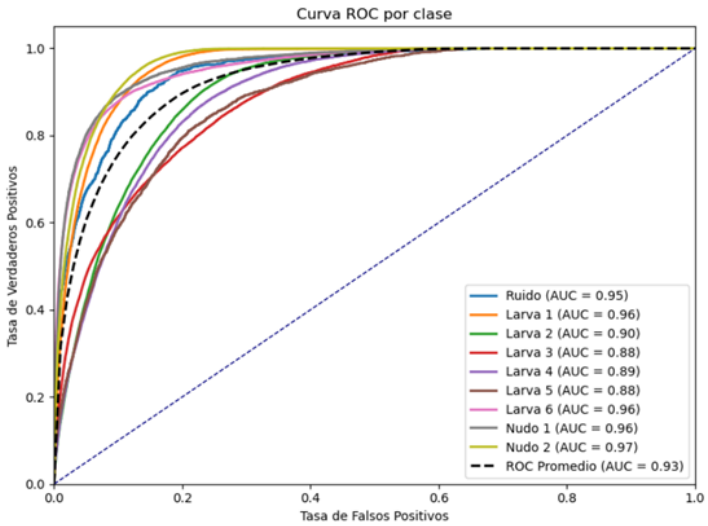


Figura 4.37: Curvas ROC por clase y promediada de SVM para un valor $C = 0{,}1$ y CV = 5.

Y las métricas obtenidas se indican en la Tabla. 4.19.

Clase	Precisión	Recall	F1-Score	Soporte
Ruido	0.02	0.00	0.00	1200
Larva 1	0.77	0.63	0.70	25091
Larva 2	0.51	0.74	0.61	26166
Larva 3	0.50	0.39	0.44	17850
Larva 4	0.51	0.38	0.43	15809
Larva 5	0.00	0.00	0.00	2697
Larva 6	0.67	0.82	0.74	19834
Nudo 1	0.73	0.36	0.49	8542
Nudo 2	0.74	0.86	0.80	30812

Tabla 4.19: Métricas de Exactitud, Recall, F1-Score y Soporte por clase

Al comparar los resultados obtenidos con el modelo SVM con $C = 0.1$ y validación cruzada de 5 folds frente a los obtenidos sin validación cruzada, observamos diferencias significativas en las métricas de precisión, recall y F1-score.

Para las clases de las larvas 1 y 6 mejora notablemente el F1-Score en el modelo con validación cruzada, lo cual reduce el desequilibrio entre la precisión y el recall. Hay una mejora muy notoria también en la capacidad de detección para el caso del nudo 1, con un incremento del valor del recall y el F1-Score muy significativo.

En cuanto al ruido y a la larva 5, el modelo sigue sin clasificarlos correctamente, y las larvas 1 y 6 sufren una disminución en la precisión a costa de una mejora en el recall, lo que indica que el modelo detecta más casos de esas clases.

En general, la validación cruzada en este caso si que mejora la capacidad de generalizar del modelo y los resultados obtenidos de las métricas son más equilibrados.

En cuanto a las curvas de aprendizaje obtenidas para el caso de óptimo funcionamiento del modelo, un valor de C de 0.1, se obtienen las reflejadas en la Fig. 4.38.

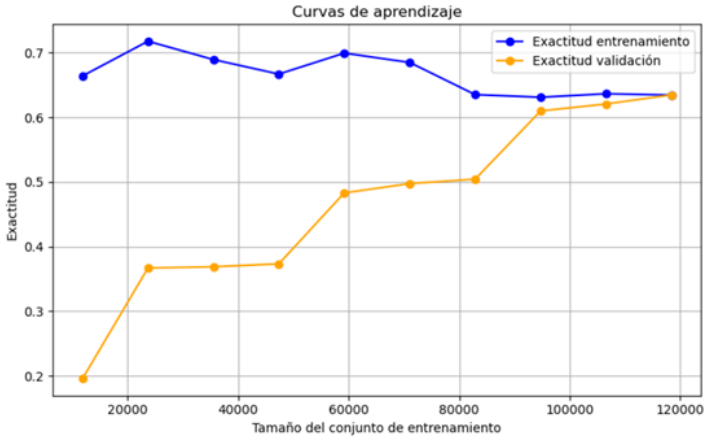


Figura 4.38: Curvas de aprendizaje para SVM con $C = 0{,}1$ y CV $= 5$.

Comparando ambas figuras se observa que, en la Fig. 4.25 la exactitud del entrenamiento y de la validación crecen de forma muy paralela y progresiva, por lo que no existe sobreajuste.

En el caso de la Fig. 4.38 se ve el característico sobreajuste de las primeras etapas de los modelos que implementan validación cruzada, mejorando a medida que aumenta el tamaño del conjunto de entrenamiento, permitiendo una estimación más robusta y más realista que el modelo que no implementa la validación cruzada.

4.5.1.4. Conclusiones para datos normalizados con CV

El modelo de validación cruzada mejora el rendimiento de los algoritmos de clasificación (Random Forest, KNN y SVM) en comparación con los resultados sin validación.

En general, la implementación de la validación cruzada aumenta las métricas de precisión, el recall y el F1-score, lo que indica una mejor generalización de todos los modelos implementados. En Random Forest y KNN, se observa una mejora significativa en clases con pocos datos, como el ruido y algunas clases de larvas, mientras que en SVM, el rendimiento mejora, aunque no tan marcadamente. Sin embargo, el costo computacional aumenta con la validación cruzada, puesto que el número de iteraciones que realizan los algoritmos aumenta. En todos los modelos, se detecta un mayor equilibrio entre precisión y recall, lo que reduce los falsos positivos y mejora su estabilidad. En cuanto a las curvas de aprendizaje, se ve una tendencia a converger de ambas, entrenamiento y validación, en el caso de los modelos KNN y SVM, no siendo así en el caso de Random forest, puesto que tiende a generalizar peor.

4.5.2. Estandarización de los datos con implementación de validación cruzada

4.5.2.1. Random Forest

En este caso, el número de árboles se mantiene constante mientras se varía el parámetro CV. Este valor se determinó en pruebas previas, donde se observó que 200 árboles ofrecían las mejores métricas de clasificación, teniendo en cuenta la carga computacional. Las distintas ejecuciones que se han llevado a cabo proporcionan los resultados presentados en la Tabla. 4.20.

CV	Exactitud	AUC
2	0.7496	0.96
5	0.7556	0.96
10	0.7570	0.96
25	0.7578	0.96

Tabla 4.20: Resultados para diferentes números de validación cruzada (CV) en Random Forest con nº de árboles = 200.

Se evidencia que con un valor de CV = 25 se obtienen los mejores resultados en cuanto a exactitud y a área bajo la curva ROC. A pesar de esto, la carga computacional es tan grande que no compensa la ligera mejora en las métricas que presenta. Por esto, en conjunto, el valor de CV más lógico con el que evaluar la capacidad de clasificación del algoritmo es 10.

Se visualizan los resultados de la clasificación, mostrando la curva ROC promediada y la correspondiente a cada una de las clases. Se reflejan en la Fig. 4.39.

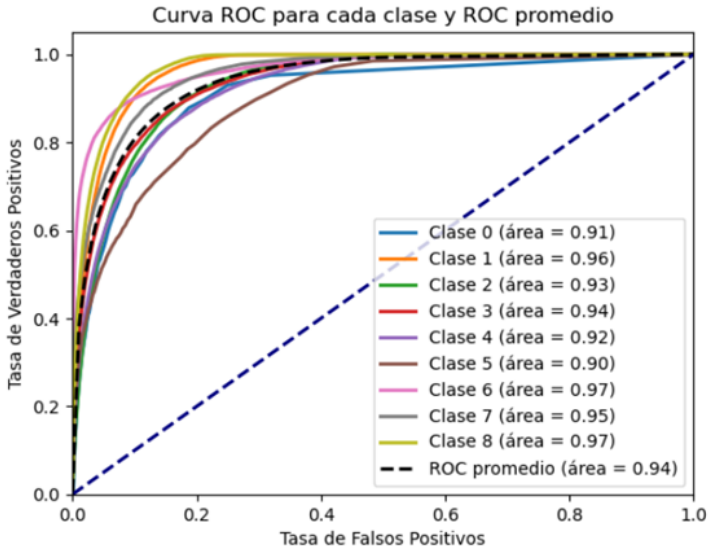


Figura 4.39: Curvas ROC por clase y promediada en Random Forest para 200 árboles y CV = 10.

Y las métricas obtenidas son las indicadas en la Tabla. 4.21.

Clase	**Precisión**	**Recall**	**F1-Score**	**Soporte**
Ruido	0.63	0.32	0.42	1200
Larva 1	0.78	0.78	0.78	25091
Larva 2	0.68	0.80	0.74	26166
Larva 3	0.70	0.70	0.70	17850
Larva 4	0.67	0.64	0.65	15809
Larva 5	0.78	0.26	0.39	2697
Larva 6	0.85	0.80	0.82	19834
Nudo 1	0.77	0.76	0.76	8542
Nudo 2	0.83	0.83	0.83	30812

Tabla 4.21: Métricas de Exactitud, Recall, F1-Score y Soporte por clase

Si empleamos dichos resultados para compararlos con los obtenidos con el modelo de clasificación Random Forest, con 200 árboles y sin implementar la validación cruzada, obtenemos que:

En la comparación se observan diferencias notables en la precisión, recall y F1-score. Estas diferencias nos permiten evaluar si el uso de validación cruzada realmente aporta beneficios al modelo o si, por el contrario, introduce una pérdida de rendimiento significativa.

- La precisión del modelo en cuanto a las distintas clases es más o menos constantes en ambos modelos.
- De igual manera ocurre en los parámetros 'Recall' y 'F1-Score', por lo que el modelo que implementa la validación cruzada no presenta una mejora en la generalización respecto al que no implementa la validación cruzada, pero tampoco un empeoramiento.

En cuanto a las curvas de aprendizaje para este modelo, obtenidas a medida que se añaden datos tanto a la hora de entrenar como a la hora de validar, se obtiene la gráfica reflejada en la Fig. 4.40.

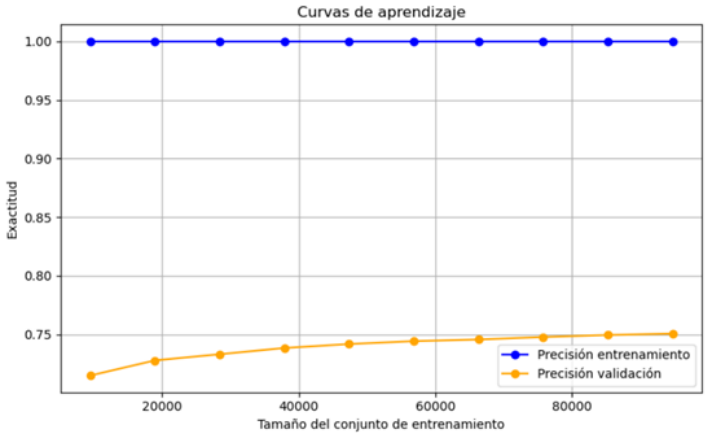


Figura 4.40: Curvas de aprendizaje para Random Forest con 200 árboles y CV = 10.

Se observa en la Fig. 4.40, comparándola con la Fig. 4.27, que tanto la curva de exactitud del entrenamiento como la curva de exactitud de la validación siguen la misma trayectoria a lo largo de la ejecución del modelo.

Por tanto, se puede deducir que el hecho de haber implementado validación cruzada a este algoritmo en este caso, no ha mejorado ni la capacidad de generalizar del modelo ni las diversas métricas asociadas a las distintas clases. Sigue produciéndose overfitting y en este caso la validación cruzada no ha sido la solución.

4.5.2.2. K-Nearest Neighbors

Puesto que en pruebas anteriores se dedujo que para el conjunto de datos estandarizado con el que se está trabajando, el nº de vecinos que maximizaba el rendimiento era 30, es ese dato el que se ha utilizado para hacer las pruebas empleando validación cruzada. Se han obtenido los resultados presentados en la Tabla. 4.22.

CV	Exactitud	AUC
2	0.7109	0.93
5	0.7182	0.93
10	0.7207	0.93
25	0.7217	0.93
50	0.7221	0.93
100	0.7222	0.93

Tabla 4.22: Resultados para diferentes números de validación cruzada (CV) en KNN con 30 vecinos.

Se observa que, a medida que se aumenta el número de folds en el algoritmo, la exactitud aumenta gradualmente, no siendo así con la AUC, puesto que se mantiene constante independientemente del valor del parámetro CV.

Que la AUC se mantenga constante indica que el modelo tiene la capacidad de distinguir entre clases de manera bastante robusta y que no varía esta distinción en función de cómo se realice la partición de los datos. A pesar de esto, sí que sería interesante trabajar en torno a los 100 folds, puesto que de trabajar con menos no estaríamos optimizando el algoritmo y de trabajar con más le estaríamos agregando una carga computacional que no está acorde a la mejora de los resultados.

El hecho de que la exactitud aumente gradualmente con el aumento del número de folds indica que se está favoreciendo una mejor generalización del modelo de cara a la clasificación de datos nuevos.

Se muestran las curvas ROC tanto promedio como la que representa a cada una de las clases, en la Fig. 4.41.

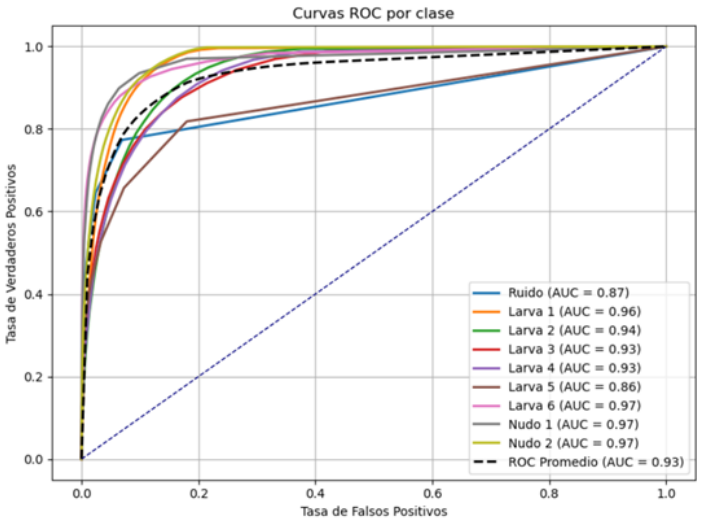


Figura 4.41: Curvas ROC por clase y promediada en KNN para 30 vecinos y CV=100.

Y las métricas obtenidas son las indicadas en la Tabla. 4.23.

Clase	**Precisión**	**Recall**	**F1-Score**	**Soporte**
Ruido	0.58	0.24	0.34	1200
Larva 1	0.75	0.79	0.77	25091
Larva 2	0.62	0.80	0.70	26166
Larva 3	0.66	0.60	0.63	17850
Larva 4	0.62	0.58	0.60	15809
Larva 5	0.80	0.13	0.23	2697
Larva 6	0.81	0.79	0.80	19834
Nudo 1	0.79	0.65	0.71	8542
Nudo 2	0.83	0.78	0.80	30812

Tabla 4.23: Métricas de Exactitud, Recall, F1-Score y Soporte por clase

Al comparar los resultados obtenidos con el modelo de clasificación KNN con 30 vecinos y validación cruzada (CV=100) con los del mismo modelo sin validación cruzada, se observan diferencias significativas en las métricas de precisión, recall y F1-score. Estas diferencias permiten evaluar si la validación cruzada aporta beneficios en términos de generalización o si, por el contrario, afecta negativamente al rendimiento del modelo.

En cuanto al recall, no existe un cambio notable entre las clases del modelo entrenado con y sin validación cruzada, lo que indica que el modelo con CV detecta el mismo número de instancias de estas clases en comparación con el

modelo sin CV. Sin embargo, se observa una ligera mejora en la precisión de la larva 5, lo que sugiere que el modelo con validación cruzada comete menos falsos positivos en ese caso. El F1-score se mantiene relativamente estable en la mayoría de las clases, lo que indica que la validación cruzada no introduce grandes variaciones en el equilibrio entre precisión y recall.

Respecto a las curvas de aprendizaje para este modelo, obtenidas a medida que se añaden datos tanto a la hora de entrenar como a la hora de validar, se reflejan en la Fig. 4.42.

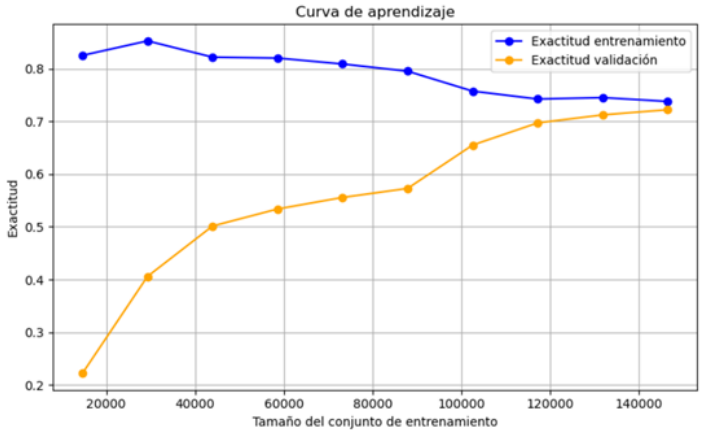


Figura 4.42: Curvas de aprendizaje para KNN con 30 vecinos y CV = 10.

Se observa en la Fig.4.42, comparándola con la Fig.4.29, que ambas curvas presentan trayectorias completamente opuestas. En cuanto a las curvas de entrenamiento, en el caso sin validación cruzada Fig.4.29 se aprecia una pendiente positiva, lo que indica que el modelo mejora su rendimiento conforme dispone de más datos para entrenar, aprovechando toda la información disponible.

En cambio, en el modelo con validación cruzada Fig.4.42, la curva de entrenamiento muestra una pendiente negativa. Esto se debe a que el modelo entrena con diferentes subconjuntos de datos en cada pliegue, lo que limita su capacidad de memorizar los ejemplos y reduce su rendimiento en entrenamiento, permitiendo normalmente incrementarlo a la hora de validar. Esta estrategia permite una evaluación más realista de la capacidad de generalización del modelo, evitando el sobreajuste y proporcionando una evaluación más fiable de los casos no vistos y por tanto su clasificación.

En la curva de aprendizaje de la exactitud en la validación se observa una diferencia principal entre ambas figuras. Aunque las dos presentan una pendiente positiva, la curva de la Fig.4.42, correspondiente al modelo con validación cruzada, parte de una exactitud mucho menor que la observada en el caso sin validación cruzada Fig.4.29. Esta diferencia inicial se debe a que, en la validación cruzada,

el modelo comienza entrenando con pliegues muy reducidos, lo que dificulta su capacidad de generalización en las primeras iteraciones. Sin embargo, a medida que aumenta el tamaño del conjunto de entrenamiento, la exactitud mejora de forma sostenida hasta alcanzar valores similares al caso sin CV. Este crecimiento progresivo refleja una evaluación más realista y fiable del rendimiento, ya que el modelo es sometido a múltiples particiones del conjunto de datos y no se beneficia de una estructura de entrenamiento fija.

Se puede concluir con que el hecho de haber implementado validación cruzada a este algoritmo en este caso ha mejorado ligeramente la capacidad de generalizar del modelo o la tendencia que presenta. Las diversas métricas asociadas a las distintas clases, sin embargo, se mantienen con valores similares. El hecho de implementar validación cruzada y de que la curva de aprendizaje de entrenamiento y validación converjan reduce el riesgo de sobreajuste, por lo que es una mejora respecto al anterior.

4.5.2.3. Support Vector Machine

El valor del parámetro C con el que se obtuvieron mejores resultados en el caso anterior en el que el algoritmo se implementa sin CV es 10. Por tanto, es el dato que se ha empleado para implementar en el algoritmo la validación cruzada. Los datos obtenidos de las simulaciones se presentan en la Tabla. 4.24.

CV	**Exactitud**	**AUC**
2	0.7448	0.96
5	0.7497	0.96
10	0.7502	0.96
25	0.7510	0.96
100	0.7511	0.96

Tabla 4.24: Resultados para diferentes números de validación cruzada (CV) en SVM con $C = 10$.

A medida que se incrementa el valor del hiperparámetro CV se ve que mejoran los valores de la exactitud . Por esto, se decide utilizar CV = 25, puesto que es la situación que más favorece la correcta generalización del algoritmo de clasificación empleado, teniendo en cuenta la carga computacional y el tiempo empleado también.

Se representan las curvas ROC promedio y las de las distintas clases empleadas en la clasificación, en la Fig. 4.43.

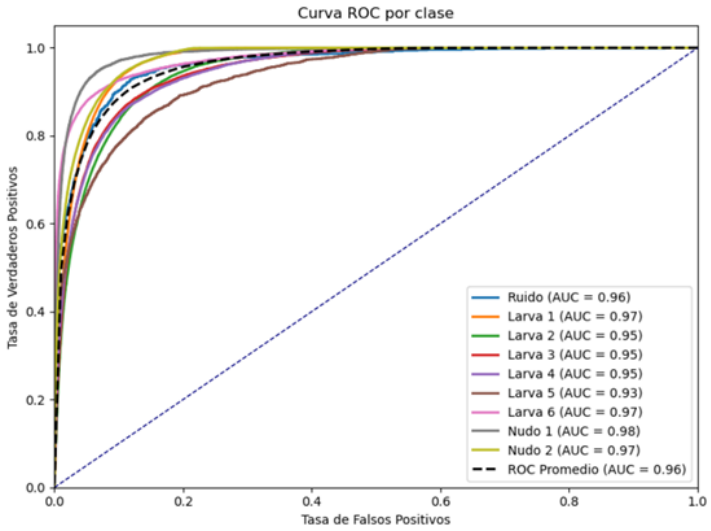


Figura 4.43: Curvas ROC por clase y promediada en SVM con CV=25 para $C = 10$.

Y las métricas obtenidas se indican en la Tabla. 4.25.

Clase	Precisión	Recall	F1-Score	Soporte
Ruido	0.66	0.31	0.42	1200
Larva 1	0.77	0.79	0.78	25091
Larva 2	0.67	0.79	0.73	26166
Larva 3	0.70	0.66	0.68	17850
Larva 4	0.67	0.65	0.66	15809
Larva 5	0.79	0.23	0.35	2697
Larva 6	0.85	0.80	0.83	19834
Nudo 1	0.75	0.76	0.76	8542
Nudo 2	0.83	0.81	0.82	30812

Tabla 4.25: Métricas de Exactitud, Recall, F1-Score y Soporte por clase

Si comparamos los resultados obtenidos con el modelo SVM con $C = 10$ y validación cruzada de 25 folds frente al mismo modelo sin validación cruzada, observamos ciertas diferencias en las métricas de clasificación.

Al comparar los resultados del modelo con y sin validación cruzada, se ve que en general las métricas se mantienen bastante estables, e incluso mejoran ligeramente en algunas clases. Por ejemplo, Larva 3 y Ruido tienen un pequeño aumento en el F1-score, y Larva 1, 2 y 4 se comportan de forma muy similar en ambos casos. Larva 6 y los Nudos siguen destacando con buenos resultados, lo cual es una buena señal. La única clase que baja un poco es Larva 5, seguramente

porque tiene muy pocas muestras y eso hace que los resultados varíen más.

Concluyendo, usar validación cruzada ayuda a tener una evaluación más completa y fiable del modelo.

En cuanto a las curvas de aprendizaje para este modelo, obtenidas a medida que se añaden datos tanto a la hora de entrenar como a la hora de validar, se presentan en la Fig. 4.44.

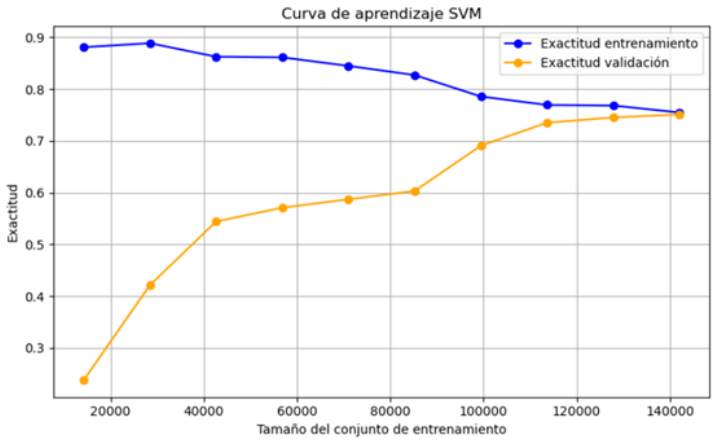


Figura 4.44: Curvas de aprendizaje para SVM con $C = 10$ y CV $= 25$.

Al comparar las curvas de aprendizaje, se nota una diferencia clara entre el modelo sin validación cruzada, el visto en la Fig. 4.31, y el que sí la utiliza, el visto en la Fig. 4.44. En la Fig. 4.31, la diferencia entre la exactitud en entrenamiento y validación es relativamente pequeña y ambas mejoran de forma estable al aumentar los datos, aunque la validación sube más lentamente. En cambio, con validación cruzada, la curva de entrenamiento parte muy alta pero desciende al aumentar el conjunto, mientras que la validación mejora notablemente y se va acercando.

Esto indica que el modelo que implementa la validación cruzada generaliza mejor y evita el sobreajuste, ya que no memoriza tanto los datos utilizados durante el entrenamiento y aprende a adaptarse mejor a datos nuevos.

4.5.2.4. Conclusiones para datos estandarizados con CV

La implementación de la validación cruzada mejora el rendimiento de los algoritmos de clasificación empleados (Random Forest, KNN y SVM) en comparación con los resultados sin validación vistos anteriormente. En general, se observa un aumento en la precisión, el recall y el F1-score, y por tanto en la generalización de los modelos. En el caso de Random Forest y KNN, la validación cruzada muestra una mejora notable en las clases que presentan un número de

muestras menor, mientras que en SVM, el rendimiento también mejora, aunque no de manera tan destacable. No obstante, el costo computacional se incrementa con la validación cruzada.

Al igual que en el apartado en el que se implementa validación cruzada pero se normalizan los datos, la convergencia en las curvas de aprendizaje sucede en los algoritmos KNN y SVM, no siendo así en el caso de Random Forest, por lo que no está dando indicios de ser un modelo fiable, al no tener tanta capacidad de generalización.

Usar validación cruzada en los algoritmos de clasificación tiene varias ventajas importantes. Permite evaluar el modelo de forma más fiable, ya que lo entrena y lo prueba en distintos subconjuntos de datos. Esto hace que sea menos probable que el modelo se ajuste demasiado a los datos de entrenamiento y no generalice bien con datos nuevos.

Al utilizar validación cruzada, se obtiene una mayor fiabilidad de cómo el modelo se comportará en situaciones realistas, puesto que evalúa su desempeño en diversas particiones de los datos. Además, al comparar varios algoritmos de clasificación bajo el mismo enfoque de validación cruzada, se pueden obtener comparaciones más justas y equilibradas entre ellos, evitando sesgos que surgen cuando solo se utiliza un conjunto de entrenamiento y otro de prueba. Por esto, la validación cruzada es una herramienta esencial para mejorar la fiabilidad y la estabilidad de los modelos.

4.6. Algoritmos para la discriminación entre nudos y larvas

Una vez que se han implementado diferentes algoritmos para tratar de diferenciar entre las 8 clases existentes (las 6 clases correspondientes a larvas y las 2 a nudos), se decide realizar otras pruebas siguiendo la línea consistente en distinguir únicamente entre nudos y larvas, tratando de que de esta manera, la exactitud, la precisión, el recall y la f1 score mejoren sus valores.

Para esta implementación se ha partido de los códigos utilizados en apartados anteriores y se han realizado ciertas modificaciones. Los datos se han estandarizado, puesto que anteriormente fue la manera de escalar los datos que mejores resultados proporcionó, y se ha mantenido la validación cruzada activa. Para la implementación, se ha considerado un clasificador binario, asignando la clase 0 a las clases anteriormente pertenecientes a las larvas (de la clase 1 a la clase 6 ambas incluidas) y se ha asignado la clase 1 a las que anteriormente pertenecían a nudos (las clases 7 y 8).

4.6.0.1. Random Forest

En este caso, se ha implementado un modelo de clasificación utilizando Random Forest para diferenciar entre larvas y nudos en el conjunto de datos.

Para optimizar su rendimiento y seleccionar los mejores hiperparámetros, se ha utilizado validación cruzada estratificada junto con una búsqueda en rejilla (GridSearchCV).

Como parte del preprocesamiento, los datos se han estandarizado de manera previa a entrenar el clasificador, lo que evita que ciertas características dominen sobre otras debido a diferencias en escala.

La optimización del modelo se ha centrado en dos hiperparámetros clave:

- Número de árboles en el bosque: Se han evaluado valores entre 10 y 200 en incrementos de 20.
- Profundidad máxima de los árboles: Se han probado los valores None, 10, 20 y 30. Este parámetro define cuán profundo puede crecer cada árbol antes de detenerse. Si se deja como None, los árboles crecen hasta que todas las hojas sean puras o contengan menos del número mínimo de muestras requerido. Un valor como None puede llevar al sobreajuste, mientras que un valor demasiado bajo podría impedir que el modelo aprenda patrones complejos en los datos [14]. En este caso, se implementa como hiperparámetro, dejando que sea el propio algoritmo el que elija el valor óptimo. De esta manera se evita tanto el sobreajuste como el infraajuste y este se ajusta a la complejidad de los datos del conjunto.

Tras seleccionar la mejor combinación de hiperparámetros, se ha realizado una evaluación detallada del modelo, y se h generado un reporte de clasificación para analizar las métricas de precisión, recall y F1-score, una matriz de confusión para visualizar los aciertos y errores en la clasificación, y una curva ROC para evaluar la capacidad de discriminación del modelo entre las distintas clases de larvas y nudos.

Los valores obtenidos en función del CV para el número óptimo de árboles y para la óptima profundidad son los presentados en la Tabla. 4.26.

CV	**Número de árboles**	**Profundidad máxima**	**Exactitud**
5	190	None	0.9017
10	190	30	0.9020
30	170	30	0.9022
60	170	None	0.9022
200	190	30	0.9024

Tabla 4.26: Valores de 'número de árboles', 'Profundidad máxima' y 'Exactitud' para distintos valores de CV

En este caso y, a pesar de que se puede observar que, a medida que incrementamos el número de folders, incrementa la exactitud del modelo, no se han podido realizar simulaciones con más folders por el tiempo de computación desmesurado

requerido. En cuanto a la reducción del número óptimo de árboles a medida que se incrementa CV, esto denota que el modelo generaliza mejor y necesita menos árboles para alcanzar buenos resultados.

Teniendo esto en cuenta, la ejecución que mejores resultados proporciona es la que se realiza con CV = 60.

La curva ROC que se obtiene en este caso y con la que se determina el parámetro AUC es la reflejada en la Fig. 4.45.

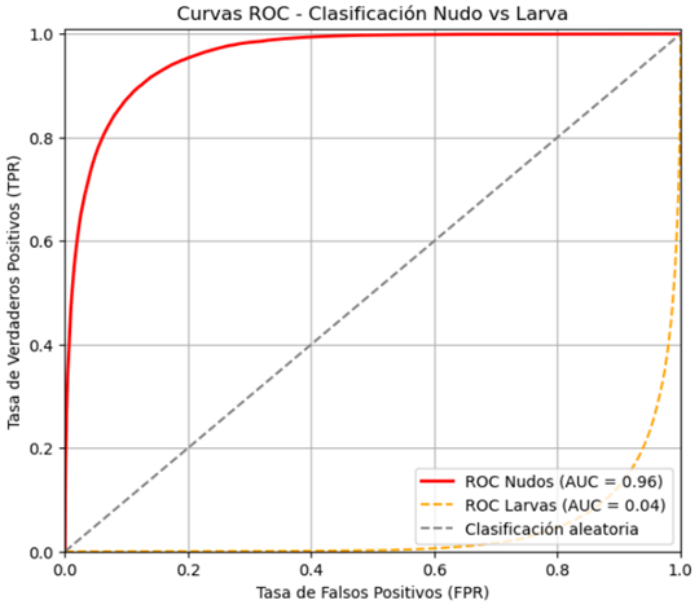


Figura 4.45: Curva ROC del algoritmo RF discrimando entre nudos y larvas

Para el caso decidido como mejor balance entre estabilidad y rendimiento, se obtiene el reporte de clasificación indicado en la Tabla. 4.27.

Clase	**Precisión**	**Recall**	**F1-Score**	**Soporte**
0.0	0.93	0.94	0.93	108643
1.0	0.83	0.79	0.81	39354

Tabla 4.27: Métricas de Exactitud, Recall, F1-Score y Soporte por clase

La clase 0.0 (larvas) tiene unas métricas bastante altas, con un 93 % de precisión y un 94 % de recall, lo que indica que el modelo detecta muy bien estos casos. En cambio, para la clase 1.0 (nudos), los valores bajan un poco (83 % de precisión y 79 % de recall), aunque siguen siendo aceptables, sobre todo teniendo

en cuenta que hay menos ejemplos de esta clase. En general, el modelo consigue un buen equilibrio entre acertar y no dejarse casos sin detectar.

Se ha implementado, como en casos anteriores, la curva de aprendizaje, de cara a visualizar la exactitud tanto a lo largo del entrenamiento como de la validación. Se reflejan en la Fig. 4.46.

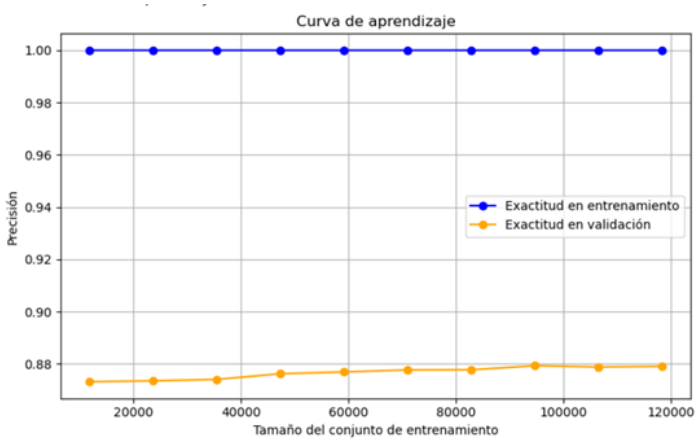


Figura 4.46: Curvas de aprendizaje para el clasificador 'Nudo VS Larva' Random Forest con CV = 60.

El la Fig. 4.46 se observa la gran brecha que hay entre las curvas de exactitud del entrenamiento y de la validación. Esto, junto con que la curva de validación apenas mejora a medida que se incrementa el tamaño del conjunto del entrenamiento, indica un sobreajuste. No logra generalizar bien con nuevos datos. Para solucionar esto se puede tomar la decisión de imponer un valor al hiperparámetro 'Profundidad máxima' menor que el actual para esa ejecución, 'None'.

Se ha ejecutado el mismo modelo, ajustando el hiperparámetro de 'Profundidad máxima' a 10, con el resultado observado en la Fig. 4.47.

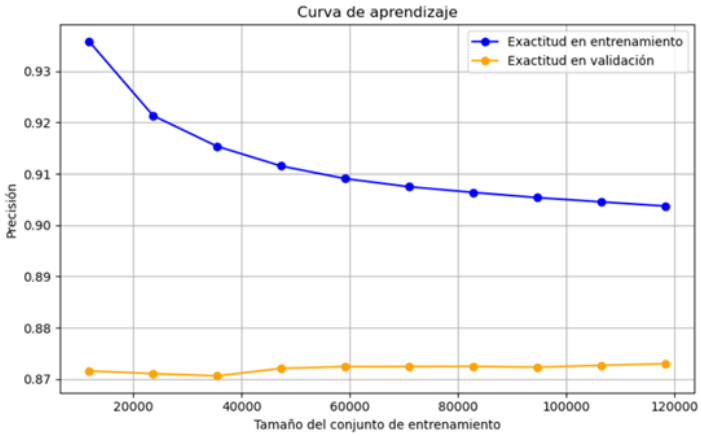


Figura 4.47: Curvas de aprendizaje para el clasificador 'Nudo VS Larva' Random Forest con CV = 60 y 'Max_depth = 10.

y ajustando el mismo hiperparámetro a 20, con la gráfica vista en la Fig. 4.48 como resultado.

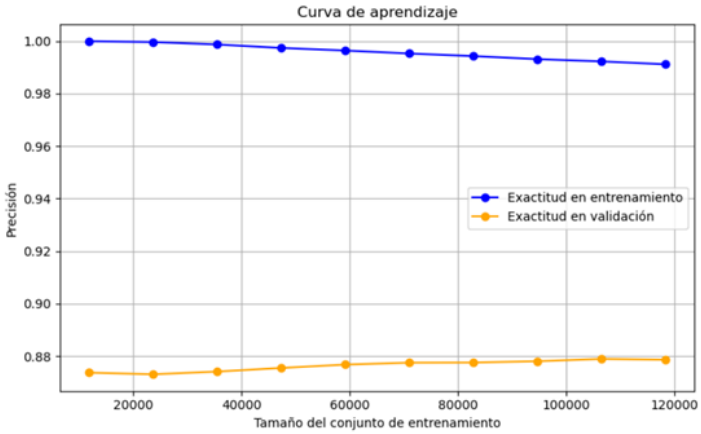


Figura 4.48: Curvas de aprendizaje para el clasificador 'Nudo VS Larva' Random Forest con CV = 60 y 'Max_depth = 20.

Se puede observar que la brecha entre las curvas en ambos casos se reduce, comparándolo con la Fig. 4.46. A pesar de esto, la exactitud en la curva de validación no mejora, por lo que el modelo sigue sin generalizar bien a pesar de haber fijado un valor más bajo de 'Profundidad máxima'. No es una solución en este caso el cambio de dicho parámetro para evitar el sobreajuste.

4.6.0.2. K Nearest neighbors

En este caso se ha implementado de cara a distinguir entre larvas y nudos en un conjunto de datos un modelo KNN de clasificación. La validación cruzada estratificada y una búsqueda de hiperparámetros (GridSearchCV) están también implementadas en este caso.

Se ha definido una búsqueda de hiperparámetros en función del número de vecinos, evaluando valores en el rango [10, 200] con incrementos de 20. A su vez, se ha empleado validación cruzada estratificada con un número de divisiones o folds variable para medir la precisión de cada una de las posibles configuraciones. Para la evaluación del modelo se selecciona el mejor número de vecinos basado en la mayor precisión obtenida en la validación cruzada y se genera un reporte con las métricas más relevantes y con la curva ROC que permite visualizar la capacidad de discriminación del algoritmo entre nudos y larvas.

Los valores obtenidos en función del CV para el número óptimo de vecinos y para la precisión son los presentados en la Tabla. 4.28.

CV	**Mejor número de vecinos**	**Exactitud**
5	30	0.8886
10	30	0.8890
30	30	0.8895
60	30	0.8897
100	30	0.8898
500	30	0.8898
1000	30	0.8897

Tabla 4.28: Valores de 'número de vecinos' y 'Exactitud' para distintos valores de CV

Se observa que para todos los casos el número óptimo de vecinos ha sido 30, y la exactitud se estabiliza cuando se llega a valores de entre 100 y 500 folds, reduciéndose progresivamente si se eleva ese valor. Esta pequeña reducción se debe a un aumento en la variabilidad de los resultados debido a la menor cantidad de datos en cada partición de entrenamiento.

Puesto que la variación del tiempo de ejecución no es reseñable, se considera que el valor de CV para el que se obtienen mejores resultados es 100. El desempeño del modelo es sólido en la clasificación entre larvas y nudos, y la curva ROC sugiere una buena discriminación entre ambas clases, confirmando que el modelo es adecuado para esta tarea. Queda reflejada dicha curva ROC en la Fig. 4.49.

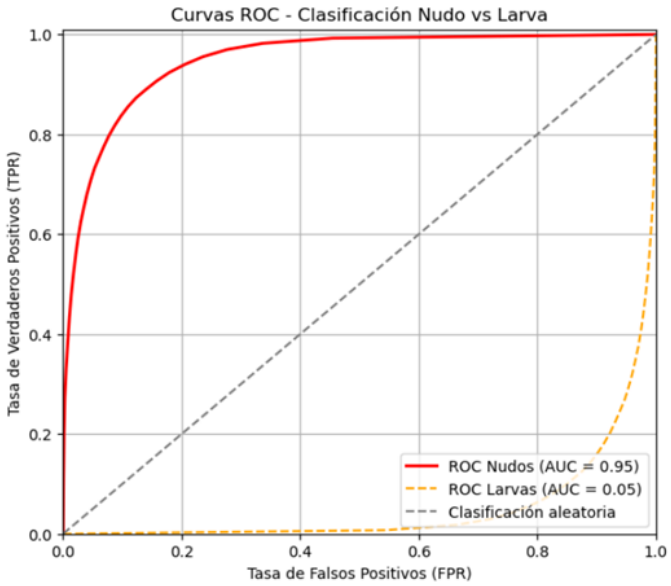


Figura 4.49: Curva ROC del algoritmo KNN discrimando entre nudos y larvas y con CV = 100.

Para el caso decidido como mejor balance entre estabilidad y rendimiento, se obtiene el reporte de clasificación visto en la Tabla. 4.29.

Clase	**Precisión**	**Recall**	**F1-Score**	**Soporte**
0.0	0.91	0.95	0.93	108643
1.0	0.83	0.73	0.78	39354

Tabla 4.29: Métricas de Exactitud, Recall, F1-Score y Soporte por clase

El modelo es muy preciso para identificar larvas, pero tiene más dificultad para identificar correctamente los nudos. Esto puede deberse a la descompensación de datos, puesto que hay muchos más datos pertenecientes a las larvas que a los nudos. Aplicando un balanceo de datos podrían subir las métricas para los nudos también.

En este caso, la curva de aprendizaje obtenida se visualiza en la Fig. 4.50.

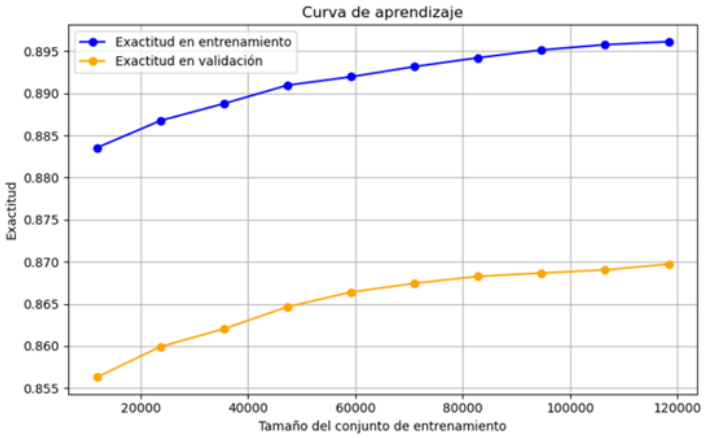


Figura 4.50: Curvas de aprendizaje para el clasificador 'Nudo VS Larva' KNN con CV = 100.

El la Fig. 4.50 se observa que a pesar de que la brecha entre las curvas de exactitud de entrenamiento y de validación, que puede ser consecuencia del sobreajuste, la pendiente sube a medida que se incrementa el tamaño del conjunto de entrenamiento. El modelo, aunque ligeramente, sí que se beneficia de dicho incremento y no ha alcanzado su límite de capacidad de generalización. Muestra menos riesgo de sobreajuste que el visto en la Fig. 4.46.

4.6.0.3. Support Vector Machine

En este caso se utiliza el clasificador SVM para diferenciar entre los sonidos generados por larvas y los provenientes de los nudos. Uno de los problemas encontrados al aplicarlo es que la optimización del parámetro C en el modelo SVM es muy costosa, puesto que requiere altos tiempos de computación. Incluso cuando se intentaron 20,000 iteraciones para encontrar el valor adecuado de C, no fue posible obtener una optimización completa en el tiempo disponible debido a la gran cantidad de datos y al proceso de validación cruzada implementado.

El parámetro C controla cuánto penaliza el modelo los errores de clasificación. Un valor más alto de C hace que el modelo intente ajustarse rigurosamente a los datos del entrenamiento, tratando de clasificar todos los datos correctamente, mientras que un valor bajo de dicho hiperparámetro permite que el modelo tolere más errores, haciendo el modelo más generalizable. Sin embargo, optimizar C requiere mucho tiempo y recursos, especialmente cuando se hace en todo el conjunto de datos con validación cruzada.

Dado que los recursos computacionales disponibles son limitados y no permiten optimizar C, se ha optado por emplear la siguiente estrategia. Se ha entrenado un modelo SVM usando la clase SGDClassifier de scikit-learn, que permite implementar un clasificador lineal con el algoritmo de optimización

conocido como Stochastic Gradient Descent (SGD) [15]. Este método entrena el modelo de forma iterativa procesando pequeños bloques de datos, lo que reduce significativamente la carga computacional y hace posible ajustar el modelo incluso con grandes volúmenes de datos.

En lugar de tratar de optimizar C de manera eficiente, se opta por optimizar el parámetro *alpha* para controlar la regularización del modelo. La regularización es una técnica que evita que el modelo se sobreajuste, es decir, que no aprenda demasiado de las peculiaridades de los datos empleados a lo largo del entrenamiento y que no se aplican en los datos nuevos. Valores altos de *alpha* penalizan al modelo, simplificándolo y reduciendo el sobreajuste, a diferencia de valores más reducidos de *alpha*, con los que el modelo puede alcanzar una mayor complejidad, asumiendo la existencia de un mayor riesgo de sobreajuste.

Se decide optimizar este parámetro porque es más sencillo en términos computacionales que hacerlo con C, y al hacerlo se puede mejorar la capacidad del modelo para generalizar bien a nuevos datos. Además, SGDClassifier permite explorar diferentes tipos de regularización, como l2 (también conocida como ridge), y elasticnet, que combina l1 (lasso) y l2 [16]. Esta última opción resulta especialmente interesante en conjuntos de datos con muchas características, como en este caso, ya que:

- La parte l1 de elasticnet puede eliminar características irrelevantes, reduciendo la dimensionalidad efectiva del problema.
- La parte l2 ayuda a estabilizar el modelo, evitando que dependa en exceso de una sola variable.

Para obtener un equilibrio entre las dos penalizaciones implementadas se tiene un tercer parámetro, l1_ratio, que permite encontrar una combinación eficaz de parámetros sin un consumo excesivo de recursos computacionales [17].

Las ventajas que proporciona optimizar *alpha* en vez de C son:

- **Menor carga computacional**: La optimización de *alpha* es mucho más rápida que la de C debido a que afecta a la regularización en lugar de a la penalización de errores en las clasificaciones. Al ser menos costosa computacionalmente, la optimización de *alpha* es viable incluso con recursos computacionales limitados.
- **Prevención del sobreajuste**: La regularización controla el sobreajuste de manera eficaz, lo que puede ser más crítico en modelos con un gran número de características (como en este caso, con un conjunto de datos grande y complejo). Optimizar *alpha* permite que el modelo sea más generalizable, evitando que se ajuste en exceso a los datos de entrenamiento.
- **Ajuste más flexible**: La regularización elasticnet, que utiliza *alpha*, es buena para trabajar con datos que tienen muchas características. Además, ayuda a quitar las que no son tan importantes, así el modelo queda más sencillo y fiable, sin llegar a tiempos de computación insostenibles.

Las desventajas de optimizar *alpha* en vez de C son:

- **Menos control sobre el ajuste a los datos**: C es un parámetro más directo para controlar el ajuste del modelo a los datos. Optimizar C puede proporcionar un ajuste más preciso en los casos en los que se desea maximizar la exactitud de clasificación, algo que *alpha* no puede hacer directamente.
- **Posible subajuste en ciertos casos**: Al optimizar *alpha* para regularizar el modelo, existe el riesgo de que se subajuste a los datos, especialmente si el valor de *alpha* es demasiado alto. Esto se traduce en una reducción de la capacidad del modelo para capturar patrones complejos en los datos.
- **Menos enfoque en la clasificación perfecta**: Aunque *alpha* ayuda a evitar tanto el sobreajuste como el subajuste, no se centra tanto en la reducción de errores de clasificación. Optimizar C podría ser más apropiado en caso de necesitar obtener la mayor precisión posible del modelo, ya que se enfoca en reducir errores de clasificación en lugar de controlar la complejidad del modelo.

Para ajustar el modelo se definió el siguiente espacio de búsqueda de hiperparámetros:

```
param_grid = {
    'svm__alpha': [1e-4, 5e-4, 1e-3],
    'svm__penalty': ['l2', 'elasticnet'],
    'svm__l1_ratio': [0.15, 0.5, 0.85]
}
```

Aunque el espacio de búsqueda es pequeño, esta estrategia permite encontrar un buen modelo sin necesidad de un gran gasto de recursos. Así, se optimiza el parámetro *alpha* para mejorar la capacidad del modelo de generalizar, mientras que se mantiene el proceso eficiente en términos de tiempo y recursos computacionales.

Una vez que hemos encontrado los mejores hiperparámetros, aplicamos el modelo de clasificación al conjunto completo de datos. A partir de ahí, generamos un reporte con métricas como precisión, recall y F1-score, que nos ayudan a entender qué tan bien está funcionando el modelo. También creamos la matriz de confusión para ver cuántos aciertos y errores cometemos en la clasificación, y trazamos la curva ROC para evaluar cómo de bien distingue el modelo entre larvas y nudos.

Los valores obtenidos en función del CV para el hiperparámetro *alpha* son los presentados en la Tabla. 4.28.

CV	Mejor alpha	Exactitud	AUC
5	0.0001	0.8503	0.86
10	5e-05	0.8487	0.86
30	0.0001	0.8498	0.86
60	0.0001	0.8494	0.86
100	0.0001	0.8499	0.87
500	0.0001	0.8487	0.87
1000	0.0001	0.8495	0.86

Tabla 4.30: Valores de 'alpha' y 'Exactitud' para distintos valores de CV

El valor óptimo de alpha se mantiene en 0.0001 en todos los casos salvo en el que implementa CV = 10, con una mayor exactitud al usar 5 folds, aunque esta cae ligeramente con más particiones. A mayor número de folds, la AUC mejora inicialmente (mayor estabilidad en la separación de clases), pero también comienza a descender con CV = 1000, reflejando el impacto de la mayor varianza al entrenar con conjuntos más pequeños. Esto sugiere que un número moderado de folds logra un buen equilibrio entre exactitud y capacidad de generalización del modelo.

Uno de los valores más equilibrados será el de CV = 100, ya que mantiene una exactitud alta y alcanza el valor máximo de AUC sin la inestabilidad que corresponde al uso de un número elevado de folds. Es un modelo fiable.

La curva ROC que se obtiene de dicha simulación se refleja en la Fig. 4.51.

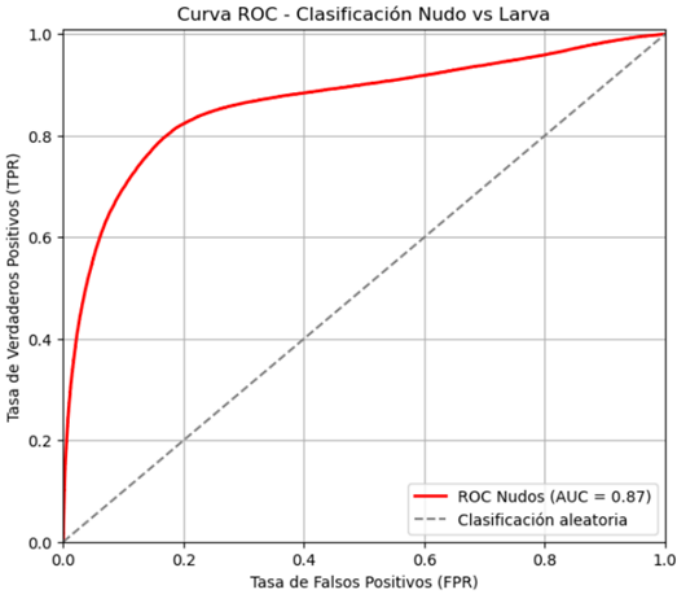


Figura 4.51: Curva ROC del algoritmo SVM discriminando entre nudos y larvas.

Para el caso decidido como mejor balance entre estabilidad y rendimiento, se obtiene el reporte de clasificación visto en la Tabla. 4.31.

Clase	**Precisión**	**Recall**	**F1-Score**	**Soporte**
0.0	0.88	0.92	0.90	108643
1.0	0.75	0.65	0.70	39354

Tabla 4.31: Métricas de Exactitud, Recall, F1-Score y Soporte por clase

Como conclusión, vemos que la clase 0 presenta valores altos en precisión, recall y F1-score, lo que indica que el modelo identifica larvas con bastante fiabilidad. En cambio, los valores más bajos para la clase 1 muestran que el modelo tiene más dificultades para detectar correctamente los nudos, lo que sugiere que hay un ligero desbalance o mayor complejidad al clasificar esta clase. Un *alpha* pequeño (0.0001) ha permitido mantener una buena capacidad de generalización, especialmente en la clase mayoritaria, la de las larvas.

Un *alpha* mayor habría reducido el riesgo de sobreajuste, pero a cambio habría reducido la capacidad predictiva del algoritmo y habría disminuido el rendimiento global del modelo.

En cuanto a la curva de aprendizaje, se obtiene la gráfica representada en la Fig. 4.52.

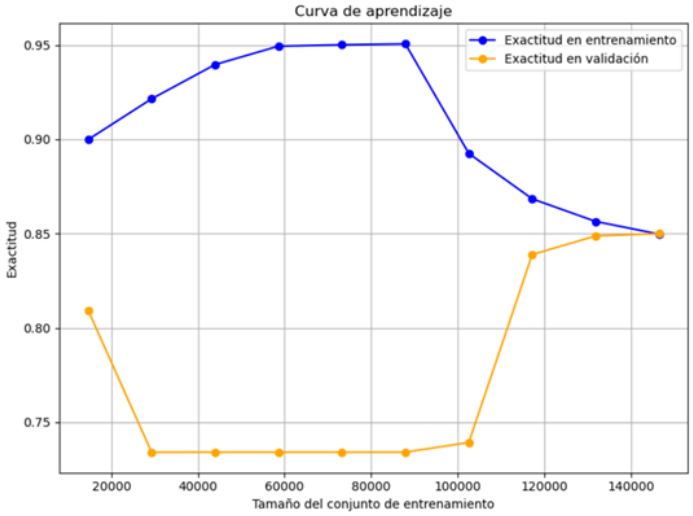


Figura 4.52: Curvas de aprendizaje para el clasificador 'Nudo VS Larva' SVM con CV = 100.

En la Fig. 4.52 se puede apreciar que, al ir incrementando el tamaño del conjunto de entrenamiento, la exactitud del entrenamiento aumenta hasta estabilizarse, al contrario de lo que sucede con la curva de exactitud de la validación. Sin embargo, a partir de cierto número de muestras para el entrenamiento, la exactitud de la validación incrementa muy bruscamente, a diferencia de la curva de exactitud del entrenamiento. Ambas terminan convergiendo en una exactitud de 0,85.

Este comportamiento puede justificarse mediante el uso de una validación cruzada con un número elevado de folds, lo que reduce el tamaño de los subconjuntos empleados durante el entrenamiento y genera una mayor varianza en los resultados. Además, el empleo de un valor tan bajo de *alpha* (*alpha* = 0,0001) permite que el modelo se ajuste con mucha libertad, apareciendo cierto sobreajuste cuando el tamaño del conjunto de entrenamiento es reducido, lo cual produce una disminución de la exactitud en la validación.

El hecho de que, conforme se incrementa dicho conjunto, ambas curvas converjan, indica que el modelo mejora su capacidad de generalización y deja de sobreajustarse.

4.6.0.4. Conclusiones para discriminación entre nudo y larva

De los resultados de los tres modelos implementados se deduce que es KNN el que mejores resultados proporciona, y que es el que muestra el mejor equilibrio entre rendimiento y robustez, puesto que la exactitud que alcanza es del 89 %. A pesar de que la clasificación que realiza es muy robusta y de que es la más fiable de los tres casos implementados, al haber desbalance entre el número de

muestras de las distintas clases, presenta más dificultades al clasificar los nudos correctamente.

En cuanto al modelo Random Forest, no se ha conseguido que el modelo sea capaz de generalizar de manera efectiva, y el sobreajuste que presenta sigue siendo muy notorio.

El modelo SVM implementado, con la variación del SGDClassifier, ha resultado ser muy útil a la hora de eliminar las características irrelevantes, puesto que, mediante la regularización, penaliza los coeficientes grandes y tiende a reducir a cero los pesos de las características que no aportan valor al modelo.

Capítulo 5

Conclusiones y líneas futuras

5.1. Conclusiones

En este trabajo se ha desarrollado un método para clasificar sonidos generados por insectos durante su actividad xilófaga en vigas de madera, utilizando características acústicas extraídas mediante esquemas implementados en LabVIEW. La selección de las características a extraer y su posterior implementación resulta fundamental, ya que trabajar con características derivadas permite reducir la complejidad del análisis y enfocar el modelo en los aspectos más representativos del sonido, evitando el ruido y la alta dimensionalidad que conlleva tratar directamente con la señal acústica original. Además, la obtención de los valores de estas características ha permitido evaluar su relevancia y la cantidad de información que pueden proporcionar al clasificador, mejorando la calidad de los datos utilizados para el entrenamiento. Para ello, la implementación de algoritmos en LabVIEW que facilitaron la extracción precisa y sistemática de estas características acústicas ha sido clave.

Se compararon diferentes algoritmos de clasificación, Random Forest, K-Nearest Neighbors y Support Vector Machine, aplicados tanto sobre datos normalizados como estandarizados, evaluando su rendimiento con y sin validación cruzada. Los resultados indican que la estandarización de los datos como método de escalado, en la gran mayoría de los casos, proporciona valores más altos en las distintas métricas evaluadas que la normalización, ya que mantiene la distribución de las características y favorece la convergencia de los modelos. En cuanto a los algoritmos, Random Forest muestra mayor robustez frente a variaciones en los datos, K-Nearest Neighbors destaca por su simplicidad y capacidad de genralización, y Support Vector Machines presenta la mejor capacidad para separar clases complejas mediante la optimización de hiperparámetros, así como para generalizar. Además, se realizó un análisis específico para distinguir entre nudos y larvas, junto con la generación de curvas de aprendizaje para evaluar la estabilidad y generalización de los modelos.

Los resultados muestran que, aunque todos los métodos alcanzan un rendimiento aceptable, la combinación de optimización de hiperparámetros y validación cruzada resulta fundamental para mejorar la capacidad de generalización del modelo, ya que esta última permite evaluar el rendimiento en distintos subconjuntos de los datos, evitando el sobreajuste y asegurando que el modelo aprenda patrones generalizables. Esto se refleja claramente en las curvas de aprendizaje, donde al incrementar el tamaño del conjunto de entrenamiento se observa, como norma general, una convergencia progresiva entre la exactitud en entrenamiento y validación, indicando un modelo más estable y con mejor capacidad para predecir nuevos datos, evitando que el modelo simplemente memorice los datos de entrenamiento sin aprender patrones enfocados a la generalización.

De los resultados reseñables obtenidos, se destacan los siguientes:

- A lo largo de las ejecuciones de los distintos algoritmos implementados, habiendo normalizado los datos y sin aplicar validación cruzada, se observa que, a pesar de que el modelo Random Forest presenta los mejores valores de AUC y exactitud, también es el que muestra un mayor grado de sobreajuste, siendo este mínimo o prácticamente inexistente en K-Nearest Neighbors y Support Vector Machine. Esto sugiere que Random Forest tiene una mayor capacidad para aprender los patrones presentes en los datos de entrenamiento, lo que puede comprometer su capacidad de generalización frente a nuevos datos.

- A lo largo de las ejecuciones que implementan validación cruzada, tanto sobre datos normalizados como estandarizados, se observa en las curvas de aprendizaje una clara convergencia entre los errores de entrenamiento y validación en los modelos K-Nearest Neighbors y Support Vector Machine, lo cual indica una buena capacidad de generalización. En cambio, el modelo Random Forest no muestra esta convergencia, evidenciando una mayor tendencia al sobreajuste. A pesar de que Random Forest alcanza valores de AUC y exactitud iguales o incluso superiores a los de los otros modelos, su menor capacidad para generalizar sugiere que está aprendiendo patrones específicos del conjunto de entrenamiento en lugar de abstraer reglas útiles para nuevos datos.

- Para el caso de implementación de los algoritmos para la discriminación entre nudo y larva, destaca la AUC de 0,96 que ofrece el algoritmo Random Forest, puesto que es de los tres el que mayor valor para dicho parámetro ofrece. Sin embargo, destaca también la convergencia tan brusca que se presenta en el caso de la curva de aprendizaje del modelo Support Vector Machine.

Por otro lado, se puede concluir del análisis general de rendimiento que ciertas clases presentan mejor comportamiento que otras. Esto se relaciona directamente con la posición física de los nudos en las vigas y con las posiciones en las que se han introducido las larvas en ellas, además de la trayectoria que han seguido

posteriormente al morder la madera. A su vez, se relaciona con el número de detecciones de cada larva del que disponemos, puesto que no es equitativo para todas las clases.

Las clases que representan a la Larva 6 y al Nudo 2 han sido clasificadas con una alta precisión. Esto se debe a que ambas clases son de las que más representación en cuanto a Support presentan, siendo este de 4044 y 6256 respectivamente. Además, no se encontraban posicionadas en la estructura de vigas en posiciones fácilmente solapables con otras larvas.

No ocurre lo mismo en el caso de las larvas 3 y 5. Estas clases presentan un mayor grado de confusión, especialmente entre ellas. Se dispone de 3554 capturas para la larva 3, pero de la larva 5 solo existen 505, lo cual es un impedimento a la hora de entrenar la capacidad del modelo para generalizar respecto a dicha clase. Además, por las posiciones físicas en las que se han introducido estas dos larvas, y sabiendo que la trayectoria que ambas han seguido las hace estar más próximas entre sí, se puede deducir que parte de las señales que proceden de ellas han convergido acústicamente en el espacio entre ambas vigas, lo que complica su separación por parte de los modelos.

Esta hipótesis se ve reforzada por el análisis de los errores de clasificación y las curvas de aprendizaje, donde modelos como Support Vector Machine y K-Nearest Neighbors muestran dificultades puntuales al discriminar entre estas clases a pesar de su buena capacidad de generalización global.

Se ha observado que algunas confusiones con los nudos podrían deberse a su proximidad con otras fuentes activas, ya que el sonido se propaga entre vigas cercanas. Además, las fibras longitudinales de la madera favorecen la transmisión del ruido, lo que puede provocar que el sonido generado por una larva se detecte antes en otro punto de la viga, distinto al lugar transversal donde realmente se encuentra. Esto dificulta identificar con precisión el origen del sonido y sugiere que la ubicación influye de forma significativa en la clasificación.

5.2. Líneas futuras

Este estudio proporciona una base útil para seguir avanzando en la clasificación acústica de insectos xilófagos, mostrando que, con un trabajo efectivo de extracción de características y una selección de los algoritmos, es posible identificar su actividad de forma efectiva. Aunque el conjunto de datos utilizado es limitado, los resultados indican que técnicas como la validación cruzada y la optimización de hiperparámetros marcan una diferencia clara en la capacidad del modelo para generalizar.

A partir de aquí, sería interesante continuar explorando con conjuntos de datos más amplios y variados, probar nuevas formas de representar la información acústica o incluso combinar distintos modelos para aprovechar lo mejor de cada uno. También sería interesante implementar técnicas de balanceo de datos para

entrenar y validar con un conjunto más equitativo en cuanto a las clases que se pretenden detectar. Todo esto podría ayudar a construir sistemas más precisos y robustos que puedan aplicarse en escenarios reales de detección temprana y control de plagas en estructuras de madera, con un enfoque más práctico y automatizado.

5.3. Objetivos de Desarrollo Sostenible

Los Objetivos de Desarrollo Sostenible (ODS) constituyen un conjunto de 17 metas globales, mostradas en la Fig. 5.1, adoptadas en 2015 por los Estados miembros de la ONU como parte de la Agenda 2030 para el Desarrollo Sostenible.

Esta iniciativa representa una oportunidad para que los países y sus sociedades emprendan un nuevo camino hacia la mejora de la calidad de vida de todas las personas, fomentando el crecimiento económico y abordando necesidades sociales como la educación, la sanidad, la protección social y la conservación del medio ambiente, entre otras.

Figura 5.1: Objetivos de Desarrollo Sostenible.

Este Trabajo de Fin de Grado contribuye a ciertos objetivos, sirviendo como bien social en los siguientes aspectos:

- ODS 9: Industria, innovación e infraestructura. A través de este trabajo se promueve la aplicación de nuevas tecnologías al ámbito del control biológico y de plagas. Esto favorece la creación de dinámicas de control sostenibles que pueden incorporarse a sistemas e infraestructuras comprometidas con el medio ambiente.
- ODS 11: Ciudades y comunidades sostenibles. Mediante métodos para la detección temprana y no invasiva de insectos xilófagos, se contribuye a preservar el patrimonio arquitectónico y a mejorar la seguridad urbanística.
- ODS 12: Producción y consumo responsables. La capacidad de monitoreo de insectos xilófagos permite reducir radicalemnte el uso de pesticidas,

minimizando el impacto ambiental.

- ODS 15:Vida de ecosistemas terrestres. Gracias al uso de tecnologías no invasivas de detección de insectos xilófagos se impulsa la acción temprana sobre plagas que amenazan la salud de diversos ecosistemas. De esta manera se contribuye a conservar dichos hábitats y a apoyar una gestión forestal responsable que respete los ciclos de los ecosistemas.

Bibliografía

[1] Roberto Diego Martínez. *Acoustic detection and localisation system for Hylotrupes bajulus L. larvae using a MEMS microphone array.* DOI: APAC-D-23-00403R3.

[2] *National Instruments*. Accessed: 2024. URL: https://www.ni.com/es.html.

[3] *Python*. Accessed: 2024. URL: https://www.python.org/.

[4] *Python: ¿qué es y para qué sirve?* Accessed: 2024. URL: https://www.arsys.es/blog/python-que-es-y-para-que-sirve#tree-3.

[5] *Libreria scikit-learn python, aprende todo*. Accessed: 2024. URL: https://www.tokioschool.com/noticias/que-es-scikit-learn/.

[6] Fabián Aguirre Martín. «Desarrollo y análisis de clasificadores de señales de audio». Máster en ingeniería acústica. Gandía, España: Universidad politécnica de Valencia, 2017.

[7] *Métricas de análisis de vibraciones: Curtosis y Asimetría (Skewness)*. Accessed: 2024. URL: https://dynamox.net/es/blog/metricas-de-analisis-de-vibraciones-curtosis-y-asimetria-skewness.

[8] Alberto García García. «Análisis del uso de algoritmos de Deep Learning en un Sistema acústico de detección de larvas de insectos xilófagos». TFG grado en ingeniería en tecnologías específicas de telecomunicación. Valladolid, España: Universidad de Valladolid, 2024.

[9] Roberto Díaz. *The machine Learners*. URL: https://www.themachinelearners.com/metricas-de-clasificacion/ (visitado 24-10-2024).

[10] *Cómo preparar un conjunto de datos para machine learning y análisis*. Accessed: 2024. datos.gob.es. 2023. URL: https://datos.gob.es/es/blog/como-preparar-un-conjunto-de-datos-para-machine-learning-y-analisis.

[11] Richard O. Duda, Peter E. Hart y David G. Stork. *Pattern Classification*. 2nd. New York: Wiley-Interscience, 2001. ISBN: 978-0-471-05669-0.

[12] Gustavo A. Betancourt. «Las máquinas de soporte vectorial (SVMs)». En: *Scientia et Technica* Año XI.No 27 (abr. de 2005). ISSN: 0122-1701.

[13] *Cross-Validation: definición e importancia en Machine Learning*. Accessed: 2024. URL: https://datascientest.com/es/cross-validation-definicion-e-importancia.

[14] J Pujol J. Minguillón. «Árboles de decisión». En: (), pág. 7.

[15] *Scikit-learn: Descenso de Gradiente Estocástico (SGD)*. Accessed: 2025-05-26. URL: https://areatutorial.com/scikit-learn-descenso-de-gradiente-estocastico/.

[16] Towards Data Science. *Regularization in Machine Learning: L1, L2 and Elastic Net*. 2019. URL: https://towardsdatascience.com/regularization-in-machine-learning-76441ddcf99a.

[17] scikit-learn developers. *1.1. Linear Models — scikit-learn 1.6.1 documentation*. 2024. URL: https://scikit-learn.org/stable/modules/linear_model.html.