PROYECTO FIN DE GRADO

TÍTULO: USO DE REDES NEURONALES PARA CLASIFICACIÓN DE IMÁGENES HIPERESPECTRALES EN TEJIDOS BIOLÓGICOS

AUTOR/A: ANDRADA DIMACHE

TITULACIÓN: TELEMÁTICA

DIRECTOR/A: ALEJANDRO MARTÍNEZ DE TERNERO RUIZ

TUTOR/A: EDUARDO JUÁREZ MARTÍNEZ

DEPARTAMENTO: INGENIERÍA TELEMÁTICA Y ELECTRÓNICA

VºBº TUTOR/A

Miembros del Tribunal Calificador:

PRESIDENTE/A: ALEJANDRO GARCÍA LAMPÉREZ

TUTOR/A: EDUARDO JUÁREZ MARTÍNEZ

SECRETARIO/A: MIGUEL CHAVARRÍAS LAPASTORA

Fecha de lectura:

Calificación:

El Secretario/La Secretaria,

Resumen

Uso de redes neuronales para clasificación de imágenes hiperespectrales en tejidos biológicos.

El propósito general del proyecto es mejorar la precisión de la localización y delimitación del tejido cerebral, abordando la problemática del desplazamiento cerebral. Este fenómeno puede conllevar a posibles errores de precisión de las cirugías guiadas por imágenes. El uso de imágenes hiperespectrales junto con algoritmos de aprendizaje automático presenta una solución innovadora para detectar estos desplazamientos y permite una diferenciación precisa de los tejidos de forma no invasiva y más económica en comparación con tecnologías tradicionales.

La principal limitación tecnológica del proyecto consiste en la escasa disponibilidad computacional para el procesamiento de los datos y el entrenamiento de los modelos de aprendizaje automático. Desde una perspectiva económica, la implementación de esta tecnología promete reducir costes a largo plazo al reducir la necesidad de equipos caros y procedimientos prolongados. Desde un punto de vista ambiental, se trata de un enfoque beneficioso, ya que minimiza el uso de materiales y recursos en comparación con métodos más invasivos.

La metodología utilizada en el proyecto incluye el desarrollo de un sistema integrado que permite procesar datos, entrenar modelos, predecir y generar un informe. Se han probado diversos modelos de clasificación : máquinas de vectores de soporte (SVM), bosques aleatorios y diferentes arquitecturas de redes neuronales. Cada modelo se ha evaluado mediante métricas como precisión, recuperación o área bajo la curva ROC, buscando un equilibrio entre la detección de falsos positivos y exclusión de verdaderos positivos.

Los resultados indican que los modelos presentan márgenes de mejora. Se propone una búsqueda de hiperparámetros más exhaustiva, un aumento de datos y la implementación de métodos de aprendizaje en conjunto.

En general, el proyecto tiene un impacto positivo en el sector de la medicina al promover varios Objetivos de Desarrollo Sostenible (ODS), como la salud y el bienestar, la reducción de desigualdades, la producción y el consumo responsable o la alianza para alcanzar objetivos.

Abstract

Use of neural networks for hyperespectral image classification in biological tissues.

The general purpose of the project is to improve the accuracy of the localization and delimitation of brain tissue, addressing the issue of brain shift. This phenomenon can lead to possible precision errors in image-guided surgeries. The use of hyperspectral imaging together with machine learning algorithms presents an innovative solution to detect these shifts and allows for precise tissue differentiation in a non-invasive and more economical manner compared to traditional technologies.

The main technological limitation of the project is the limited computational availability for data processing and machine learning model training. From an economic perspective, the implementation of this technology promises to reduce long-term costs by decreasing the need for expensive equipment and lengthy procedures. From an environmental point of view, it is a beneficial approach as it minimizes the use of materials and resources compared to more invasive methods.

The methodology used in the project includes the development of an integrated system that allows for data processing, model training, prediction and report generation. Various classification models have been tested: support vector machines (SVM), random forests and different neural network architectures. Each model has been evaluated using metrics such as accuracy, recall or area under the ROC curve, seeking a balance between the detection of false-positive and true-positive exclusion.

The results indicate that the models present room for improvement. A more exhaustive hyperparameter search, data augmentation, and the implementation of ensemble learning methods are proposed.

Overall, the project has a positive impact on the medical sector by promoting several Sustainable Development Goals (SDGs), such as health and well-being, reduction of inequalities, responsible production and consumption, or alliance to achieve goals.

Índice de figuras

Índice de tablas

Lista de Acrónimos

Adam Adaptative Moment Estimation. v, vii, 21, 41–44

AUC Area Under the Curve. vi, 16, 35, 36, 39, 45–50, 55, 65, 67, 69, 71, 72

CEL Cross Entropy Loss. v, vii, 42–44

FN False Negative. 15

FP False Positive. 15

FPR False Positive Ratio. 16, 46

GB Gygabite. 26

GELU Gaussian Error Linear Unit. 19, 33

HSI Hyperespectral image. v, 7, 8, 34

iMRI Resonancia magnética intraoperatoria. 1

iUS Ultrasonido intraoperatorio. 1

RAM Random Access Memory. 26

RBF Radial Base Function. 12

ReLU Rectifier Linear Unit. 18

RGB Red Green Blue. v, 7, 8, 27, 37, 47, 72

ROC Receiver Operating Characteristic. i, iii, v, vi, 16, 17, 34–36, 39, 45–50, 55, 65, 67, 69, 71, 72

SGD Stochastic Gradient Descent. v, vii, 20, 21, 42–45

SVM Support Vector Machine. i, iii, vii, 11, 25, 30, 32, 35, 39, 40, 45–50, 55

TN True Negative. 15

TP True Positive. 15

TPR True Positive Ratio. 16, 46

Índice de contenidos

Capítulo 1

Introducción

1.1. Marco y motivación del proyecto

En el ámbito de la neurocirugía, la precisión en la localización y delimitación de tejidos cerebrales se presenta como uno de los desafíos más significativos. Este es el caso del fenómeno denominado desplazamiento cerebral (*brain shift*), en el que el cerebro se puede mover ligeramente por el cambio de presión y las pérdidas de líquido cefalorraquídeo provocando deformaciones. Además, los procedimientos quirúrgicos de extirpación o cortes, provocan estos desplazamientos ocasionando posibles errores de precisión si no se detecta. El desplazamiento cerebral durante una intervención, puede llegar a invalidar la correspondencia entre las imágenes preoperatorias y la anatomía real del paciente [1], comprometiendo la precisión de la cirugía guiada por imágenes. Este fenómeno requiere de métodos sofisticados como la resonancia magnética intraoperatoria (iMRI) y el ultrasonido intraoperatorio (iUS). Estos métodos son costosos en material y tiempo, y limitan el área de la operación.

La implementación de imágenes hiperespectrales y algoritmos de aprendizaje automático proporciona una solución innovadora para detectar y corregir estos desplazamientos [2], [3]. Se trata de una tecnología emergente en el ámbito médico que aún no ha llegado a su máximo potencial, aún así, ofrece una serie de ventajas. La tecnología de imágenes hiperespectrales es capaz de diferenciar los distintos tipos de tejido [4] ayudando a la detección de los desplazamientos, de manera no ionizante y sin contacto [5], [6], reduciendo riesgos asociados a otras técnicas. Esta tecnología es más económica y menos intrusiva que la iMRI, haciéndola más accesible y práctica en diversos entornos clínicos [7].

Las imágenes hiperespectrales capturan información de múltiples bandas del espectro electromagnético, en el que se adquiere mayor nivel de detalle comparado con las imágenes convencionales que solamente capturan información de tres bandas (rojo, verde y azul). En el contexto del desplazamiento cerebral, mediante esta tecnología se detectan variaciones en los tejidos cerebrales no visibles con otras técnicas de imagen [8].

Esto es así debido a que la imagen hiperespectral proporciona las firmas espectrales de los materiales que están siendo observados. Estas firmas pueden contener patrones muy sutiles que son muy complejos de apreciar e interpretar a simple vista. Se observa mejor este problema en la figura 1.1, que muestra un mapa de calor (*heatmap*) de las medias de bandas por cada clase existente en nuestros datos.

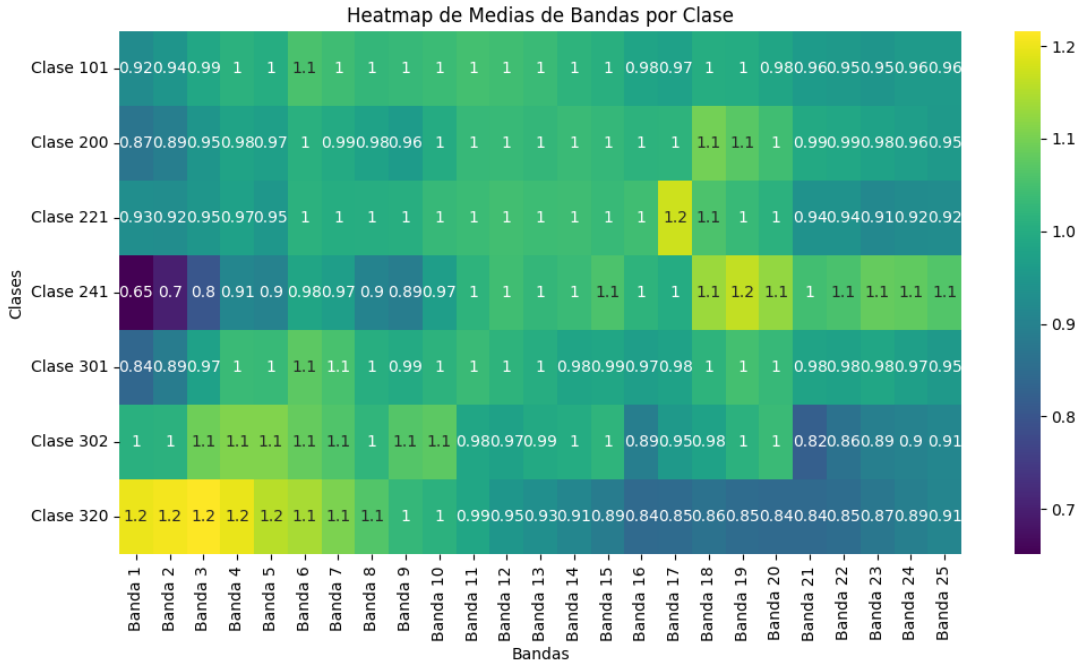


Figura 1.1. Mapa de calor de medias de bandas por clase

En la figura 1.1, cada fila se corresponde con una clase específica de nuestros datos y cada columna se corresponde con una banda espectral. Los valores contenidos en las celdas indican la media de la respuesta espectral para cada combinación de clase y banda. Aunque los colores en el mapa de calor permiten identificar algunas diferencias y similitudes entre firmas espectrales de las diversas clases, no es posible detectar todas las variaciones a simple vista. Se observa que existe variabilidad en las firmas espectrales dentro de algunas clases, esto puede dificultar la clasificación puesto que una misma clase no tiene una firma espectral uniforme. Diferentes clases tienen respuestas similares en algunas bandas espectrales, en concreto, en la figura 1.1 las bandas 10 a 15, varias clases tienen valores muy cercanos a uno. Por este motivo son necesarios los algoritmos de aprendizaje automático, porque tienen la capacidad de trabajar con grandes cantidades de datos y encontrar patrones que los seres humanos no pueden detectar.

En este Proyecto Fin de Grado, se utilizan los algoritmos de aprendizaje automático convencionales, como la máquina de vectores de soporte y los bosques aleatorios, y se comparan con modelos de redes neuronales artificiales en el ámbito médico. Se busca delimitar las distintas zonas de tejido utilizando estos algoritmos, proporcionando una evaluación comparativa de cuáles de estos algoritmos se adaptan mejor a este problema específico.

Las máquinas de vectores de soporte son conocidas por su eficacia en problemas de clasificación y su capacidad de manejar conjuntos de datos de alta dimensionalidad. Por otra parte, los bosques aleatorios son conocidos por su robustez y precisión en la clasificación de datos complejos. Los modelos de redes neuronales artificiales son prometedores en aplicaciones médicas debido a su capacidad para aprender características complejas y no lineales de los datos.

Este proyecto pretende determinar cuál de estos algoritmos proporciona la mejor delimitación de las zonas de tejido en términos de precisión, eficiencia y robustez. La precisión se refiere a la capacidad para identificar correctamente las zonas de tejido. Con eficiencia, por su parte, hace referencia al rendimiento computacional del algoritmo, incluyendo aspectos como el tiempo de entrenamiento. Finalmente, la robustez se refiere a la capacidad del algoritmo para mantener su rendimiento ante variaciones en los datos, como datos no vistos durante el entrenamiento.

Mediante la comparación de cómo de bien delimita los tejidos cerebrales cada modelo, se pretende identificar las fortalezas y debilidades de cada uno de ellos, proporcionando una guía clara sobre cuál algoritmo es más adecuado para este problema.

1.2. Objetivos técnicos y académicos

Esta sección se centra en resumir los objetivos de este proyecto fin de grado. Los objetivos desde el punto de vista técnico son:

- Análisis de algoritmos convencionales de aprendizaje automático para clasificación de imágenes hiperespectrales.
- Proposición y análisis de un algoritmo basado en redes neuronales que permita la clasificación del tejido cerebral en cuatro clases y delimitando los diferentes tipos de tejidos.
- Estudio y aplicación de diversas métricas de calidad con el fin de determinar la precisión del algoritmo propuesto.

Desde el punto de vista académico, el proyectista adquiere las siguientes competencias y habilidades:

- Análisis y procesamiento de imágenes hiperespectrales.
- Evaluación y comparación de algoritmos de clasificación, aplicación de métricas de evaluación para interpretar y validar modelos.
- Mejora de habilidades de programación en el lenguaje de programación de *Python*, utilización de bibliotecas como *scikit-learn* y *frameworks* como *Pytorch* empleados en el aprendizaje automático.

1.3. Estructura del resto de la memoria

Los próximos capítulos de la memoria componen los siguientes apartados:

- Capítulo 2: Marco tecnológico. Se realiza un análisis del marco teórico en el que está enfocado el proyecto, a grandes rasgos, las imágenes hiperespectrales y el aprendizaje automático.
- Capítulo 3: Especificaciones y restricciones de diseño. En base a los objetivos planteados en la introducción, se realizan unas especificaciones y restricciones a nivel de hardware y de software.
- Capítulo 4: Descripción de la solución. Se expone la herramienta desarrollada. Descripción de los componentes que integran la herramienta y el flujo de trabajo desde la carga de imágenes hiperespectrales hasta la obtención de los resultados.
- Capítulo 5: Resultados. Se exponen los resultados obtenidos durante los experimentos realizados. Se presenta la lista de parámetros probados en la búsqueda de hiperparámetros, los modelos utilizados en los experimentos y las métricas de evaluación de los modelos. Se realiza la comparación de los modelos probados y las métricas de evaluación de la predicción. Se incluyen tablas y gráficos que ilustran los resultados.
- Capítulo 6: Conclusiones. Se discute la eficacia de la herramienta en comparación con las expectativas iniciales. Se proponen posibles mejoras y trabajos futuros, para ampliar y optimizar los resultados obtenidos.
- Capítulo 7: Presupuesto. Se incluyen los gastos asociados a los recursos humanos, hardware y software.
- Capítulo 8: Impacto del proyecto. Se analiza el impacto desde perspectivas como impacto social, económico y ambiental.

Capítulo 2

Marco Tecnológico

En este capítulo, se aborda el marco tecnológico que conforma este proyecto. En primera instancia, se realiza una introducción a las imágenes hiperespectrales, que serán la fuente de los datos utilizados. Se realiza una breve introducción a los conceptos básicos y fundamentales, para después profundizar en las características en las que se basan los datos.

A continuación, se proporciona una explicación detallada del campo del aprendizaje automático y de los tipos de algoritmos empleados en este proyecto, en concreto, las máquinas de vectores de soporte y los bosques aleatorios. Además, se proporciona un esbozo hacia los métodos existentes en cuanto a la búsqueda de mejores parámetros para estos algoritmos.

En penúltimo lugar, se concluye el capítulo proporcionando una base conceptual sobre las redes neuronales artificiales. Finalmente, se hará una necesaria introducción hacia las métricas de evaluación usadas en el proyecto a la hora de comparar diferentes modelos.

2.1. Imágenes hiperespectrales

No se puede abordar la temática de imágenes hiperespectrales sin mencionar antes los conceptos de luz, imagen y espectro [9]. La luz es una onda inmaterial capaz de viajar a través del vacío sin ayuda de un medio, está compuesta por campos eléctricos y magnéticos que oscilan perpendicularmente. Dada la dualidad onda-corpúsculo de la luz,cuando se comporta como una partícula, se manifiesta en fotones que pueden interactuar con la materia de manera similar a las partículas. Sin embargo, también puede comportarse como una onda, caracterizándose por una velocidad, una longitud de onda y una frecuencia.

Una imagen es la reproducción de un objeto mediante la combinación de los rayos de luz que proceden de dicho objeto. El espectro, por su parte, es el conjunto de intensidades de la luz en diferentes longitudes de onda. Por lo que, una imagen espectral es una reproducción de un objeto en función de la longitud de onda que refleja o emite.

2.1.1. Fundamentos

La radiación electromagnética se presenta como una onda que se autopropaga en el vacío o en la materia. El espectro electromagnético está formado por diversas regiones con diferentes energías tal y como se observa en la figura 2.1. Cuando una onda electromagnética incide sobre

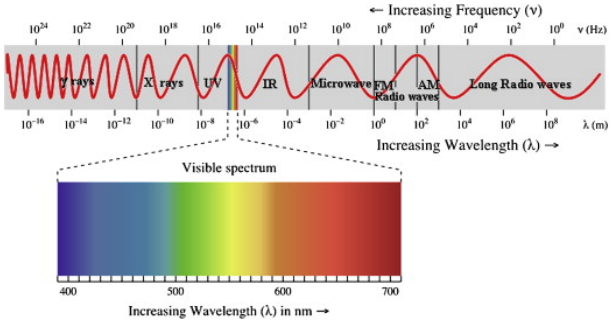


Figura 2.1. Espectro electromagnético [10].

una muestra, es decir, se ilumina, la radiación del sistema de iluminación se transmite, refleja o se absorbe. Mediante la recopilación y procesamiento de la información a lo largo del espectro electromagnético, es decir, división de la imagen en bandas del espectro visible y más allá, se forman las imágenes hiperespectrales.

Las imágenes hiperespectrales tienen su origen en la detección remota. Pertenecen al campo de la espectroscopia, ciencia que estudia la interacción de la radiación electromagnética con la materia. Permiten capturar y procesar diferentes longitudes de onda, representando información física y química del objeto observado.

Se utilizan en diversos campos como:

- La medicina, para la detección de carcinoma en ratones [11] o tumor de mama en ratas [12].
- Sector alimentario [13], para la evaluación de la calidad y seguridad de los alimentos [14] durante diferentes procesos como cocción, secado, congelación o fermentación [15] o la determinación de la distribución de agua en la carne de res durante la deshidratación [16].
- Sector automotriz, mediante la distinción de obstáculos potenciales al fondo de la carretera [17].
- La agricultura, para detectar rasgos abióticos, bióticos y de calidad en plantas en condiciones de cultivo [18].
- Sector ganadero y avícola, mediante la detección del deterioro microbiano en el sector cárnico, detección de adulterantes en leche en polvo y predicción de contenido de grasa en leche, y evaluación de tumores mamarios en perros [19].

En el contexto de este proyecto, las imágenes hiperespectrales (HSI por sus siglas en inglés) se utilizan en el campo de la medicina [20], específicamente para la clasificación de tejido cerebral y delimitación de tejido patológico.

2.1.2. Firma espectral, cubo hiperespectral y verdad terreno

La diferencia de composición química y estructura física que provoca la reflexión, absorbancia, transmitancia y/o emisión de energía electromagnética de un material en patrones diferenciables en longitudes de ondas específicas se denomina firma espectral [21]. Las firmas espectrales son utilizadas para caracterizar e identificar características únicas en imágenes. En el caso de las imágenes de tejido cerebral expuesto, estas firmas pueden revelar diferencias en la composición y estructura del tejido permitiendo identificar áreas afectadas por tumor, áreas sanas o áreas que constituyen venas o arterias.

Los datos provenientes de las imágenes son bidimensionales (alto, ancho), al agregar una nueva dimensión con información espectral, los datos se percibirán como un cubo de datos en tres dimensiones (alto, ancho, banda espectral) denominado cubo hiperespectral. Es por ello, que este tipo de imágenes nos proporciona tanto información espectral como espacial. Cada píxel de la imagen se caracteriza por una curva espectral que puede abarcar desde la región ultravioleta hasta la infrarroja [22]. Es posible visualizar estos conceptos en la siguientes figuras 2.2 y 2.3, en la primera observamos la diferencia entre un cubo HSI con una imagen RGB y en la segunda podemos observar una representación del cubo HSI, una banda espectral del cubo y la firma espectral de un píxel del cubo HSI .

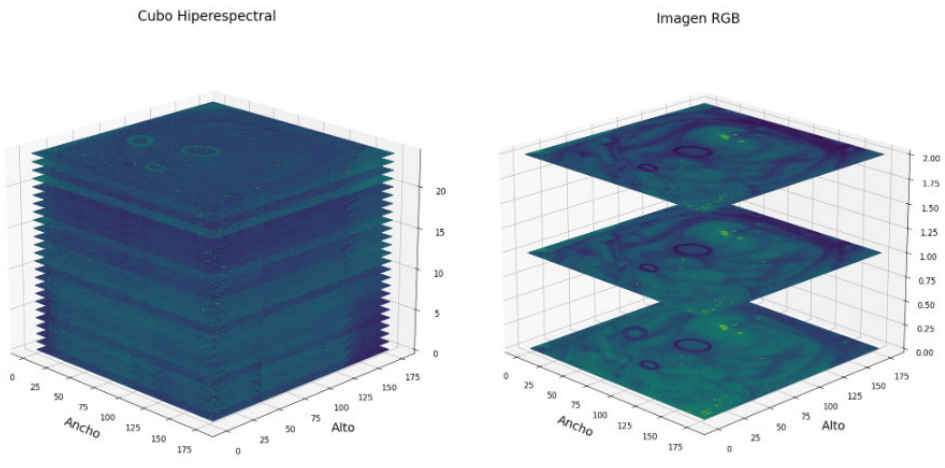


Figura 2.2. Diferencia entre un cubo HSI con una imagen RGB.

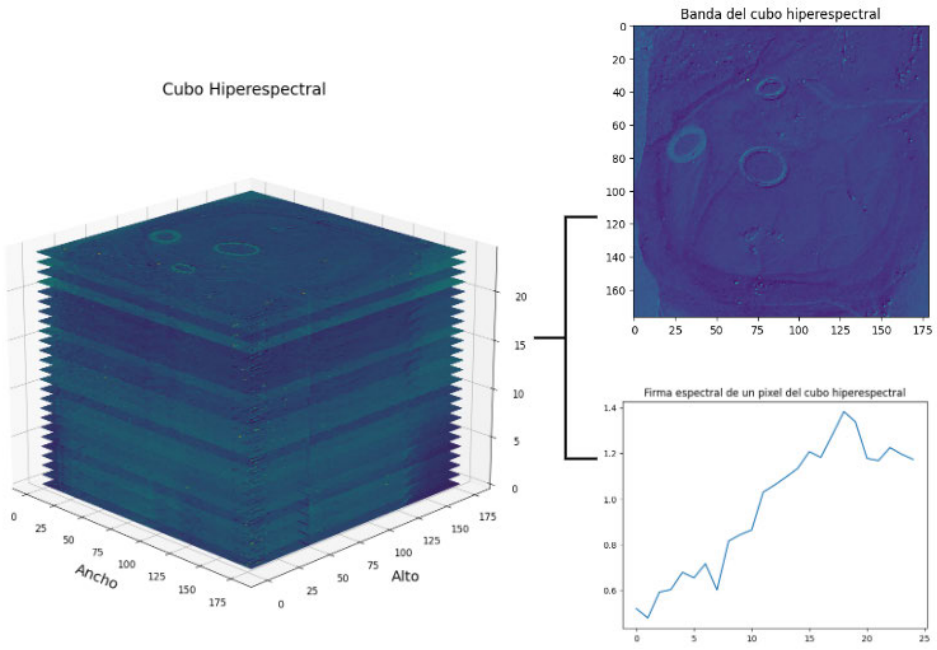


Figura 2.3. Representación del cubo HSI, banda espectral y firma espectral.

Por otro lado, la verdad terreno o *ground truth* describe los datos reales clasificados por un neurocirujano. En el marco en el que se desarrolla este proyecto, es fundamental obtener los datos etiquetados, lo que implica la validación de los datos por médicos que son capaces de identificar de manera precisa las características relevantes en las imágenes. Para visualizar mejor este concepto, se representa en la figura 2.4 la imagen RGB de la operación junto con el mapa de verdad terreno que muestra un conjunto de píxeles etiquetados por un neurocirujano. La leyenda indica las etiquetas correspondientes a cada color, y se muestra la superposición de ambas imágenes con el objetivo de visualizar que las zonas etiquetadas se corresponden directamente con la captura original.

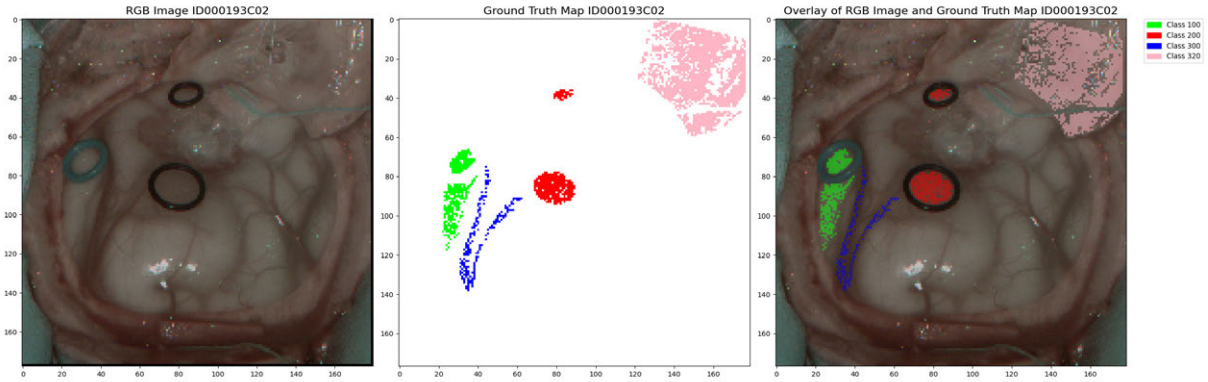


Figura 2.4. Superposición imagen RGB con mapa ground truth.

2.2. Aprendizaje automático

El aprendizaje automático es la ciencia que capacita a los ordenadores a actuar en función de los datos, sin estar programados explícitamente para actuar de una manera determinada. Se basa en un proceso en el que se aprende de los datos para descubrir patrones o predecir eventos. La primera parte del proceso implica el entrenamiento, se procesan los datos para aprender un conjunto de parámetros que nos pueden ayudar a predecir resultados y descubrir patrones en los datos. Durante esta fase, se ajustan los parámetros del modelo mediante iteraciones repetitivas para mejorar su precisión. Una vez entrenado, se procede a la validación, etapa en la que se evalúa el rendimiento del modelo utilizando un conjunto de datos diferente al utilizado en el entrenamiento. De esta manera, se pueden ajustar y optimizar los parámetros del modelo con el objetivo de mejorar su precisión y capacidad de generalización. Después de la validación, se utiliza un conjunto de datos no visto con anterioridad por el modelo, para evaluar el rendimiento final. De esta manera nos aseguramos de que el modelo generaliza bien en datos no vistos. Por último, se procede al despliegue del modelo, etapa en la que el modelo se utiliza para hacer predicciones en nuevos conjuntos de datos. Este proceso se explica de manera más visual en la figura 2.5.

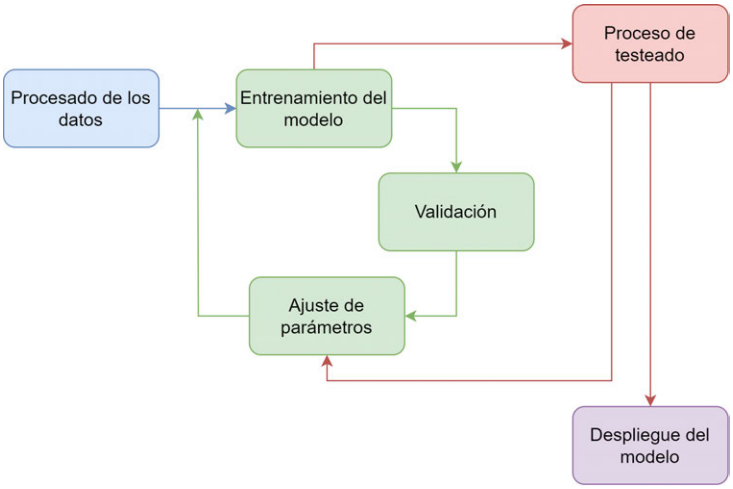


Figura 2.5. Proceso de entrenamiento, validación y prueba

El aprendizaje automático ha adquirido importancia en varios campos de investigación desde el desarrollo de modelos de aprendizaje que pueden detectar cáncer de piel [23] hasta sistemas para predecir necesidades de oxigeno de pacientes con COVID-19, ayudando así a aligerar la asignación de recursos en hospitales [24].

Según el problema al que se enfrente, se pueden encontrar tres subcategorías de aprendizaje automático:

- Aprendizaje no supervisado. Se tratan con datos que no están etiquetados o con datos de

los que se desconoce su estructura.

- Aprendizaje por refuerzo. Se basa en un agente que mejora su desempeño en función de las interacciones que tiene con el entorno, que normalmente incluye una señal de recompensa. El agente intenta maximizar la recompensa a través de interacciones con el entorno.
- Aprendizaje supervisado. Categoría que se alinea con los objetivos de este proyecto donde se utilizan datos etiquetados.

2.2.1. Aprendizaje supervisado

El aprendizaje supervisado o aprendizaje por etiquetas, es el proceso de modelar la relación entre las entradas de datos y las salidas etiquetadas. Este enfoque implica la utilización de un conjunto de datos de entrenamiento, donde cada entrada viene acompañada de una etiqueta correspondiente, con este conjunto de datos se entrena un algoritmo de aprendizaje automático.

El objetivo principal de este tipo de aprendizaje es desarrollar un modelo en base a datos de entrenamiento etiquetados que permita hacer predicciones sobre datos futuros. El proceso viene resumido en la figura 2.6 en la que se muestra la manera de utilizar las características y etiquetas para formar un conjunto de entrenamiento. El conjunto se introduce en un algoritmo de aprendizaje automático que procesa la información y desarrolla un modelo predictivo. El modelo puede recibir nuevas características y predecir etiquetas.

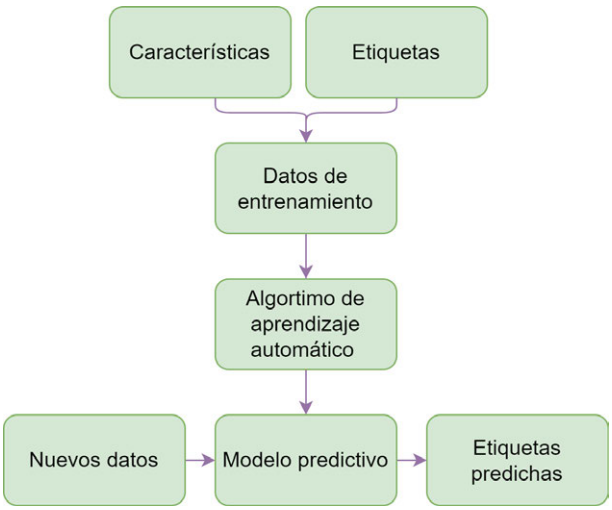


Figura 2.6. Proceso aprendizaje supervisado

La clasificación es una subcategoría del aprendizaje supervisado cuyo objetivo es predecir etiquetas de clase categóricas de nuevos datos basados en observaciones anteriores. Las etiquetas de clase son valores discretos, representan pertenencia a distintos grupos dentro de los datos. El conjunto de etiquetas de clase puede tener naturaleza multiclase como en este proyecto, entonces se trata de un problema de clasificación multiclase.

La regresión constituye otra categoría del aprendizaje supervisado, cuyo objetivo es predecir un número continuo. En este análisis de regresión, se disponen de unas variables predictoras (explicativas) y una variable de respuesta continua (resultado), donde se intenta encontrar una relación entre esas variables para lograr predecir un resultado.

2.2.2. Máquinas de vectores de soporte (SVM)

SVM es un algoritmo de aprendizaje supervisado utilizado en problemas de clasificación y regresión. En este algoritmo, a la hora de entrenar un modelo para clasificar datos, el objetivo es encontrar una frontera o línea que separe las clases de datos, denominado hiperplano. El margen es el espacio entre este hiperplano y los puntos de datos más cercanos de cada clase. Se busca conseguir que el margen sea lo más amplio posible, mejorando así la robustez del modelo y desempeño con nuevos datos nunca vistos. Los puntos de datos que están más cerca del hiperplano, denominados vectores de soporte, determinan cómo de ancho será el margen. Los vectores de soporte influyen en la posición de la línea de separación, una vez que se haya determinado el hiperplano, los demás puntos de datos no afectan al modelo. Cuando se entrena un modelo SVM, el objetivo final es maximizar el ancho del margen haciéndolo lo mas amplio posible y minimizar el error de margen, es decir, evitar que los puntos de datos queden en el lado incorrecto del hiperplano.

En dos dimensiones, el hiperplano es una línea que divide los puntos de datos en una clase positiva y otra negativa. En la figura 2.7 se aprecia los margenes en los dos lados de la línea paralela y equidistante al hiperplano.

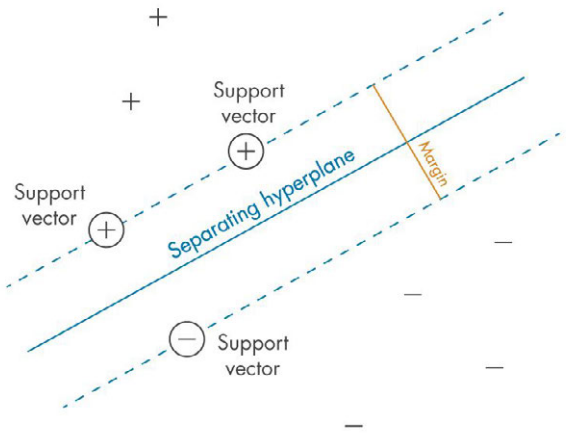


Figura 2.7. Hiperplano de margen máximo [25].

Cabe destacar que SVM permite crear un límite de decisión en datos no separables linealmente. La idea consiste en transformar los datos a una dimensión superior mediante transformaciones no lineales. Se encontrará el hiperplano en nuevas dimensiones en las que se podrán separar los datos en dos clases. Para ello, se hace uso de un método llamado truco de kernel o *kernel trick*. Un

Kernel es una función matemática que encuentra el producto escalar en el espacio transformado. Los cálculos se realizan en dimensiones inferiores y el límite de decisión se crea en dimensiones superiores, truco de kernel. A continuación, se resume los kernels utilizados en este proyecto:

- Lineal. Se utiliza cuando los datos son linealmente separables. Calcula el producto entre dos vectores de entrada.

$$k(x_1, x_2) = x_1 \cdot x_2 \tag{2.1}$$

- Función de base radial (RBF) o función gaussiana. Transforma los datos en un espacio infinito de dimensiones. Mide la distancia entre los vectores y les aplica una función exponencial, donde σ controla cuánto influyen los puntos de datos entre sí.

$$k(x_1, x_2) = \exp\left(-\frac{|x_1 - x_2\|^2}{2\sigma^2}\right) \tag{2.2}$$

2.2.3. Bosques aleatorios (*Random Forest*)

Un árbol de decisión es un método de aprendizaje automático utilizado tanto en problemas de clasificación como regresión. Su funcionamiento se basa en la división de los datos en subconjuntos, dicha división se realiza mediante pruebas condicionales hasta llegar a una predicción. Un árbol de decisión está formado por la raíz, nodo inicial ubicado en la parte superior del árbol; nodos interiores, puntos de decisión donde se aplican pruebas condicionales; y las hojas, nodos finales que contienen la predicción. Cada subconjunto o muestra de datos, pasan por una serie de pruebas condicionales desde la raíz hasta una hoja, en la que se tomará la decisión final.

El proceso de construcción de un árbol es recursivo. En cada etapa se dividen los datos en base a la prueba que mejor separa las clases, si los datos no son suficientes o ya no se pueden dividir, se crea un nodo hoja. Los criterios para decidir cómo dividir los datos son :

- Entropía. Mide la incertidumbre de los datos. Si los datos son todos de la misma clase, la entropía es 0, en cambio, si los datos están igualmente distribuidos la entropía es 1.

$$\text{Entropía} = -\sum_{i=1}^{c} p_i \log(p_i) \tag{2.3}$$

  Donde c es el número de clases y p_i es la proporción de datos pertenecientes a la clase i.

- Índice de Gini. Mide la impureza de una división, minimizando la probabilidad de clasificación incorrecta. La impureza es máxima si las clases están perfectamente mezcladas.

$$\text{Índice de Gini} = 1 - \sum_{i=1}^{c} p(i|t)^2 \tag{2.4}$$

Donde c es el número de clases, $p(i|t)$ es la probabilidad condicionada de que un elemento en el nodo t pertenezca a la clase i.

Mediante la poda, eliminación de ramas que tienen poca importancia, se mejora la eficiencia del árbol. La poda se puede realizar definiendo una profundidad máxima del árbol, minimizando el número de muestras para dividir un nodo o minimizando las muestras en una hoja.

En el contexto del aprendizaje supervisado, un bosque aleatorio optimiza la construcción de árboles de decisión. Un bosque aleatorio es un algoritmo que funciona como un clasificador. Está compuesto por una colección de vectores aleatorios idénticamente distribuidos, es decir, es una combinación de muchos clasificadores de estructura de árbol. La función de cada árbol consiste en emitir un voto unitario por la clase más popular en la entrada [26]. El principio básico es la construcción de múltiples árboles de decisión durante el entrenamiento y la combinación de resultados para optimizar el aprendizaje.

Los bosques aleatorios son muy populares, su uso en clasificación abarca desde la detección temprana de diabetes [27] a la detección de fraudes con tarjetas de crédito [28].

2.2.4. Ajuste de hiperparámetros

A diferencia de los parámetros aprendidos por el modelo durante la fase de entrenamiento como los pesos, los hiperparámetros son parámetros que no se aprenden directamente de los datos, sino que afectan cómo y qué aprende el modelo. Es necesario evaluar el modelo para múltiples valores posibles de hiperparámetros.

Existen diversas técnicas para buscar la mejor combinación de hiperparámetros en un modelo de aprendizaje automático. Entre estas técnicas se incluyen la búsqueda aleatoria (*random search*), los algoritmos genéticos, y la búsqueda de cuadrícula (*grid search*), entre otras. En este proyecto, se utiliza la búsqueda de cuadrícula (*grid search*), un proceso de búsqueda exhaustiva para encontrar la mejor combinación de hiperparámetros evaluando todas las posibles combinaciones en un rango. El proceso comienza definiendo un conjunto de valores posibles para cada hiperparámetro para después entrenar y evaluar el modelo con cada posible combinación. La combinación que proporcione mejor rendimiento, es decir, que maximice las métricas de precisión, exactitud y recuperación en los datos de validación, se seleccionará como mejor combinación. El rendimiento se refiere a cómo de bien el modelo predice correctamente las clases y cómo de bien minimiza los errores, logrando un balance entre todas las métricas que son evaluadas.

2.2.5. Validación cruzada

En el aprendizaje supervisado, se dividen los datos en un conjunto de entrenamiento y un conjunto de prueba. Esta única división puede no ser representativa, incluso llevar a una evaluación sesgada del modelo si el conjunto es pequeño o está distribuido desigualmente. Para solucionar

esta posible problemática, se puede rotar sistemáticamente las partes de entrenamiento y prueba para evaluar como se generalizará el modelo con datos nunca vistos, esto se denomina validación cruzada. La validación cruzada proporciona una mejor evaluación del rendimiento del modelo al utilizar múltiples particiones de datos para el entrenamiento y prueba.

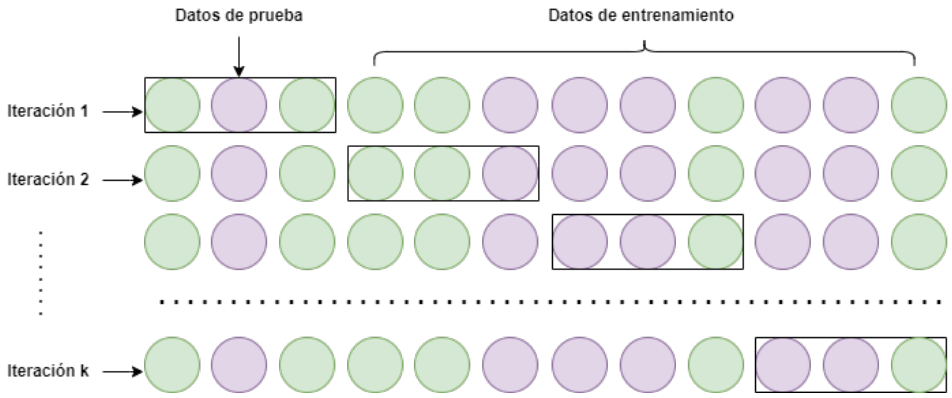


Figura 2.8. Proceso validación cruzada

El proceso comienza dividiendo el conjunto de datos de entrenamiento en partes iguales k o pliegues como se muestra en la figura 2.8. Se itera sobre el conjunto de datos, en la primera iteración la primera parte se utiliza como conjunto de prueba y las otras tres partes se usaran como conjunto de entrenamiento. Se entrena el modelo con datos de entrenamiento y se evalúa con datos de prueba. Se calcula el rendimiento y se vuelve a iterar. En la siguiente iteración, la segunda parte se usa como prueba y las otras tres como entrenamiento, se repite el proceso k veces, rotando la parte usada como conjunto de prueba. Al final de las k iteraciones, se tendrán k métricas de rendimiento, se promedian para obtener una estimación general del rendimiento del modelo.

Entre los tipos de validación cruzada existentes, se destaca la validación cruzada estratificada, usada en este proyecto. Este tipo de validación, utiliza un muestreo estratificado para asegurar que cada pliegue tiene la misma proporción de clases que el conjunto de datos en vez de dividir aleatoriamente, es muy útil cuando tenemos clases desbalanceadas, es decir, cuando algunas clases están representadas con menor frecuencia que otras clases en el conjunto de datos.

2.3. Métricas de evaluación

Una vez creado el modelo de aprendizaje y ajustado para predecir resultados para datos nuevos, se debe asegurar que el modelo sea preciso. En este apartado se explicarán las métricas usadas en este proyecto para evaluar el rendimiento de un modelo.

2.3.1. Matriz de confusión

Una matriz de confusión es una tabla útil para visualizar el desempeño de un algoritmo de clasificación comparando las predicciones con las clases reales. En la figura 2.9, las filas representan las clases reales y las columnas representan las clases predichas. La matriz de confusión está compuesta por los siguientes componentes representados en la figura 2.9.

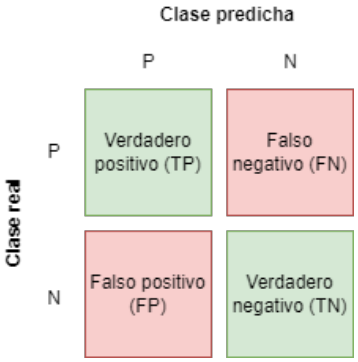


Figura 2.9. Matriz de confusión

- Verdaderos negativos (TN). Casos correctamente predichos como negativos.
- Falsos positivos (FP). Casos incorrectamente predichos como positivos.
- Falsos negativos (FN). Casos incorrectamente predichos como negativos.
- Verdaderos positivos (TP). Casos correctamente predichos como positivos.

Las métricas derivadas de esta matriz son:

- Precisión. Medida que indica la relación entre número de puntos positivos predichos correctamente y el número de todos los puntos que se han predicho como positivos. Se calcula de la siguiente manera:

$$PRE = \frac{TP}{TP + FP} \tag{2.5}$$

- Recuperación (Recall). Medida que indica la proporción de elementos positivos que se identifican correctamente entre todos los elementos que son realmente positivos. Se calcula de la siguiente manera:

$$REC = \frac{TP}{TP + FN} \tag{2.6}$$

- Exactitud (Accuracy). Medida que indica el porcentaje de predicciones correctas.

$$ACC = \frac{TP + TN}{Total} \tag{2.7}$$

- Puntuación F1 (F1-Score). Puntuación que se obtiene realizando la media armónica de precisión y recuperación proporcionando un equilibrio entre ambas. Se calcula de la siguiente manera:

$$F1 = 2 \cdot \frac{PRE \cdot REC}{PRE + REC} \tag{2.8}$$

2.3.2. Curva ROC

Una herramienta gráfica para evaluar el rendimiento de un modelo es la curva ROC [29]. Representa la relación entra la tasa de verdaderos positivos (TPR) y la tasa de falsos positivos (FPR) en diferentes umbrales de clasificación. La tasa de verdaderos positivos (TPR) o recuperación (recall) es la proporción de positivos reales que son correctamente identificados, se calcula con la ecuación 2.6. La tasa de falsos positivos (FPR) es la proporción de negativos reales que son identificados incorrectamente como positivos. Se calcula mediante la ecuación 2.9.

$$FPR = \frac{FP}{FP + TN} \tag{2.9}$$

En problemas de clasificación, los modelos pueden predecir probabilidades de las clases en vez de las etiquetas de clase directamente. Un umbral define el punto de corte para decidir si una predicción es positiva o negativa, al modificar el umbral cambiamos la TPR y FPR.

El área bajo la curva o AUC según sus siglas en inglés, mide el área total bajo la curva ROC. Este valor proporciona una única métrica para evaluar el rendimiento de un modelo de clasificación.

Un AUC de 0,5 indica que el modelo tiene un rendimiento aleatorio, en cambio, un AUC de 1,0 indica un rendimiento perfecto donde el modelo clasifica correctamente todos los ejemplos positivos y negativos. Por lo que, un área más cercano a uno nos indicará un mejor rendimiento del modelo.

En la figura 2.10 se muestran las curvas ROC para cada clase en un problema de clasificación multiclase. Los colores de la figura se corresponden con las diferentes clases. La línea diagonal punteada representa un clasificador aleatorio y sirve como referencia para evaluar el rendimiento del modelo. Un clasificador aleatorio asigna etiquetas sin ningún criterio informativo y tiene una tasa de verdaderos positivos igual que la tasa de falsos positivos para todos los umbrales posibles. Un modelo que sigue esta diagonal, tendrá un AUC de 0,5 lo que indica que no tiene capacidad de discriminación entre clases y su rendimiento equivale a realizar una predicción aleatoria.

En la figura 2.10, la clase 3 tiene el AUC más alto con 0,99, esto indica que el modelo tiene

un rendimiento casi perfecto para esta clase, es decir, el modelo clasifica muy bien los ejemplos de esta clase con muy pocos errores. Para las clases 0 y 1, se tiene un área de 0,95 lo que sugiere que el modelo tiene un alto rendimiento para estas clases, con escasos errores en la clasificación. Por último, la clase 2, tiene un área de 0,94 ligeramente más bajo que las otras clases pero aún así un valor muy alto. Esto nos indica que el modelo tiene un buen rendimiento aunque con un poco más de errores en comparación con las otras clases.

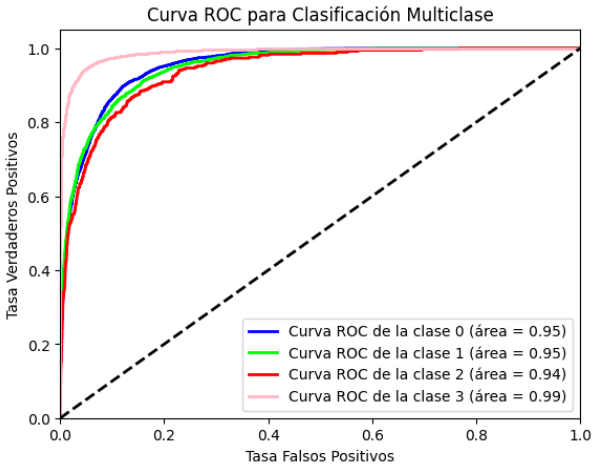


Figura 2.10. Curva ROC para clasificación multiclase.

2.4. Redes neuronales artificiales

Las redes neuronales artificiales son una rama del aprendizaje automático que se inspira en la actividad de las neuronas del cerebro humano. Están diseñadas para aprender patrones a partir de los datos mediante un proceso de entrenamiento. La figura 2.11 muestra la comparación entre una neurona biológica y una artificial.

2.4.1. Componentes

La unidad computacional básica en una red neuronal artificial es similar a una neurona que acepta señales de entrada de múltiples fuentes $(x_1, x_2, ...)$, aplica una ponderación a cada entrada $(w_1, w_2, ...)$, suma las ponderaciones y pasa el resultado a través de una función de activación. La fórmula matemática básica que resume este proceso es

$$y = f\left(\sum_{i=1}^{n} w_i \cdot x_i + b\right) \tag{2.10}$$

donde x_i son las entradas, w_i son los pesos asociados a cada entrada, b es el sesgo, f es la función de activación y y es la salida de la neurona.

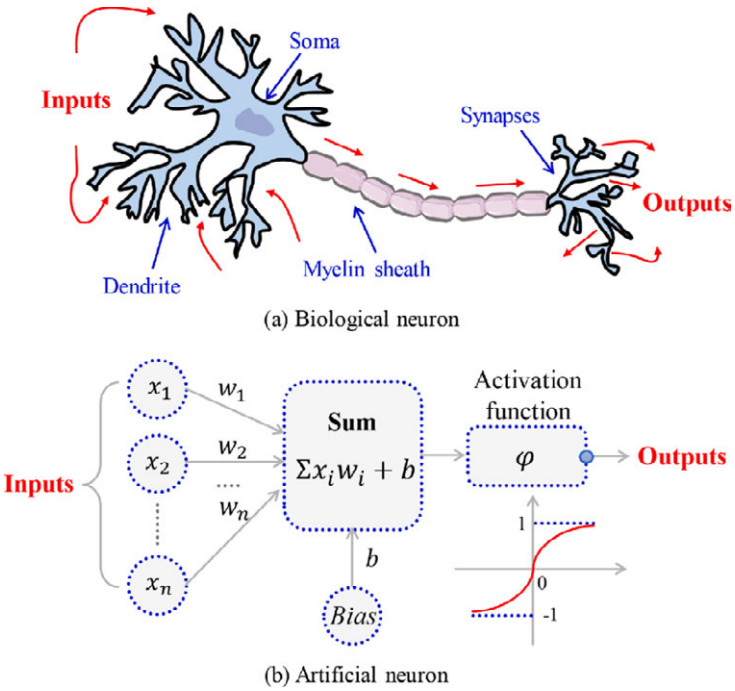


Figura 2.11. Comparación entre una neurona biológica y una neurona artificial [30].

2.4.1.1. Funciones de activación

Las funciones de activación son necesarias para introducir no linealidad en la red, permitiendo que las redes aprendan relaciones complejas en los datos. Sin la no linealidad, las redes se comportarían como un modelo lineal limitando su capacidad de resolver problemas complejos. Las funciones de activación que se han usado en este proyecto son:

- Activación unidad lineal rectificada ($ReLU$). Habilita la entrada si es positiva y no se activa para una entrada negativa. Esta característica permite que la red sea eficiente en términos de cálculo activando solamente las neuronas que realmente contribuyen a la salida. Sin embargo, esta característica puede llevar a que algunas neuronas dejen de activarse y no contribuyan al aprendizaje, especialmente si se reciben valores negativos en la entrada.

$$ReLU(x) = max(0, x) \tag{2.11}$$

- Función *softmax*. Se utiliza principalmente en la capa de salida de una red neuronal para problemas de clasificación multiclase. En lugar de dar un índice de clase único, proporciona la probabilidad de cada clase. Convierte un vector de valores z en un vector de probabilidades, donde cada valor se encuentra entre 0 y 1 y la suma de todas las probabilidades es 1.

$$softmax(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}} \tag{2.12}$$

- Activación lineal exponencial gaussiana (*GELU*) [31]. Calcula la probabilidad de que una unidad de entrada sea activa multiplicando la entrada por la distribución normal acumulativa de la entrada. Esto permite una activación suave y probabilística, proporcionando una mejor convergencia en redes profundas y un mejor rendimiento del modelo [32]. Se define mediante la ecuación 2.13

$$GELU(x) = x \cdot \Phi(x) \tag{2.13}$$

donde $\Phi(x)$ es la función de distribución normal acumulativa.

2.4.2. Funciones de pérdida

Una función de pérdida calcula la diferencia entre los valores reales y las predicciones realizadas por un modelo, cuantificando su error y proporcionando información sobre como se desenvuelve. El entrenamiento de un modelo se basa en minimizar esa diferencia tanto como sea posible, ajustando sus parámetros, como los pesos de la red neuronal, con la finalidad de mejorar su desempeño.

El gradiente de la función de pérdida con respecto a los parámetros, proporciona la dirección en la que se deben ajustar los parámetros para reducir la pérdida. El gradiente se calcula usando técnicas de derivación, que miden cómo cambia la función de pérdida cuando se modifican los parámetros.

Para minimizar la pérdida, los parámetros se actualizan iterativamente en la dirección opuesta al gradiente, método de ajuste conocido como descenso de gradiente. En cada iteración, se calcula el gradiente de la función de pérdida con respecto a los parámetros actuales. Después, se actualizan los parámetros moviéndose en dirección opuesta al gradiente, ya que el gradiente apunta hacia el aumento de la función de pérdida. Este proceso de actualización se repite un número determinado de iteraciones o hasta que la función de pérdida converge a un valor mínimo, esto ocurre cuando los cambios en la función de pérdida entre iteraciones son insignificantes.

Una función de pérdida destacada en los problemas de clasificación es la función de perdida de entropía cruzada o *Cross Entropy Loss* [33]. Penaliza el modelo por producir salidas erróneas con alta probabilidad, incentivando al modelo a asignar una alta probabilidad a las clases correctas y una baja probabilidad a clases incorrectas. En clasificación multiclase se define como

$$CrossEntropyLoss = -\sum y \log \hat{y} \tag{2.14}$$

donde y representa la distribución de probabilidad real de las clases, y $\hat{y}$ representa las probabilidades predichas por el modelo para cada clase. El sumatorio se realiza sobre todas las clases posibles. En el contexto de clasificación multiclase, estas distribuciones se representan como vectores de probabilidad. Si $\hat{y}$ es alta para una clase incorrecta , donde $y = 0$, el término $y \log \hat{y}$

contribuye poco a la pérdida porque y es 0, por lo que se penaliza por altas probabilidades a clases incorrectas. En cambio, si $\hat{y}$ es baja para la clase correcta, donde $y = 1$, el término $y \log \hat{y}$ se vuelve negativo y grande, aumentando la pérdida ya que el logaritmo de un número cercano a 0 es un valor negativo grande, por lo que, se penaliza por bajas probabilidades a la clase correcta.

La entropía cruzada siempre es no negativa, es mínima cuando la distribución predicha es igual a la distribución real y es diferenciable, por lo que es útil en la optimización de redes neuronales. Otra función de pérdida relevante en este proyecto es la función de pérdida focal o *Focal Loss* [34]. Está función está diseñada para abordar la desproporción de clases. Da menos importancia a las muestras bien clasificadas y pone más énfasis a las muestras difíciles, ayudando así a reducir el impacto del desequilibrio de clases en el entrenamiento del modelo. Se puede aplicar a problemas de clasificación multiclase, la ecuación 2.15 queda de la siguiente manera:

$$\mathrm{FL}(p_t) = -\alpha_t (1 - p_t)^{\gamma} \log(p_t) \tag{2.15}$$

donde p_t es la probabilidad pronosticada del modelo para la clase verdadera t, α_t es un factor de ponderación para la clase t y γ es el parámetro de enfoque que ajusta la amplitud del efecto focal. En el contexto multiclase [35], la probabilidad p_t se obtiene aplicando una función *softmax*, descrita en la ecuación 2.12, a las salidas del modelo, dando como resultado una distribución de probabilidad sobre todas las clases.

2.4.3. Optimizadores

Los optimizadores son algoritmos que se utilizan para ajustar los pesos y sesgos de la red durante el entrenamiento, minimizando la función de pérdida. El objetivo principal de un optimizador es encontrar los mejores valores para los parámetros del modelo que reduzcan el error en las predicciones. Los optimizadores más comunes y usados en este proyecto son:

- Descenso del gradiente estocástico (SGD) [36]. Ajusta los pesos de la red utilizando un solo ejemplo de entrenamiento o un mini lote de ejemplos en cada iteración. En lugar de calcular el gradiente de la función de pérdida sobre todo el conjunto de datos, lo realiza de manera aproximada sobre un subconjunto de datos. De esta manera, se consigue mayor velocidad al trabajar con menos datos en cada iteración y debido a la naturaleza ruidosa del gradiente estimado a partir de un subconjunto de datos, el SGD puede evitar quedar atrapado en mínimos locales poco profundos. Sin embargo, puede tener una convergencia menos estable porque se basa en una pequeña cantidad de datos en cada paso. Además, se puede utilizar con diferentes estrategias para mejorar su rendimiento:

  - *Momento.* Ayuda a acelerar la convergencia acumulando un promedio ponderado de gradientes pasados. Esta información se usa para actualizar los parámetros, manteniendo la dirección y disminuyendo oscilaciones, mejorando así la velocidad de la convergencia.

- *Decay del aprendizaje.* Ajusta la tasa de aprendizaje durante el entrenamiento, puede ayudar a que el modelo se ajuste mejor al mínimo global de la función de pérdida. Puede ser lineal, exponencial o basado en un esquema adaptativo.
- *Mini-batch SGD.* En lugar de utilizar solo un ejemplo de entrenamiento, se pueden usar pequeños lotes de datos, mini-batches, para calcular el gradiente. Esto balancea la varianza de estimación del gradiente ofreciendo una convergencia más estable.

- *Adam* (*Adaptative Moment Estimation*). Combina las mejores ideas de dos métodos de optimización AdaGrad y RMSProp. Adam ajusta las tasas de aprendizaje para cada parámetro de manera adaptativa, lo que significa que puede cambiar cuánto se ajustan los pesos en cada paso dependiendo de cómo han cambiado anteriormente. Además, mantiene una media móvil de los gradientes y una media móvil de los cuadrados de los gradientes, permitiendo un ajuste más preciso y estable de los pesos. Por último, incluye correcciones de sesgo que ajustan las medias móviles en función del número de iteraciones, mejorando la precisión del optimizador [37].

2.4.4. Estructura

Las redes están formadas por nodos o neuronas que están interconectadas y organizadas en capas, como se puede apreciar en la figura 2.13. Una red neuronal se compone de tres tipos de capas.

- La primera capa de la red es la capa de entrada, que recibe los datos iniciales. Cada neurona o nodo de esta capa representa una característica del conjunto de datos.
- La última capa de la red es la capa de salida y nos proporciona el resultado final de la red. La cantidad de nodos en esta capa dependen del tipo de problema al que nos enfrentemos, en el caso de un problema de clasificación multiclase, la cantidad de nodos viene definido por el número de clases presentes en las etiquetas de nuestros datos.
- Un conjunto de una o más capas intermedias o capas ocultas. Cada capa aplica una serie de transformaciones no lineales.

2.4.5. Entrenamiento

El entrenamiento de una red se basa en ajustar los pesos y sesgos para minimizar la diferencia entre predicciones y valores reales. Este proceso consta de tres fases principales mostradas en la figura 2.12: propagación hacia adelante, cálculo de la función de pérdida y propagación hacia atrás.

En la fase de propagación hacia adelante (*Forward propagation*), las entradas se transmiten a través de las capas de la red, desde la capa de entrada hasta la capa de salida. En cada capa,

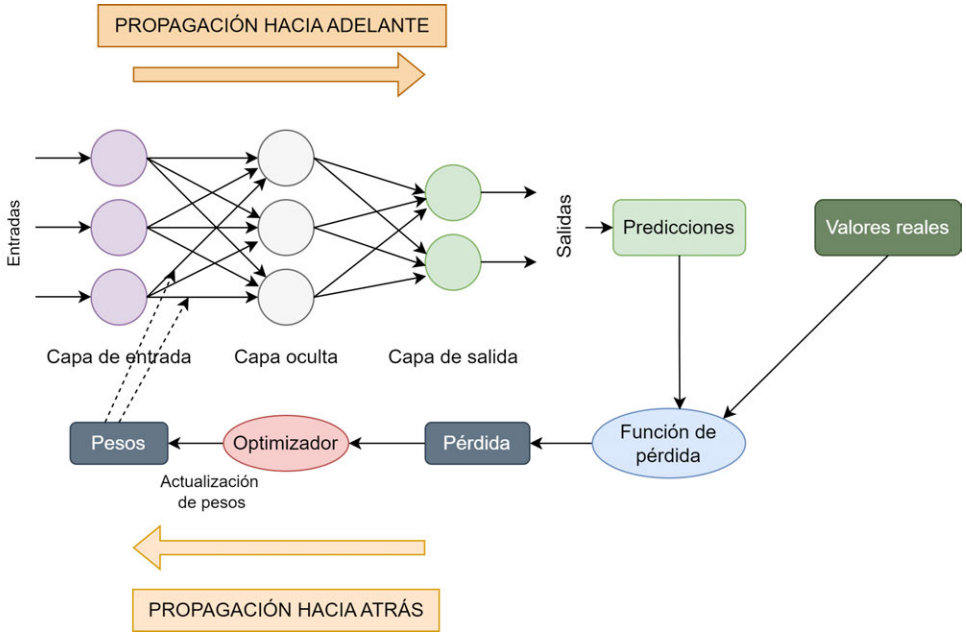


Figura 2.12. Proceso de entrenamiento de una red neuronal

los nodos calculan la salida aplicando una función de activación a una combinación lineal de sus entradas. Mediante este proceso se generan las predicciones de la red. La figura 2.13 muestra la estructura de una red de propagación hacia adelante.

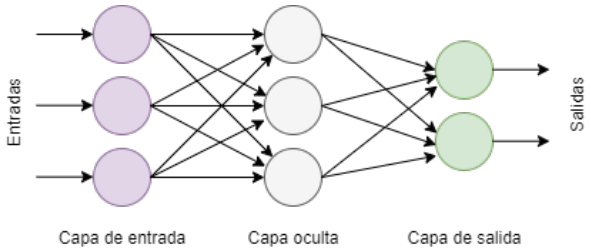


Figura 2.13. Red neuronal feedforward

Después de la propagación hacia adelante, se calcula la función de pérdida J, que mide la diferencia entre las predicciones de la red y los valores reales. La función de pérdida proporciona una medida cuantitativa del error del modelo, y su minimización es el objetivo principal del entrenamiento.

Para minimizar la función de pérdida, se utiliza el algoritmo de retropropagación (*Backpropagation*). Primero, se calcula la salida de la red y la función de pérdida. Luego, en la propagación hacia atrás, se calculan los gradientes de la función de pérdida con respecto a cada peso utilizando la regla de la cadena. Estos gradientes indican cómo deben ajustarse los pesos para reducir el error.

La actualización de los pesos se realiza mediante el algoritmo de optimización, como el des-

censo de gradiente. Los pesos se actualizan de la siguiente manera:

$$w_i = w_i - \eta \frac{\partial J}{\partial w_i} \tag{2.16}$$

donde η es la tasa de aprendizaje y J es la función de pérdida. La tasa de aprendizaje determina el tamaño del paso que se da en la dirección del gradiente negativo. La optimización implica actualizar iterativamente los parámetros en la dirección opuesta al gradiente de la función de pérdida. Este proceso se repite un número determinado de iteraciones o hasta que la función de pérdida converge a un valor mínimo.

2.4.6. Regularización y Normalización

El sobreajuste ocurre cuando una red aprende demasiado bien el conjunto de datos de entrenamiento y no es capaz de generalizar a nuevos datos nunca vistos. Las técnicas de regularización y normalización son necesarias a la hora de mejorar el rendimiento, la estabilidad y evitar el sobreajuste. Las técnicas más comunes se describen a continuación.

- Abandono (*Dropout*). Técnica de regularización. Consiste en desactivar aleatoriamente algunas neuronas en cada capa durante el entrenamiento, evitando así que la red dependa demasiado de ciertas neuronas. Ayuda a que la red generalice mejor y reduce el sobreajuste.
- Regularización L2 (*Weight Decay*). Añade una penalización a la función de pérdida basada en la magnitud de los pesos. Dicha penalización ayuda a evitar que los pesos se vuelvan demasiado grandes, los modelos serán más simples y generalizables.
- Normalización por lotes (*Batch Normalization*). Técnica utilizada para mejorar la velocidad, la estabilidad y el rendimiento de la red. Normaliza las activaciones de las neuronas en cada capa para cada mini lote de datos de entrada. Se ajustan y escalan las activaciones para que tengan una media cercana a cero y una desviación estándar cercana a uno, de esta manera se pueden utilizar tasas de aprendizaje más altas.

Capítulo 3

Especificaciones y restricciones de diseño

En este capítulo se detallan las especificaciones y restricciones de diseño que se deben cumplir en el desarrollo de este proyecto. Estas directivas son necesarias para asegurar que se la solución final cumpla con los objetivos establecidos en esta memoria.

3.1. Especificaciones

La solución desarrollada debe cumplir con una serie de especificaciones para garantizar cumplir y el cumplimiento y alcance de los objetivos marcados inicialmente. El objetivo principal del proyecto consiste en la exploración y el desarrollo de un modelo de redes neuronales que consiga competir con los algoritmos de aprendizaje automático *SVM* y *Random Forest* en la clasificación de tejido cerebral. La herramienta debe cumplir con las especificaciones enumeradas a continuación:

- La herramienta debe ser capaz de trabajar con imágenes hiperespectrales, desde la carga de este tipo de datos hasta el procesamiento y visualización de los mismos.
- La herramienta debe ser capaz de clasificar las firmas espectrales de los tejidos cerebrales de acuerdo a cuatro clases: sano, tumor, vena, duramadre.
- La herramienta debe ser capaz de comparar los modelos entre sí, otorgando una puntuación a cada modelo en función de sus métricas, por lo que, estas métricas deben estar en concordancia con la problemática que se aborda en este proyecto.
- La herramienta debe presentar los resultados de los modelos en un formato que sea comprensible por un usuario no experto en la materia. La visualización de los resultados debe ser intuitiva y clara.

3.2. Restricciones de diseño

A continuación se detallan los dos tipos de restricciones de diseño encontradas en el desarrollo de este proyecto, clasificadas en nivel hardware y nivel software.

3.2.1. Restricciones a nivel de hardware

Las restricciones a nivel de hardware vienen dadas por elementos físicos disponibles para el desarrollo y la ejecución de la herramienta.

- Memoria y capacidad de procesamiento. El ordenador utilizado tiene especificaciones limitadas, 8GB de RAM y procesador específico. Estas limitaciones afectan a los tiempos de entrenamiento, capacidad de manejo de volúmenes altos de datos, tiempos de búsqueda de mejores hiperparámetros.

3.2.2. Restricciones a nivel de software

A nivel de software las restricciones vienen dadas por las herramientas y el lenguaje de programación seleccionado.

- Entorno de desarrollo. Basado en *Visual Studio Code*, impone restricciones de compatibilidad de extensiones y herramientas para desarrollo y depuración de código.

- Lenguaje de programación. La herramienta se ha implementado utilizando lenguaje *Python*, y bibliotecas como *Scikit-Learn* [38] para el desarrollo de los algoritmos de aprendizaje automático y *Pytorch* [39] para la implementación de los modelos de redes neuronales. Esto implica adaptarse a sus actualizaciones, documentaciones y limitaciones de funcionalidades.

Capítulo 4

Descripción de la solución propuesta

La finalidad del proyecto se puede resumir en la comparación de modelos de aprendizaje automático con modelos de redes neuronales artificiales. La solución propuesta consiste en el desarrollo de una herramienta capaz de realizar esta comparación de forma automatizada.

4.1. Descripción de los datos

Los datos se obtienen de la base de datos de *SLIM Brain* [40]. Se trata de una base de datos multimodal de imágenes de cerebros humanos *in-vivo*, creada para la detección de tumores. La base de datos incluye imágenes hiperespectrales, RGB y de profundidad capturadas durante intervenciones quirúrgicas.

El conjunto de datos o *dataset*, está formado por firmas espectrales etiquetadas que se obtienen tras aplanar el cubo hiperespectral y su posterior enmascaramiento según los píxeles etiquetados en el mapa de *ground truth*.

Para este proyecto, se seleccionaron doce pacientes de la base de datos, de los cuales nueve se utilizarán como conjunto de entrenamiento y tres se utilizarán como conjunto de test.

Cada paciente presenta un número variable de firmas espectrales, todas capturadas con la misma cámara hiperespectral,lo que asegura que cada firma comparta 25 características comunes.

El conjunto de datos contiene siete clases de etiquetas representadas en la tabla 4.1.

Clase	Tipo
101	Materia Gris
200	GBM Puro
221	Astrogial
241	Meningioma
301	Vaso sanguíneo venoso
302	Vaso sanguíneo arterial
320	Duramadre

Tabla 4.1. Tabla de etiquetas en 7 clases

4.2. Descripción de la herramienta

La herramienta permite el entrenamiento de diversos modelos, cuyos parámetros y tipo de modelo, se especifican mediante un archivo de configuración. Posteriormente, se llevan a cabo las predicciones de cada modelo utilizando datos nunca vistos por los modelos. Finalmente, se presentan los resultados en un formato legible y comprensible para un usuario no experto en la materia. Se añade una funcionalidad para buscar los mejores hiperparámetros.

En la figura 4.1 se presenta la funcionalidad de la herramienta de manera esquematizada. El flujo de trabajo de la herramienta se puede resumir en tres etapas principales:

- Preparación de los datos. El módulo de procesamiento recopila y prepara los datos necesarios para la siguiente etapa.
- Entrenamiento y evaluación. El módulo de entrenamiento ajusta los parámetros de cada modelo. Se registran los modelos y las métricas de rendimiento y se realizan predicciones sobre un conjunto de datos nunca vistos.
- Generación de un informe. El módulo generador de informes automatiza la creación de un informe con las métricas de evaluación y los resultados de las predicciones.

En las siguientes secciones se explicarán los módulos que forman la herramienta.

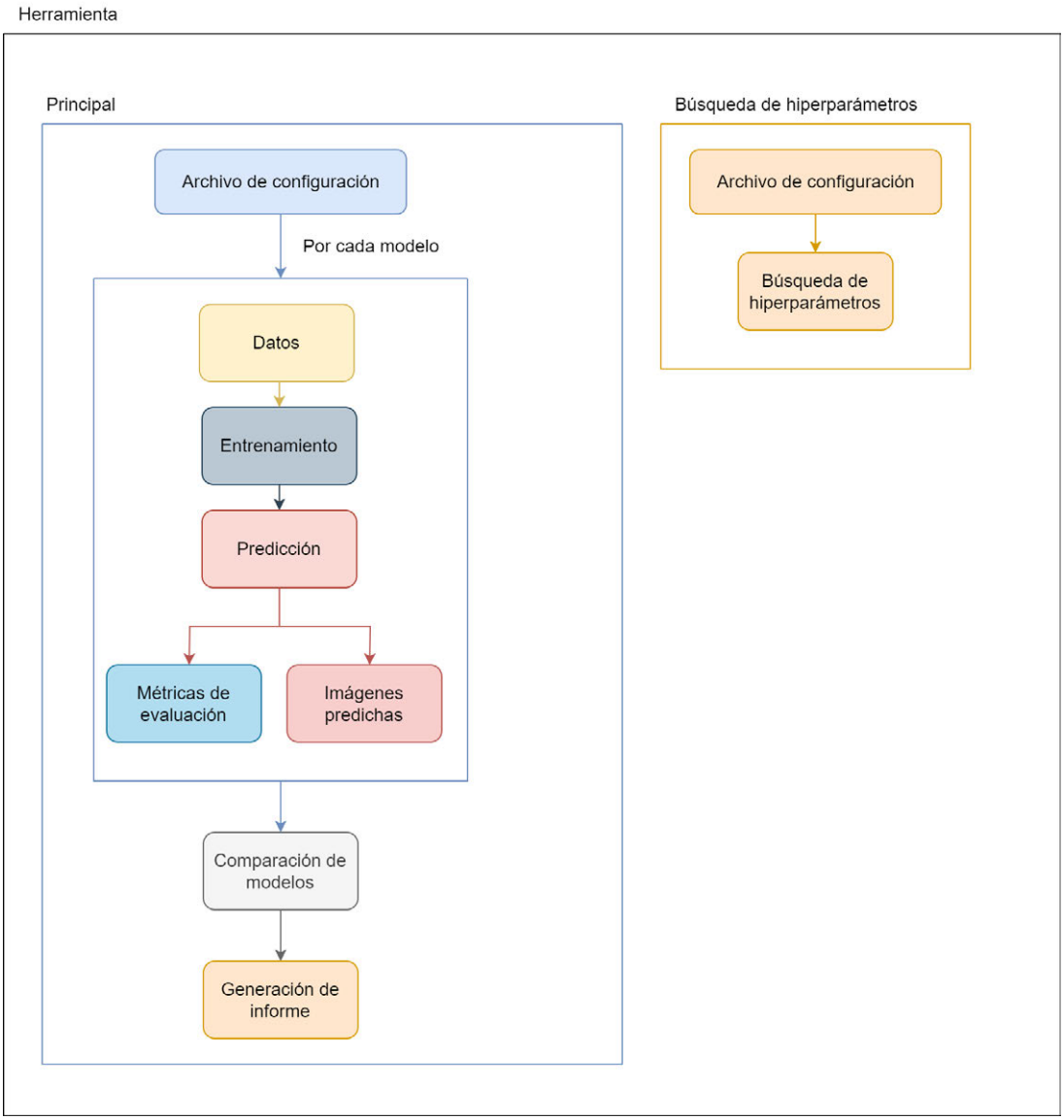


Figura 4.1. Esquema general

4.3. Módulo principal

El módulo principal integra los componentes descritos, coordinando un flujo de trabajo eficiente y estructurado. Se inicia importando el archivo de configuración, que contiene las rutas necesarias para procesar los datos y almacenar los resultados de la herramienta. También define los modelos y los parámetros específicos para el entrenamiento y la predicción.

A continuación, se procesan los datos. Tras esto, se entrena cada modelo con los parámetros definidos. En el caso de las redes neuronales, se emplea un registro de modelos que gestiona las arquitecturas disponibles, asociando cada nombre de modelo (clave) con un conjunto específico de procedimientos de configuración (valor). Cuando se necesita entrenar una red neuronal, el sistema consulta el registro para localizar y aplicar los procedimientos adecuados de configuración del modelo deseado.

Para cada modelo, se inicializa el módulo de predicción y se realizan predicciones sobre datos de validación especificados en el archivo de configuración. Estas predicciones se comparan con los datos de referencia *ground truth* para evaluar el rendimiento del modelo. Las métricas de evaluación se calculan y almacenan, y posteriormente se comparan para determinar el modelo más efectivo mediante el modulo de comparación de modelos. Finalmente, se genera un informe de resultados.

En resumen, este módulo integra las tareas necesarias para el procesamiento, entrenamiento, predicción, evaluación y reporte de resultados.

4.4. Procesado de los datos

El primer módulo de la herramienta engloba los pasos necesarios a seguir para preprocesar nuestros datos.

El módulo encargado de procesar los datos se inicia importando los datos de entrenamiento desde la fuente especificada en el archivo de configuración. A continuación, se aplican transformaciones tanto a las características como a las etiquetas, y finalmente, los datos procesados se almacenan en un formato adecuado para su uso en otros módulos del proyecto.

El proceso de carga inicia con la importación de firmas espectrales etiquetadas de los pacientes que conforman el conjunto de entrenamiento. Al concatenar estos datos, se obtiene un *dataset* más completo. Posteriormente, se procede a la transformación de características aplicando un escalado estándar mediante *StandardScaler* de la biblioteca de *scikit-learn*. Este escalador ajusta los datos para que presenten una media de cero y una desviación estándar de uno. Este paso es fundamental para la normalización de los datos y la mejora del rendimiento del modelo. Es especialmente útil en modelos basados en distancias, como SVM, ya que funcionan mejor cuando las características están en una misma escala [41], [42]. Sin este escalado, las características con

valores mayores podrían dominar el comportamiento del modelo, afectando a su desempeño.

En relación con las etiquetas, se realiza una conversión que agrupa las clases originales de siete a cuatro, como se muestra4.2. Este agrupamiento no elimina datos sino que simplifica el problema de clasificación, lo que puede mejorar la precisión del modelo especialmente si las clases originales presentan similitudes [43]. Las etiquetas se transforman posteriormente empleando *Label Encoder* de *scikit-learn*, convirtiendo las etiquetas categóricas en numéricas asegurando que estén en un formato adecuado. Así, las etiquetas se convierten en los valores $0, 1, 2, 3$. En la tabla 4.2 se muestra la correspondencia de cada etiqueta. Este formato simplifica la interpretación y procesamiento por parte del modelo, mejorando la eficiencia computacional y reduciendo errores. Las etiquetas deben presentarse en este formato ya que la función de pérdida *Cross Entropy Loss*, utilizada por defecto, requiere que las etiquetas sean valores enteros correspondientes a los índices de las clases para asignar correctamente la probabilidad predicha a la clase correspondiente. En la figura 4.2 se resume el procesado de características y etiquetas.

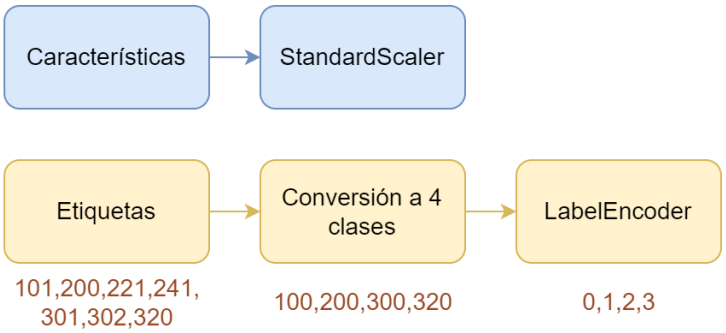


Figura 4.2. Procesado de los datos

Finalmente, los datos transformados y los objetos de transformación, *StandardScaler* y *LabelEncoder*, se almacenan para garantizar que la transformación pueda ser replicada en fases posteriores.

Clase	Etiqueta codificada	Tipo	Color
100	0	Sano	Verde
200	1	Tumor	Rojo
300	2	Vena	Azul
320	3	Duramadre	Rosa

Tabla 4.2. Tabla de etiquetas en 4 clases

4.5. Entrenamiento

El módulo de entrenamiento gestiona desde la carga de los datos preprocesados hasta el entrenamiento, evaluación y almacenamiento del modelo.

El proceso inicia con la carga de datos preprocesados almacenados previamente por el modulo procesador de datos. Se dividen los datos en conjuntos de entrenamiento y prueba, permitiendo así la evaluación imparcial de los modelos en datos no vistos anteriormente.

Dependiendo del tipo de modelo, el entrenamiento varía. Para modelos como SVM o *Random Forest* se utiliza las funciones proporcionadas por *scikit-learn*. Para las redes neuronales, sin embargo, se define una arquitectura específica utilizando *Pytorch* y el entrenamiento se lleva a cabo en mini-batches para mejorar la eficiencia y estabilidad del modelo [44].

En el caso de las redes neuronales, el entrenamiento comienza con la definición de la arquitectura de la red. La figura 4.3 muestra las diferentes arquitecturas de red neuronal desarrolladas, se detallan a continuación. Es importante destacar que la elección de la función de activación se basa en los mejores resultados obtenidos al probar al probar ReLU, Leaky ReLU y GELU.

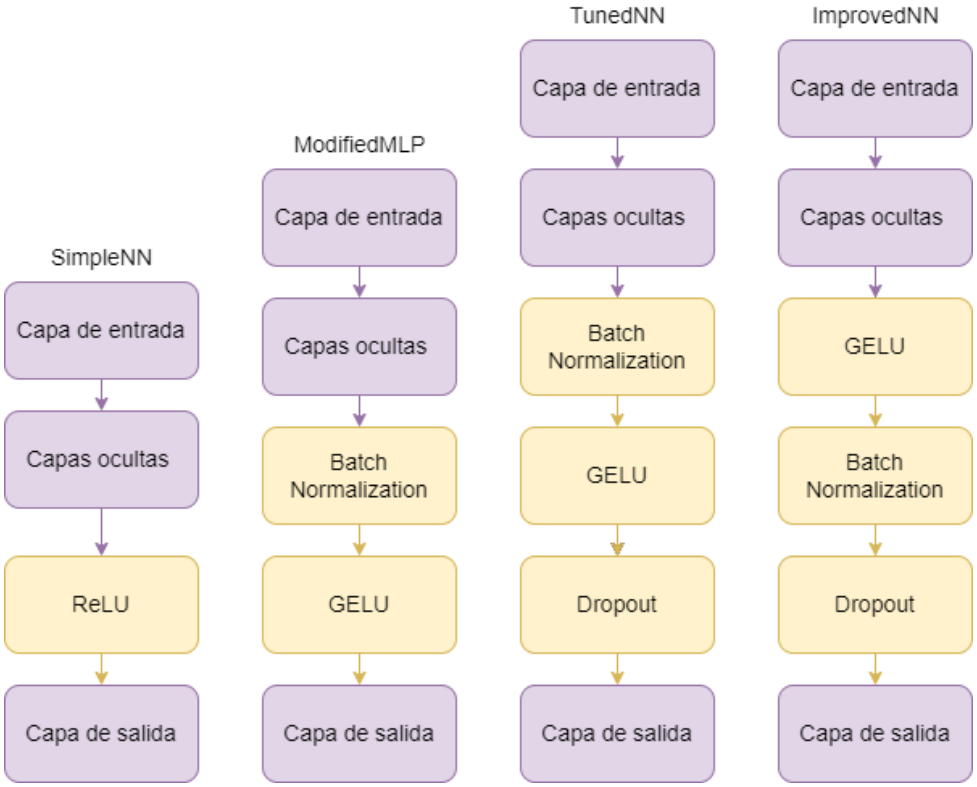


Figura 4.3. Modelos de red neuronal

- *SimpleNN*. Red neuronal simple que consiste en capas completamente conectadas y función de activación *ReLU*. En esta arquitectura, cada neurona de una capa está conectada a cada neurona de la capa siguiente. La función de activación *ReLU*, se utiliza después de cada capa completamente conectada excepto la última capa. *ReLU* es computacionalmente eficiente y ayuda a mitigar el problema de desvanecimiento de gradiente [45], problema común con funciones de activación *sigmoid* y *tanh* [46].
- *ModifiedMLP*. Es una versión modificada de la red anterior. Incluye normalización por

lotes después de cada capa completamente conectada. La función de activación GELU es particularmente compatible con la normalización por lotes ya que tiene en cuenta la distribución normal de las entradas de las neuronas. La normalización por lotes ayuda a estabilizar y acelerar el entrenamiento de la red [47], [48]. Normaliza las salidas de una capa para tener una media cercana a cero y una desviación estándar cercana a uno ayudando a mitigar el problema del desvanecimiento o explosión del gradiente y permite el uso de tasas de aprendizaje más altas [49], [50].

- *TunedNN*. Agrega la técnica de *Dropout*, además de la normalización por lotes y función de activación *GELU*. *Dropout* evita el sobreajuste desactivando aleatoriamente una fracción de las neuronas durante el entrenamiento obligando a la red a no depender de neuronas especificas y distribuir la representación aprendida de las neuronas. Esta técnica es útil en este proyecto, un problema con muchas características y un alto riesgo de sobreajuste[51]. La combinación de normalización por lotes y función de activación *GELU* estabiliza y mejora el entrenamiento aprovechando las ventajas de ambas técnicas.

- *ImprovedNN*. Es una versión avanzada que cambia el orden de las capas, colocando la función de activación antes de la normalización por lotes y el *Dropout*. Este orden especifico en el que se normalizan las salidas de las funciones de activación, puede llevar a un entrenamiento mas estable y rápido. Este enfoque ayuda a la red a aprender de manera más efectiva, ya que las salidas de las activaciones no se desvían demasiado, lo que permite una mejor propagación de los gradientes.

Posteriormente, se inicializan los pesos, y se selecciona el optimizador y la función de pérdida, que pueden ser especificados en el archivo de configuración. Por defecto, se utiliza el optimizador *Adam* y la función *Cross Entropy Loss*. El bucle de entrenamiento itera sobre un número de épocas definido, dentro de cada época se divide el conjunto de entrenamiento en mini lotes o *mini-batches*. Para cada mini-batch se realiza una pasada hacia adelante a través de la red para calcular las predicciones. Se calcula la perdida comparando las predicciones con las etiquetas reales y se realiza la retropropagación para calcular los gradientes. Por último se actualizan los pesos de la red utilizando el optimizador. Este proceso se resume en la figura 4.4.

Se registra el historial de entrenamiento, incluyendo la pérdida y la precisión en cada época, lo que permite monitorear el progreso del entrenamiento. Una vez finalizado el entrenamiento, el modelo se evalúa utilizando el conjunto de prueba o *test*. Las métricas de evaluación incluyen la precisión, recuperación, puntuación F1, exactitud, también se registra el tiempo de entrenamiento. En el caso de las redes neuronales, se utilizan tensores de *Pytorch* para calcular las predicciones y probabilidades.

Finalmente, los modelos se guardan para facilitar su reutilización e implementación. Asimismo, se almacene las etiquetas y predicciones de cada modelo para su posterior uso.

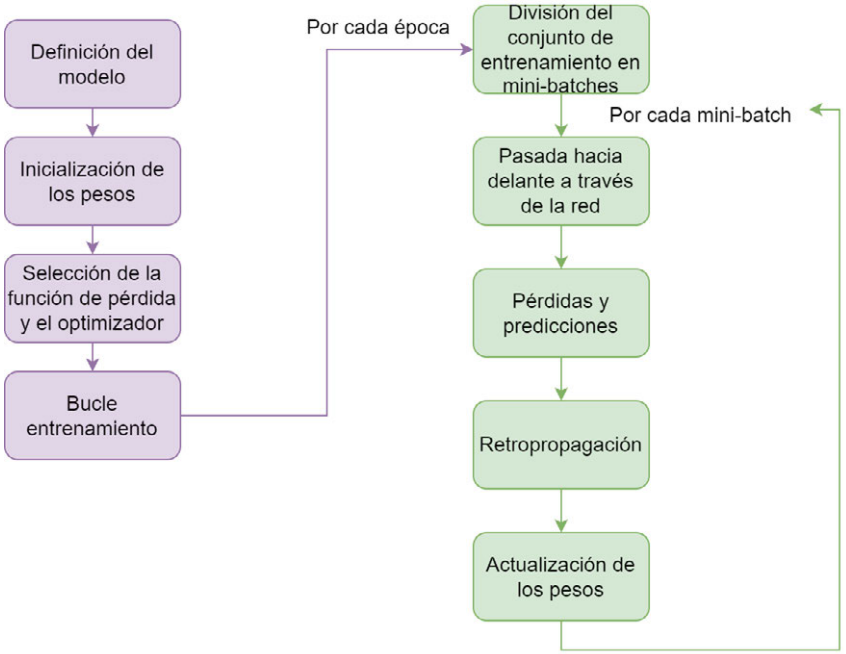


Figura 4.4. Proceso entrenamiento red neuronal

4.6. Predicción

El módulo de predicción está diseñado para realizar predicciones utilizando un modelo previamente entrenado y almacenado por el módulo de entrenamiento. Es capaz de procesar y predecir en múltiples conjuntos de datos para generar mapas de clasificación.

El proceso se inicia con la carga del modelo, así como del escalador (*StandardScaler*) y del codificador de etiquetas (*LabelEncoder*). Esto asegura que las transformaciones aplicadas a los datos de validación sean consistentes con las realizadas a los datos de entrenamiento. Los datos de validación, en forma de cubos HSI, son aplanados y normalizados utilizando el escalador.

En el caso de las redes neuronales, el módulo configura el modelo en modo de no entrenamiento (modo evaluación) y realiza predicciones en lotes. Las etiquetas predichas se transforman de nuevo a valores originales utilizando el codificador de etiquetas, y se mapean a índices de color para facilitar la visualización en los mapas de clasificación. Los mapas de clasificación resultantes, que muestran las predicciones realizas por el modelo mediante un esquema de colores personalizado mostrado en la tabla 4.2, se generan y se almacenan.

Finalmente, los resultados de las predicciones, incluyendo los mapas de clasificación, las etiquetas predichas y las probabilidades asociadas a las predicciones, se guardan para facilitar análisis posteriores, como la representación de la curva ROC.

4.7. Búsqueda de mejores hiperparámetros

En el ámbito del aprendizaje automático, SVM y *Random Forest*, el proceso de búsqueda de hiperparámetros es crucial para optimizar el rendimiento del modelo ajustando sus hiperparámetros mediante técnicas de validación cruzada y búsqueda en cuadrícula. Estas técnicas pertenecen a la biblioteca *scikit-learn*, que proporciona las herramientas necesarias para realizar estas búsquedas.

La búsqueda de hiperparámetros se lleva a cabo mediante validación cruzada estratificada (*StratifiedKFold*), una técnica que divide los datos en subconjuntos y evalúa el modelo en cada uno para obtener una estimación precisa del rendimiento. Esta estrategia es beneficiosa en presencia de clases desbalanceadas, manteniendo la proporción original de clases en cada subconjunto. Se combina con búsqueda de cuadrícula.

Una vez finalizado el proceso, los mejores hiperparámetros y sus respectivas puntuaciones se almacenan en un archivo.

Para las redes neuronales, el proceso es similar. Se inicia con la carga de la configuración del modelo y los datos necesarios para el entrenamiento. Se generan todas las combinaciones posibles de hiperparámetros especificados en la configuración.

Por cada combinación y cada modelo, se inicializa y entrena una instancia de modelo de red neuronal utilizando el módulo de entrenamiento. Se registran y almacenan las métricas de rendimiento junto con el historial de entrenamiento y el modelo entrenado, asignando identificadores únicos para cada prueba. Finalmente, se guardan las combinaciones de hiperparámetros, sus métricas asociadas y los historiales de entrenamiento.

4.8. Evaluación de las predicciones

El módulo que se encarga de evaluar las predicciones está diseñado para analizar y visualizar el rendimiento de modelos de clasificación.

El módulo se inicia cargando el codificador de etiquetas (*LabelEncoder*), las predicciones del modelo y las probabilidades asociadas. El codificador se utiliza para transformar las etiquetas predichas en formato $0, 1, 2, 3$ a formato original $100, 200, 300, 320$. Se transforman tanto las etiquetas predichas como el *ground truth*, facilitando la evaluación comparativa.

A continuación, se calculan las métricas para evaluar la precisión, la exactitud, la recuperación, la puntuación F1 y ROC-AUC. Las predicciones se evalúan utilizando el mapa de *ground truth*. Al comparar las predicciones del modelo con el mapa de *ground truth*, se puede evaluar como de bien ha funcionado el modelo en términos de localización y delimitación de las zonas de tejido. Aunque esta evaluación es limitada puesto que solo se evalúa la predicción para los píxeles etiquetados en el mapa de *ground truth*. Esto puede influir en las métricas, ya que no se

tiene en cuenta el rendimiento del modelo en áreas no etiquetadas, por lo que tendremos una visión parcial pero útil del rendimiento de cada modelo.

Una vez calculadas las métricas, se almacenan en un archivo para análisis posteriores. Además, las predicciones se almacenan en otro archivo permitiendo una revisión detallada de las etiquetas verdaderas y predichas. Para mejorar la interpretación de resultados, el módulo incluye funciones para visualizaciones gráficas como las curvas ROC para cada clase y las matrices de confusión.

4.8.1. Mejor modelo en base a las mejores métricas

El módulo permite la comparación de múltiples modelos basándose en las métricas calculadas. Se crea una puntuación final compuesta para cada modelo que se calcula mediante la ecuación 4.1. Los pesos asignados a cada métrica se representan en la tabla 4.3 y se justifican a continuación:

- Puntuación F1 (F1-Score). Es crucial en problemas de salud ya que proporciona un equilibrio entre precisión (evitar falsos positivos) y recuperación (evitar falsos negativos). Dado que se están clasificando condiciones médicas como tumores, es importante tener un buen equilibrio entre estas dos métricas. El peso asignado del 30 % refleja la importancia en garantizar un rendimiento equilibrado en todas las clases.

- ROC-AUC. Mide la capacidad del modelo para distinguir entre diferentes clases. Es especialmente útil en clasificación multiclase para evaluar cómo de bien el modelo puede separar clases entre sí. En problemas médicos, la discriminación entre clases, por ejemplo, distinguir entre tumor o tejido sano es crítica, por lo que ROC-AUC es una métrica importante y se le asigna un peso de 30 % para enfatizar la importancia de la capacidad de discriminación del modelo.

- Exactitud (Accuracy). Aunque es una métrica básica, proporciona una visión general de la eficacia del modelo. Sin embargo, en problemas de clasificación desbalanceada, puede no ser suficiente por sí sola, por lo que su peso es menor en comparación con F1-Score y ROC-AUC. Se le asigna un peso del 20 % para darle importancia pero no tanto como las métricas que manejan mejor el desbalanceo.

- Precisión (Precision). Es crucial evitar falsos positivos, especialmente para la clase tumor. Sin embargo, dado que F1-Score ya incluye precisión y recuperación, se le da un peso menor, un 10 %, para reflejar su importancia sin duplicar el enfoque que ya se captura con F1-Score.

- Recuperación (Recall). Es fundamental para asegurarse de que los casos tumor no se pasen por alto. Al igual que la precisión, se le asigna un peso del 10 % por el mismo motivo.

El cálculo de una puntuación permite la elección del modelo que se adapte mejor a las especificaciones y requisitos del proyecto.

$$\text{Score} = (\text{accuracy} \times 0{,}2 + (\text{precision} \times 0{,}1) + (\text{recall} \times 0{,}1) + (\text{f1} \times 0{,}3) + (\text{roc\_auc} \times 0{,}3) \quad (4.1)$$

Métrica normalizada	Peso
Accuracy	0.2
Precision	0.1
Recall	0.1
F1-Score	0.3
ROC-AUC	0.3

Tabla 4.3. Tabla de la importancia de cada métrica

4.9. Generación del informe de resultados

Con el objetivo de presentar de manera clara y organizada los resultados obtenidos por los modelos se crea el módulo encargado de generar un informe. Este módulo utiliza la biblioteca *FPDF* [52] de *Python* para crear un documento en el que se incluyen las métricas de evaluación así como las imágenes generadas durante la predicción. Se combinan las imágenes de *ground truth* y las imágenes RGB, permitiendo una comparación directa y clara entre las predicciones del modelo y el *ground truth*.

Mediante este informe se proporciona una visión general del rendimiento de los modelos y también se aplica la ecuación 4.1 para seleccionar el modelo que mejor desempeño ha demostrado con los datos. En el contexto de este proyecto, es fundamental no solo evaluar el rendimiento de un modelo a través de métricas cuantitativas sino también visualizar las predicciones para asegurar que el modelo funciona correctamente en un contexto clínico.

Capítulo 5

Resultados

Este capítulo está dedicado a describir las pruebas realizadas para cada modelo y analizar los resultados obtenidos. Primero, se presentan los valores probados en la búsqueda de hiperparámetros y se detallan los valores óptimos encontrados. A continuación, se muestran los resultados de la predicción . Posteriormente, se realiza una comparativa de los modelos. Finalmente, se presentan los resultados obtenidos para cada paciente, incluyendo métricas de evaluación, matrices de confusión, curvas ROC-AUC y mapas de clasificación.

5.1. Resultados de la búsqueda de hiperparámetros

Esta sección está dedicada a mostrar los resultados obtenidos de la búsqueda de hiperparámetros utilizando el módulo descrito en el capítulo anterior. Esta sección se divide en dos apartados: el primero presenta los resultados obtenidos para los modelos de aprendizaje automático, las máquinas de vectores de soporte (SVM) y los bosques aleatorios (*Random Forest*). El segundo apartado detalla los diferentes modelos de redes neuronales probados y los parámetros evaluados.

5.1.1. Modelos de aprendizaje automático

Como se explicó en el capítulo anterior, la búsqueda de hiperparámetros se lleva a cabo utilizando validación cruzada y búsqueda de cuadrícula. El criterio de evaluación se basa en la exactitud (*accuracy*). El módulo emplea una validación cruzada de cinco pliegues (*folds*), dividiendo el conjunto de datos en cinco partes. En cada iteración, uno de estos pliegues se utiliza como conjunto de validación mientras que los otros cuatro se utilizan para entrenar el modelo. El proceso se repite cinco veces, de esta manera cada pliegue se utiliza una vez como conjunto de validación. El resultado final es la media de las exactitudes obtenidas en cada iteración, proporcionando una estimación del rendimiento del modelo. La elección de cinco pliegues representa un

equilibrio entre la precisión de la estimación y el tiempo de computación requerido. Este número de pliegues es una práctica comúnmente aceptada.

5.1.1.1. Hiperparámetros SVM

Se introduce una breve introducción de los parámetros probados:

- *C*: Parámetro de regularización, controla el equilibrio entre maximizar el margen de separación y minimizar el error de clasificación. Un valor bajo de *C* permite un margen más amplio a costa de más errores de clasificación, mientras que un valor alto reduce el margen y busca minimizar errores.
- *Kernel* : Parámetro que define la función de transformación de los datos. Kernel lineal es adecuado para datos linealmente separables mientras que el kernel radial básico es adecuado para manejar relaciones no lineales más complejas.
- *Gamma*: Parámetro que define la influencia de un solo ejemplo de entrenamiento. Un valor alto indica que solo un ejemplo tiene un alcance más corto y específico, mientras que un valor bajo implica un alcance más alto y general.
- *Class weight*: Parámetro que asigna diferentes pesos a las clases para manejar problemas de desequilibrio de clases. Cuando los datos están desbalanceados el modelo puede sesgarse hacia la clase mayoritaria. El parámetro *class weight* ajusta este sesgo asignando un mayor peso a la clase minoritaria y un menor peso a la clase mayoritaria, ayudando al modelo a aprender de una manera equitativa de ambas clases.

Los mejores hiperparámetros obtenidos para el modelo SVM se representan en la tabla 5.1 y obtuvieron una exactitud del 72 % en datos nunca vistos. Los mejores valores serán utilizados en el archivo de configuración que recibe el módulo principal.

Hiperparámetro	Parámetros probados	Mejor valor
C	0.1, 1, 10	1
Kernel	lineal, rbf	rbf
Gamma	1, 0.1, 0.01	0.1
Class weight	Balanced, None	None

Tabla 5.1. Hiperparámetros probados para SVM

5.1.1.2. Hiperparámetros *Random Forest*

Se introduce una breve explicación de los parámetros probados para este modelo:

- Número de estimadores: Parámetro que indica el número de árboles en el bosque. Más árboles conducen a una mejor precisión pero también incrementan el tiempo de computación.

- Profundidad máxima: Limitar la profundidad de los árboles puede ayudar a evitar el sobreajuste. Una mayor profundidad permite aprender patrones más complejos, pero existe el riesgo de de aprender ruido en los datos.

- Número mínimo de muestras para dividir un nodo: Parámetro que controla el número mínimo de muestras necesarias para dividir un nodo interno. Un valor alto puede prevenir que el modelo sufra sobreajuste.

Los mejores hiperparámetros obtenidos para el modelo *Random Forest* se representan en la tabla 5.2 y obtuvieron una exactitud de 77 %. Los mejores valores serán utilizados en el archivo de configuración que recibe el módulo principal.

Hiperparámetro	Parámetros probados	Mejor valor
Número de estimadores	100, 200, 300	300
Profundidad máxima	10, 20, 30	30
Número mínimo de muestras para dividir	2, 5, 10	2

Tabla 5.2. Hiperparámetros probados para RF

5.1.2. Redes neuronales

Los parámetros que utiliza el módulo de búsqueda de mejores hiperparámetros para redes neuronales son los siguientes:

- Capas ocultas: Número y tamaño de las capas ocultas de la red

- Tasa de abandono (*Dropout*): Fracción de neuronas que se van a desactivar aleatoriamente durante el entrenamiento para evitar el sobreajuste.

- Tasa de aprendizaje: Velocidad a la que los pesos de la red se actualizan durante el entrenamiento.

- Tamaño del batch: Número de muestras de entrenamiento utilizadas en una sola actualización de los pesos.

5.1.2.1. Parámetros probados y resultados

En la tabla 5.3 se detallan los hiperparámetros evaluados, incluyendo las capas ocultas, la tasa de abandono (Dropout), el número de épocas, la tasa de aprendizaje (Learning rate o lr) y el tamaño del lote (batch size). Estos hiperparámetros se probaron con el optimizador por defecto *Adam* y la función de pérdida *Cross Entropy Loss*.

Hiperparámetro	Parámetros probados
Capas ocultas	[128], [256,128]
Tasa de abandono	0.3, 0.5
Épocas	150
Tasa de aprendizaje	0.01, 0.001, 0,0005
Tamaño del lote	64, 128, 256

Tabla 5.3. Hiperparámetros probados para redes neuronales

Los resultados obtenidos para cada modelo se presentan en la tabla 5.4. Los modelos muestran buenos resultados en los datos de entrenamiento pero no mantienen la precisión en la predicción, lo que indica que podrían estar sobreajustados y no generalizar bien a datos no vistos. Cuando las clases no están equilibradas, como es en el caso de este proyecto, los modelos pueden sesgarse hacia clases mayoritarias, logrando una alta precisión en el entrenamiento pero fallando en predecir las clases minoritarias.

Modelo	Capas ocultas	Dropout	Épocas	Lr	Batch size	ACC Train	ACC Prediction
SimpleNN	[128]	-	150	0.0005	128	0.832	0.699
ModifiedMLP	[128]	-	150	0.0005	256	0.831	0.688
TunedNN	[256, 128]	0.3	150	0.0005	128	0.835	0.678
ImprovedNN	[256, 128]	0.5	150	0.01	128	0.842	0.667

Tabla 5.4. Resultados hiperparámetros con Adam y CEL

Los resultados obtenidos con las configuraciones de la tabla 5.4 tienen margen de mejora. Por esta razón, se prueba una nueva configuración cambiando el optimizador y la función de pérdida, sin embargo, se mantienen los valores de hiperparámetros que se han presentado en la tabla 5.4. Se ha utilizado el optimizador SGD y la función de pérdida focal o *Focal Loss* y se han obtenido los resultados presentados en la tabla 5.5. Cabe mencionar que se han probado otros optimizadores como *RMSProp* y *AdamW*, pero no se han obtenido mejores resultados que con *Adam* y la función de pérdida *Cross Entropy Loss*, por ello, no se detallan en la memoria.

Modelo	Capas ocultas	Dropout	Épocas	Lr	Batch size	ACC Train	ACC Prediction
SimpleNN	[128]	-	150	0.0005	128	0.799	0.764
ModifiedMLP	[128]	-	150	0.0005	256	0.795	0.736
TunedNN	[256, 128]	0.3	150	0.0005	128	0.821	0.733
ImprovedNN	[256, 128]	0.5	150	0.01	128	0.815	0.745

Tabla 5.5. Resultados con SGD y Focal Loss

Se puede visualizar mejor la comparación entre los resultados de entrenamiento y predicción obtenidos con ambas configuraciones en la figura 5.1, que muestra las exactitudes de los diferentes modelos bajo ambos enfoques.

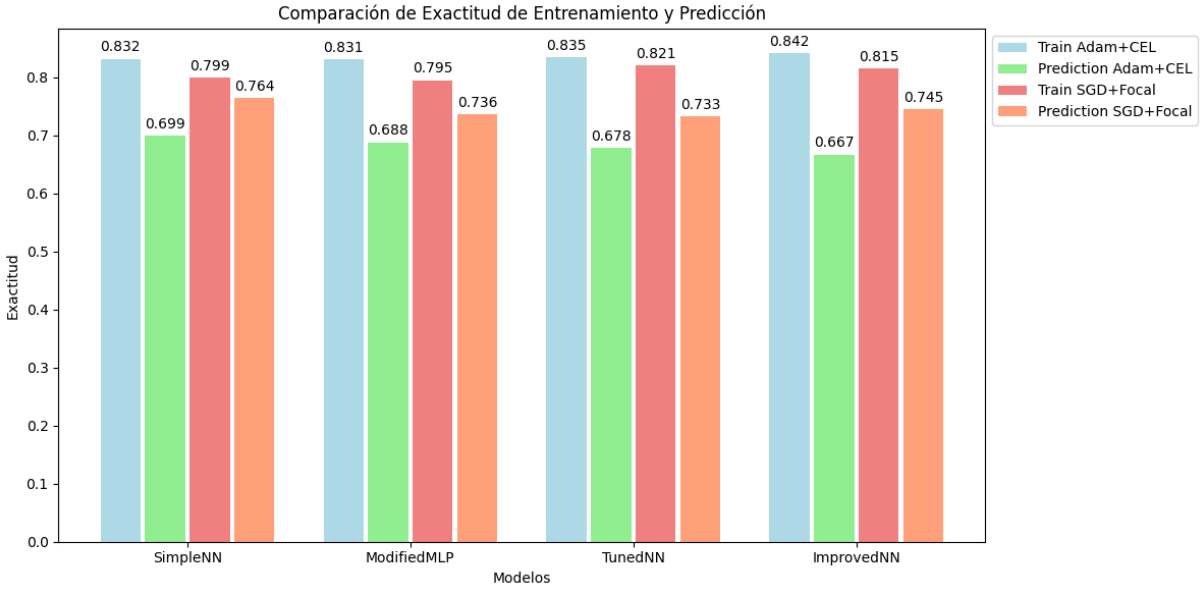


Figura 5.1. Comparación de la exactitud en el entrenamiento y la predicción

En la figura 5.2 se muestra el historial de entrenamiento de los modelos entrenados con el optimizador Adam y la función de pérdida de entropía cruzada o CEL, por sus siglas en inglés. Se observa que SimpleNN y ModifiedMLP muestran una disminución constante en la pérdida y un aumento en la precisión, pero la precisión de validación es menor comparada con la del entrenamiento, indicando un posible sobreajuste. En TunedNN e ImprovedNN, la pérdida disminuye más lentamente que en los otros modelos y la precisión se estabiliza después de un número mayor de épocas.

En la figura 5.3 se muestran las curvas de aprendizaje de los modelos entrenados con el optimizador SGD y la función de pérdida focal. Se observa que SimpleNN y ModifiedMLP muestran una mejora constante en la precisión. Sin embargo, se observa una convergencia más suave y menos fluctuaciones en la pérdida, indicando una mejor estabilidad. En TunedNN e ImprovedNN, la pérdida inicial se reduce más rápido y la precisión se estabiliza más temprano, sugiriendo un aprendizaje más efectivo.

El cambio de optimizador y función de pérdida provocan una optimización más estable. Las curvas de pérdida con SGD son más suaves y muestran menos fluctuaciones, indicando un proceso de entrenamiento más estable y posiblemente un mejor mínimo global alcanzado. La función de pérdida focal parece ayudar a los modelos a centrarse en clases minoritarias, mejorando la precisión general en datos de validación y reduciendo el sobreajuste observado con CEL.

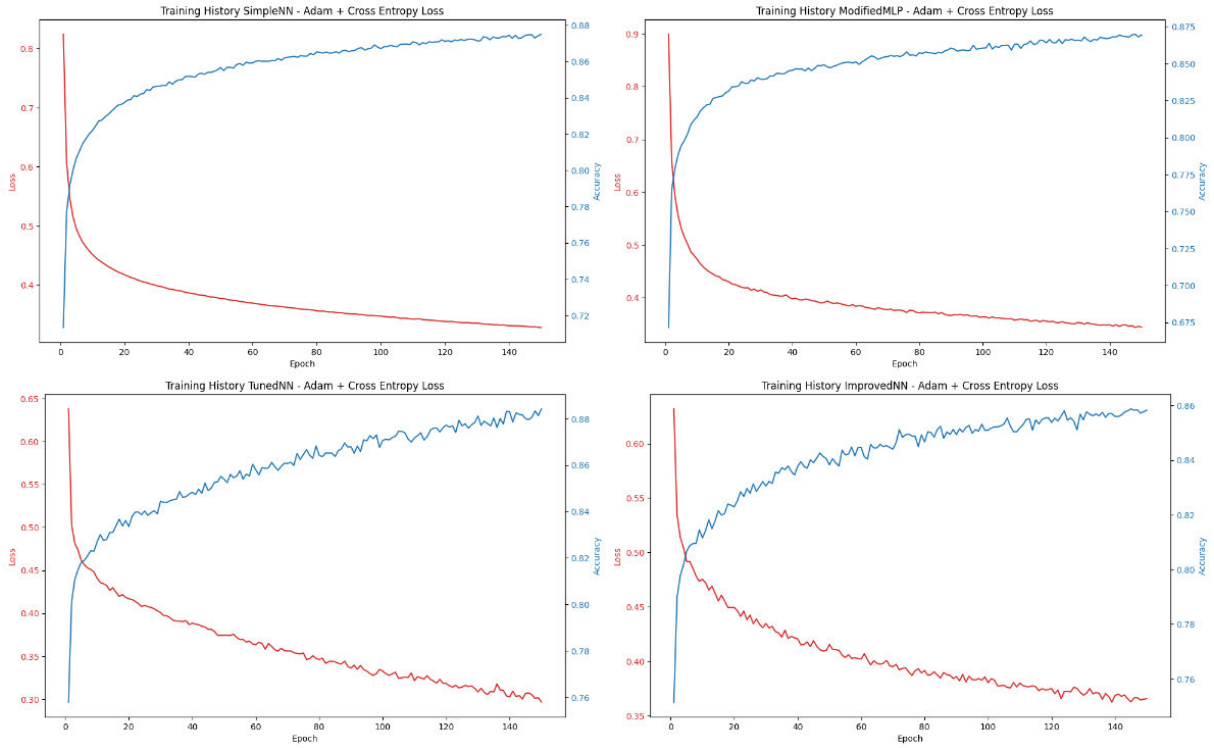


Figura 5.2. Historial entrenamiento con Adam y CEL

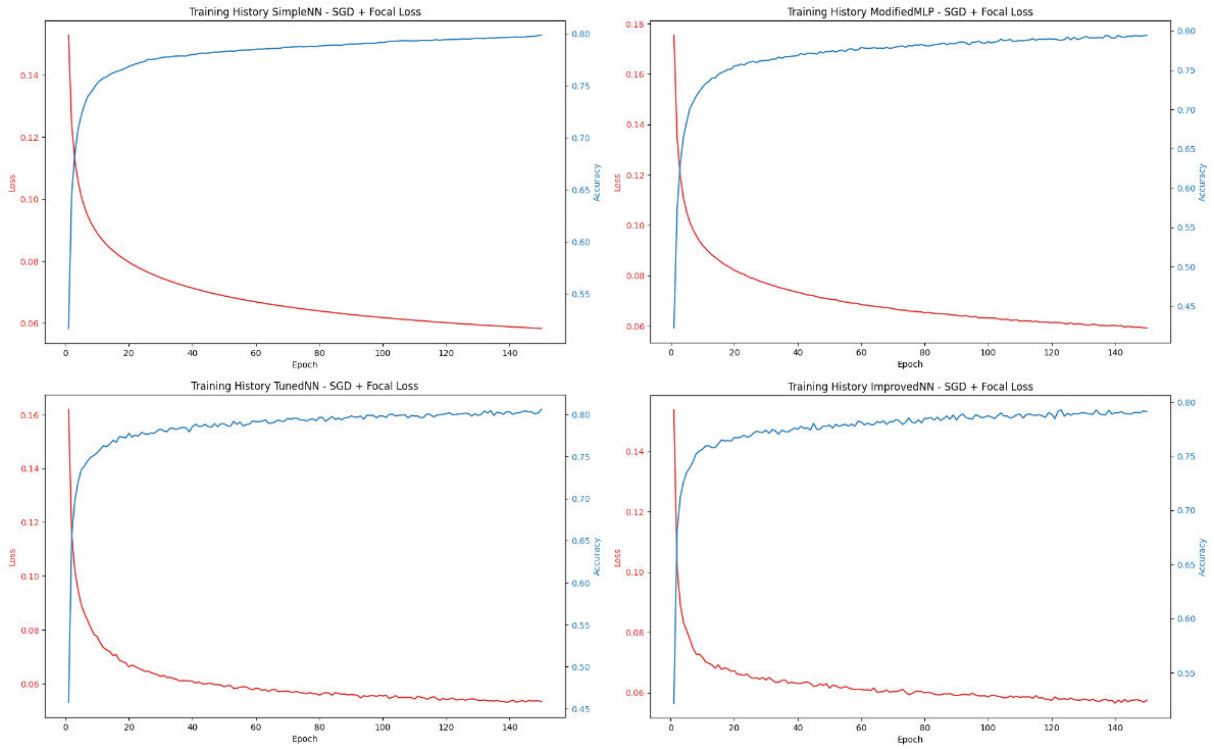


Figura 5.3. Historial entrenamiento con SGD y Focal Loss

En esta sección, se han descrito todas las pruebas realizadas y los resultados obtenidos de la búsqueda de hiperparámetros. Se concluye la sección con un resumen de los mejores hiperparámetros encontrados para cada modelo, que se escogerán para utilizarse en el módulo principal de la herramienta:

- Máquina de vectores de soporte (SVM). Los mejores hiperparámetros :
  - $C = 1$
  - $kernel = rbf$
  - $gamma = 0{,}1$
  - $classweight = None$

  Logrando una exactitud de 72 %.

- Bosques aleatorios (*Random Forest*). Los mejores hiperparámetros:
  - Número de estimadores: 300
  - Profundidad máxima: 30
  - Mínimo de muestras para dividir un nodo: 2

  Obteniendo una exactitud de 77 %.

- Redes neuronales. Se han probado diferentes modelos con diferentes hiperparámetros, diferentes optimizadores y diferentes funciones de pérdida. Se ha concluido que Focal Loss con SGD funciona mejor manejando el desbalanceo de clases. Los mejores hiperparámetros escogidos se muestran en la tabla 5.5 en la que SimpleNN logra la mejor exactitud con 76 %.

5.2. Predicción

En esta sección se presentan los resultados de la predicción del módulo principal. Los modelos evaluados son SVM, *Random Forest* y las redes neuronales. Los mejores hiperparámetros se determinaron tras una búsqueda exhaustiva.

La tabla 5.6 muestra las métricas de la predicción. Las métricas analizadas son exactitud (ACC), precisión (PRE), recuperación (REC), puntuación F1 (F1-SCORE), área bajo la curva ROC (ROC-AUC).

Modelo	ACC	PRE	REC	F1-SCORE	ROC-AUC
SVM	0.709	0.739	0.708	0.706	0.746
Random Forest	0.773	0.781	0.773	0.757	0.735
SimpleNN	0.764	0.770	0.763	0.751	0.712
ModifiedMLP	0.736	0.777	0.735	0.735	0.735
TunedNN	0.733	0.782	0.732	0.730	0.705
ImprovedNN	0.745	0.738	0.745	0.733	0.686

Tabla 5.6. Métricas de la predicción

Con el objetivo de presentar de manera más visual los datos de la tabla 5.6, se introduce la figura 5.4.

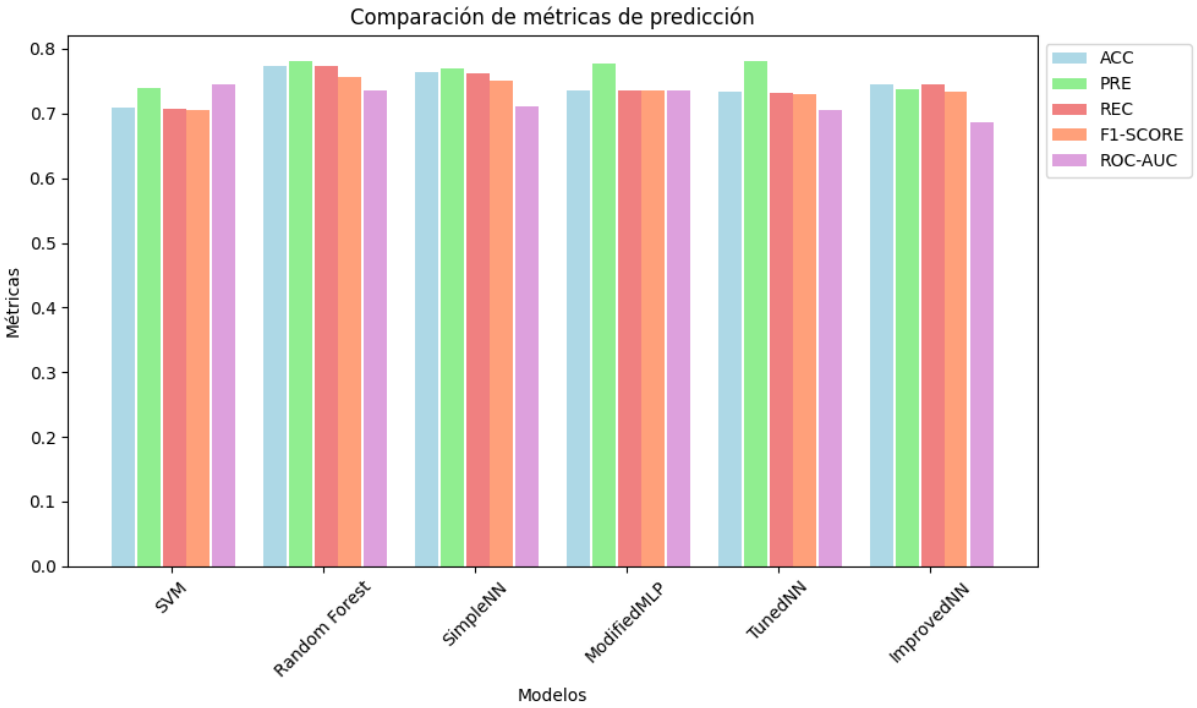


Figura 5.4. Comparación de las métricas de predicción

Los resultados de la predicción indican que el modelo *Random Forest* muestra el mejor rendimiento general y ofrece un equilibrio entre todas las métricas. Los valores de exactitud 0,773 y puntuación F1 0,757, sugiere que maneja bien tanto las predicciones correctas como las incorrectas en datos no vistos. Esto se debe a su capacidad para manejar mejor la variabilidad de los datos y su robustez frente al sobreajuste.

El modelo SVM, tiene una predicción aceptable por sus valores de exactitud 0,709 y ROC-AUC 0,746, lo que indica un buen equilibrio entre TPR Y FPR.

Las redes neuronales ImprovedNN y SimpleNN muestran un rendimiento competitivo con una exactitud de 0,745 y 0,764 respectivamente, pero sin superar a *Random Forest*. ModifiedMLP

y TunedNN, muestran un rendimiento aceptable. La capacidad de estas redes para aprender representaciones complejas es ventajosa, pero puede no ser suficiente para superar la capacidad de generalización de *Random Forest*.

5.2.1. Comparación de los modelos

En este apartado, se comparan los modelos basados en su métricas de predicción presentadas en la tabla 5.6. Las puntuaciones de cada modelo se calcularon utilizando la ecuación 4.1, asignando mayor importancia a las métricas *F1-Score* y ROC-AUC . Las puntuaciones obtenidas se presentan en la tabla 5.7. Este enfoque se emplea para identificar el modelo que mejor se ajusta tanto a los datos como al entorno. El modelo *Random Forest* obtiene la mejor puntuación con 0,758 seguido de SimpleNN con 0,745. Esto sugiere que *Random Forest* es el modelo más equilibrado.

Modelo	Puntuación
SVM	0.722
Random Forest	0.758
SimpleNN	0.745
ModifiedMLP	0.739
TunedNN	0.729
ImprovedNN	0.723

Tabla 5.7. Puntuaciones obtenidas de cada modelo

Para la siguiente sección, se presentarán las predicciones de los modelos de aprendizaje automático SVM, *Random Forest*, y las predicciones de los modelos de redes neuronales SimpleNN e ImprovedNN. Aunque ModifiedMLP y TunedNN consiguieron mejor puntuación que ImprovedNN en la tabla 5.7, se ha seleccionado ImprovedNN para mostrar el potencial de una red neuronal más sofisticada que incluye técnicas avanzadas como *dropout* y *batch normalization*. De este modo, se podrá comparar los resultados gráficos de un modelo simple, que proporciona una referencia base, con un modelo de red neuronal complejo.

5.3. Predicción de cada paciente

Esta sección está dedicada a presentar los resultados de las predicciones para cada paciente. Las predicciones se han realizado utilizando un conjunto de datos de validación compuesto por tres pacientes. En la figura 5.5 se presenta la imagen RGB original del paciente junto con el mapa *ground truth*, que representa la clasificación real de las diferentes regiones del cerebro, y el mapa de clasificación predicho generado por cada modelo. Este mapa visualiza cómo cada modelo ha clasificado las diferentes regiones del cerebro del paciente, permitiendo una comparación directa con el mapa *ground truth*.

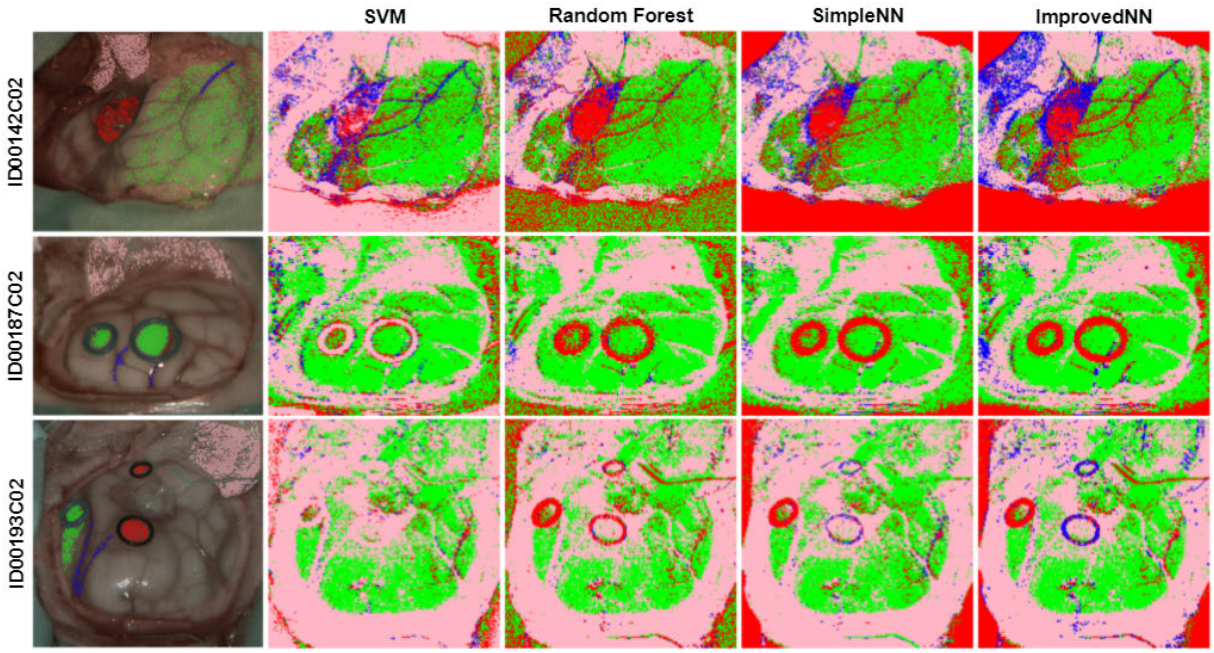


Figura 5.5. Mapas de clasificación por cada modelo

En la figura 5.5, para el primer paciente, se observa que los modelos *Random Forest* y SimpleNN son los más capaces en delimitar la clase tumor. ImprovedNN no consigue delimitar completamente la clase tumor, pero sí la clase vena. SVM es el que peor delimita el tumor pero el que mejor delimita la vena. Todos los modelos delimitan bien las clases duramadre y sano. Para el segundo paciente, todos los modelos delimitan bien la clase sano, pero tienen problemas con la clase vena y duramadre. Para el tercer paciente, todos los modelos delimitan y clasifican bien la clase sano. Sin embargo, la clase tumor presenta un área de dificultad tanto en la delimitación como en la clasificación para todos los modelos.

Para cada paciente, se presenta un análisis del rendimiento de los distintos modelos:

- Paciente ID00142C02. En la tabla 5.8 se presentan las métricas de predicción, destacando el modelo con mejor rendimiento global, *Random Forest*, con una excelente capacidad de discriminación entre clases. SimpleNN muestra un rendimiento ligeramente superior a ImprovedNN en términos de exactitud y puntuación F1. SVM presenta problemas de exactitud y recuperación, lo que sugiere dificultades con falsos negativos.

  Las matrices de confusión A.1 indican que *Random Forest* presenta mayor cantidad de verdaderos positivos y menor cantidad de falsos negativos y positivos, especialmente en las clases sano, tumor y duramadre. SimpleNN e ImprovedNN tienen buen rendimiento, aunque SimpleNN presenta más falsos negativos que *Random Forest* e ImprovedNN tiene dificultades con la clase vena. SVM tiene buen rendimiento en duramadre y sano, pero una tasa de falsos positivos en tumor, lo que es preocupante en aplicaciones médicas. Las curvas ROC-AUC A.2, confirman la superioridad en la discriminación de todas las clases

Modelo	ACC	PRE	REC	SCORE F1	ROC-AUC
SVM	0.775	0.848	0.774	0.801	0.891
Random Forest	0.837	0.875	0.836	0.847	0.939
SimpleNN	0.820	0.870	0.8198	0.837	0.871
ImprovedNN	0.805	0.861	0.804	0.825	0.871

Tabla 5.8. Métricas de la predicción paciente 142

de *Random Forest*.

- Paciente ID00187C02. La tabla 5.9 muestra que SVM tiene una buena precisión y recuperación, con la mejor ROC-AUC indicando mejor capacidad de discriminación. *Random Forest* muestra el mejor rendimiento en casi todas las métricas excepto en ROC-AUC que es más baja. SimpleNN muestra un rendimiento cercano a *Random Forest* pero con una ROC-AUC también baja. ImprovedNN muestra la menor ROC-AUC, indicando dificultades en la discriminación entre clases.

  Las matrices de confusión A.3 indican que *Random Forest* e ImprovedNN son los mejores en detectar las clases sano y duramadre, pero tienen problemas con la clase vena. SVM y SimpleNN muestran un buen rendimiento en las clases sano y duramadre, pero SVM presenta mas dificultades con la clase vena. Las curvas ROC-AUC A.4, confirman que *Random Forest* tiene la mejor discriminación para la clase sano, pero menor rendimiento para vena y duramadre, mientras que ImprovedNN y SimpleNN presentan dificultades similares.

Modelo	ACC	PRE	REC	SCORE F1	ROC-AUC
SVM	0.784	0.858	0.783	0.789	0.658
Random Forest	0.851	0.852	0.850	0.851	0.581
SimpleNN	0.843	0.894	0.842	0.841	0.564
ImprovedNN	0.834	0.827	0.833	0.828	0.488

Tabla 5.9. Métricas de la predicción paciente 187

- Paciente ID00193C02. Las métricas presentadas en la tabla 5.10, se observa que SVM tiene una capacidad de discriminación aceptable pero su exactitud y puntuación F1 son las mas bajas. *Random Forest* tiene la mayor exactitud y buena puntuación F1 pero su ROC-AUC es ligeramente inferior a SimpleNN, que presenta un buen rendimiento global con alta exactitud y la mayor ROC-AUC. ImprovedNN muestra una buena capacidad de discriminación, su exactitud y puntuación F1 son inferiores a las de *Random Forest* y SimpleNN.

  Las matrices de confusión A.5 indican que *Random Forest* e ImprovedNN clasifican bien las clases sano y duramadre, mientras que SVM tiene dificultades con la clase tumor y vena. SimpleNN destaca en la clasificación de la clase sano. Las curvas ROC-AUC A.6, confirman que SimpleNN y *Random Forest* tienen la mejor capacidad de discriminación global, especialmente para la clase sano. ImprovedNN es excelente para la clase vena, y

Modelo	ACC	PRE	REC	SCORE F1	ROC-AUC
SVM	0.568	0.510	0.566	0.528	0.689
Random Forest	0.631	0.615	0.630	0.573	0.684
SimpleNN	0.630	0.547	0.629	0.576	0.701
ImprovedNN	0.595	0.526	0.593	0.547	0.700

Tabla 5.10. Métricas de la predicción paciente 193

SVM es aceptable para sano y vena, pero débil para tumor y duramadre.

Después de analizar los resultados de la predicción y las métricas para los tres pacientes mencionados, se puede analizar el desempeño de los diferentes modelos. *Random Forest* ha mostrado la mayor exactitud en comparación con otros modelos. Muestra una capacidad de discriminación alta, especialmente para la clase sano. Destaca por su alta discriminación para la clase sano y tumor, generalmente ha obtenido la menor cantidad de falsos negativos en la clase duramadre. Tiene algunos problemas en la discriminación de la clase vena aunque es mejor que algunos modelos en ciertos pacientes.

SimpleNN ha mostrado una exactitud muy cercana a *Random Forest*. Muestra la mayor ROC-AUC en varios casos indicando una excelente capacidad de discriminación, especialmente en la clase sano y la clase tumor. Tiene mejores resultados de la clase vena en comparación con otros modelos. Sin embargo, muestra áreas de confusión para duramadre y tumor. ImprovedNN tiene una excelente ROC-AUC para la clase vena pero mas débil en tumor y duramadre. SVM ha obtenido la menor exactitud de todos los modelos evaluados. Tiene una capacidad de discriminación razonable para la clase sano pero dificultades significativas para la clase tumor y vena. Además,tiene mayor cantidad de falsos negativos en corporación con otros modelos.

Aunque estas métricas no son exactas puesto que solo se han comparado con los píxeles etiquetados en el mapa *ground truth*, se puede concluir que *Random Forest* muestra el mejor rendimiento global, es el más robusto y confiable para la clasificación de este tipo de imágenes. SimpleNN es una excelente alternativa a *Random Forest*. ImprovedNN y SVM, pueden ser útiles como parte de un conjunto de modelos para mejorar la robustez del sistema.

Capítulo 6

Presupuesto

En este capítulo se detallan los gastos provenientes del desarrollo de este proyecto. El presupuesto se divide en tres bloques: los recursos humanos, los recursos hardware y los recursos software.

Los recursos humanos comprenden las horas dedicadas a realizar este proyecto. Tal y como se especificó en el anteproyecto, se resume en la tabla 6.1.

Personal	Horas	Salario	Coste
Sueldo del alumno	324	12€/h	3.888€
Tutor y director	40	30€/h	1.200€
		Total	**5.088€**

Tabla 6.1. Presupuesto recursos humanos

Los recursos hardware se resumen en la tabla 6.2:

Material	Descripción	Coste
Ordenador de sobremesa	Microsoft Windows 11 de 64 bits y 8GB RAM	520€
	Total	**520€**

Tabla 6.2. Presupuesto recursos hardware

Los recursos software se resumen en la tabla 6.3:

Material	Descripción	Coste
Visual Studio Code	Editor de código fuente	0€
Python	Intérprete y bibliotecas	0€
Overleaf	Plataforma de edición de documentos LaTex	0€
	Total	**0€**

Tabla 6.3. Presupuesto recursos software

En la tabla 6.4 se resume el coste total del proyecto en base a los recursos humanos, los recursos hardware y los recursos software descritos anteriormente.

Descripción	Coste
Recursos humanos	5.088€
Recursos hardware	520€
Recursos software	0€
Total	**5.608€**

Tabla 6.4. Presupuesto total

Capítulo 7

Impacto del proyecto

En este capítulo se exponen las implicaciones sociales, de salud y seguridad, económicas, ambientales, tecnológicas e industriales del trabajo realizado, así como su contribución a los objetivos de desarrollo sostenible (ODS) [53].

7.1. Implicaciones

- Implicaciones sociales. La utilización de técnicas de clasificación de imágenes hiperespectrales médicas ayudan a mejorar la precisión y velocidad con la que los médicos pueden delimitar y predecir la presencia de tumores en el tejido cerebral. Supone un impacto directo en la calidad de vida de los pacientes al permitir una intervención médica temprana y un tratamiento más efectivo no invasivo. La implementación de modelos de clasificación podría llegar a democratizar el acceso a diagnósticos avanzados en regiones con recursos médicos limitados reduciendo desigualdades en atención sanitaria.
- Implicaciones de salud y seguridad. El desarrollo de estos modelos puede ayudar a reducir errores de delimitación de tumores. Una mejor detección de las características tumorales contribuye a un mejor manejo de las mismas. De esta manera, se optimizan los recursos sanitarios reduciendo la carga en los sistemas de salud y disminuyendo la necesidad de procedimientos invasivos a la hora de diagnosticar esta enfermedad, por lo que se reducen riesgos asociados a estos procedimientos [54].
- Implicaciones ambientales. La optimización en recursos médicos con diagnósticos más rápidos y precisos reduce la huella ambiental en el sector de salud. Se precisan menos cantidad de pruebas y tratamientos lo que supone un menor consumo de material y energía [55].
- Implicaciones económicas. La aplicación de modelos de clasificación puede tener un impacto económico positivo reduciendo costes asociados a diagnósticos incorrectos o tardíos [56]. La implantación de estas tecnologías puede generar oportunidades de empleo.

- Implicaciones tecnológicas e industriales. El avance en la clasificación de este tipo de imágenes, representa un salto en el campo del aprendizaje automático aplicados a la medicina. Este proyecto contribuye al desarrollo de herramientas de apoyo para los médicos y promueve la investigación y desarrollo continuo del campo. Las industrias se pueden aprovechar de estas innovaciones mediante la mejora de sus productos y servicios manteniéndose competitivas en el sector.

7.2. Contribución a los Objetivos de Desarrollo Sostenible

El proyecto contribuye a varios ODS:

- ODS 3: Salud y Bienestar [57]. Con las mejora de herramientas de apoyo en la delimitación y predicción de tejido tumoroso se promueve la salud y bienestar de todas las personas.
- ODS 4 [58] : Educación de calidad. Se fomenta la formación en tecnologías avanzadas promoviendo la educación en el sector científico y tecnológico.
- ODS 9 [59]: Industria, Innovación e Infraestructura. Se fomenta la innovación tecnológica y la infraestructura en el sector médico.
- ODS 10 [60]: Reducción de desigualdades. Se proporciona una herramienta de análisis accesible que genera un informe automatizado, ayudando así, a reducir las desigualdades en el acceso a la atención médica.
- ODS 12 [61]: Producción y consumo responsable. Se optimiza el uso de recursos en el sector de la salud promoviendo prácticas sostenibles.
- ODS 17 [62]: Alianza para lograr objetivos. Colaboración entre instituciones académicas, hospitales y empresas tecnológicas fortalecen las alianzas para alcanzar estos objetivos.

Capítulo 8

Conclusiones

En este último capítulo, se presenta una visión general del trabajo realizado, comenzando con el problema propuesto y finalizando con la solución planteada. El proyecto se ha basado en la comparación de diferentes modelos de aprendizaje automático, en concreto, la máquina de vector de soporte, los bosques aleatorios y diversos modelos de redes neuronales, para la clasificación de imágenes hiperespectrales. Este trabajo se ha centrado en la delimitación de distintas zonas de tejido cerebral, evaluando la precisión de cada algoritmo. A partir de esta evaluación, se ha seleccionado el modelo que ha mostrado mejor rendimiento, en términos de precisión y eficacia en la clasificación de las zonas de tejido.

Los resultado obtenidos mostraron que, en general, los cuatro modelos finales analizados, son capaces de clasificar las zonas de tejido con cierta eficacia, aunque cada uno tiene sus fortalezas y debilidades. *Random Forest* ha mostrado el mejor rendimiento global considerando tanto las métricas cualitativas (mapas de clasificación) como cuantitativas (matrices de confusión y curvas ROC-AUC), que lo hacen el modelo más robusto y confiable. El segundo mejor modelo obtenido ha sido SimpleNN que ofrece un rendimiento competitivo y una excelente alternativa a *Random Forest*. ImprovedNN ha obtenido un rendimiento más bajo pero ha mostrado fortalezas en discriminación de ciertas clases, por lo que puede ser útil en combinaciones con otros modelos. Finalmente, SVM ha obtenido el rendimiento más bajo, sin embargo, todavía puede ser útil combinándolo con otros modelos con el objetivo de mejorar la robustez del sistema.

A partir de los resultados, se puede derivar que la validez del sistema es prometedora, pero no definitiva. Los modelos evaluados tienen margen de mejora en la precisión de los tejidos cerebrales, pero se han detectado aspectos que deben ser abordados en trabajos futuros. Un ejemplo de ello, es la variabilidad del rendimiento entre diferentes pacientes, que sugiere que los modelos podrían beneficiarse de una mayor cantidad de datos de entrenamiento y una mejor optimización de hiperparámetros.

Para futuras lineas de trabajo, se propone lo siguiente:

- Optimización de hiperparámetros. Realizar una búsqueda más exhaustiva de hiperparámetros, especialmente para los modelos de red neuronal.
- Aumento de datos para crear más variabilidad en el conjunto de entrenamiento y utilización de más pacientes en la predicción para obtener una mejor evaluación de la capacidad de generalización de cada modelo.
- Combinación de varios modelos en un enfoque de aprendizaje en conjunto para aprovechar las fortalezas de cada modelo.
- Desarrollo y creación de una interfaz de programación de aplicaciones (API) que permita a los sistemas médicos integrar el mejor modelo desarrollado para realizar predicciones en tiempo real. Incluyendo medidas de monitorización que permitan evaluar el rendimiento del modelo y se realicen ajustes necesarios según una retroalimentación recibida.
- Investigación continua mediante la exploración de nuevas arquitecturas de redes neuronales.

Referencias

[1] I. J. Gerard, M. Kersten-Oertel, K. Petrecca, D. Sirhan, J. A. Hall y D. L. Collins, "Brain shift in neuronavigation of brain tumors: A review", *Medical Image Analysis*, vol. 35, págs. 403-420, 2017, ISSN: 1361-8415. DOI: https://doi.org/10.1016/j.media.2016.08.007. [en línea]. Disponible en: https://www.sciencedirect.com/science/article/pii/S1361841516301566.

[2] J. Yoon, "Hyperspectral Imaging for Clinical Applications", *BioChip Journal*, vol. 16, n.º 1, págs. 1-12, 2022, ISSN: 2092-7843. DOI: 10.1007/s13206-021-00041-0. [en línea]. Disponible en: https://doi.org/10.1007/s13206-021-00041-0.

[3] F. Manni, F. van der Sommen, H. Fabelo et al., "Hyperspectral imaging for glioblastoma surgery: Improving tumor identification using a deep spectral-spatial approach", *Sensors*, vol. 20, n.º 23, pág. 6955, 2020.

[4] D. T. Dicker, J. Lerner, P. V. Belle et al., "Differentiation of normal skin and melanoma using high resolution hyperspectral imaging", *Cancer Biology & Therapy*, vol. 5, n.º 8, págs. 1033-1038, 2006, PMID: 16931902. DOI: 10.4161/cbt.5.8.3261. eprint: https://www.tandfonline.com/doi/pdf/10.4161/cbt.5.8.3261. [en línea]. Disponible en: https://www.tandfonline.com/doi/abs/10.4161/cbt.5.8.3261.

[5] Y. H. El-Sharkawy, "Advancements in non-invasive hyperspectral imaging: Mapping blood oxygen levels and vascular health for clinical and research applications", *Vascular Pharmacology*, vol. 155, pág. 107 380, 2024, ISSN: 1537-1891. DOI: https://doi.org/10.1016/j.vph.2024.107380. [en línea]. Disponible en: https://www.sciencedirect.com/science/article/pii/S153718912400106X.

[6] G. Edelman, E. Gaston, T. van Leeuwen, P. Cullen y M. Aalders, "Hyperspectral imaging for non-contact analysis of forensic traces", *Forensic Science International*, vol. 223, n.º 1, págs. 28-39, 2012, ISSN: 0379-0738. DOI: https://doi.org/10.1016/j.forsciint.2012.09.012. [en línea]. Disponible en: https://www.sciencedirect.com/science/article/pii/S0379073812004495.

[7] M. Milanic, L. A. Paluchowski y L. L. Randeberg, "Hyperspectral imaging for detection of arthritis: feasibility and prospects", *Journal of Biomedical Optics*, vol. 20, n.º 9,

pág. 096 011, 2015. DOI: 10.1117/1.JBO.20.9.096011. [en línea]. Disponible en: https://doi.org/10.1117/1.JBO.20.9.096011.

[8] E. L. Wisotzky, F. C. Uecker, P. Arens, S. Dommerich, A. Hilsmann y P. Eisert, "Intraoperative hyperspectral determination of human tissue properties", *Journal of Biomedical Optics*, vol. 23, n.º 9, pág. 091 409, 2018. DOI: 10.1117/1.JBO.23.9.091409. [en línea]. Disponible en: https://doi.org/10.1117/1.JBO.23.9.091409.

[9] Y. Garini, I. T. Young y G. McNamara, "Spectral imaging: Principles and applications", *Cytometry Part A*, vol. 69A, n.º 8, págs. 735-747, 2006. DOI: https://doi.org/10.1002/cyto.a.20311. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cyto.a.20311. [en línea]. Disponible en: https://onlinelibrary.wiley.com/doi/abs/10.1002/cyto.a.20311.

[10] G. ElMasry y D.-W. Sun, "CHAPTER 1 - Principles of Hyperspectral Imaging Technology", en *Hyperspectral Imaging for Food Quality Analysis and Control*, D.-W. Sun, ed., San Diego: Academic Press, 2010, págs. 3-43, ISBN: 978-0-12-374753-2. DOI: https://doi.org/10.1016/B978-0-12-374753-2.10001-2. [en línea]. Disponible en: https://www.sciencedirect.com/science/article/pii/B9780123747532100012.

[11] M. E. Martin, K. Wabuyele Musundi B.and Chen, P. Kasili et al., "Development of an Advanced Hyperspectral Imaging (HSI) System with Applications for Cancer Detection", *Annals of Biomedical Engineering*, vol. 34, n.º 6, págs. 1061-1068, 2006. DOI: 10.1007/s10439-006-9121-9. [en línea]. Disponible en: https://doi.org/10.1007/s10439-006-9121-9.

[12] S. V. Panasyuk, S. Yang, D. V. Faller et al., "Medical hyperspectral imaging to facilitate residual tumor identification during surgery", *Cancer Biology & Therapy*, vol. 6, n.º 3, págs. 439-446, 2007, PMID: 17374984. DOI: 10.4161/cbt.6.3.4018. eprint: https://doi.org/10.4161/cbt.6.3.4018. [en línea]. Disponible en: https://doi.org/10.4161/cbt.6.3.4018.

[13] L. M. Dale, A. Thewis, C. Boudry et al., "Hyperspectral Imaging Applications in Agriculture and Agro-Food Product Quality and Safety Control: A Review", *Applied Spectroscopy Reviews*, vol. 48, n.º 2, págs. 142-159, 2013. DOI: 10.1080/05704928.2012.705800. eprint: https://doi.org/10.1080/05704928.2012.705800. [en línea]. Disponible en: https://doi.org/10.1080/05704928.2012.705800.

[14] J. Ma, D.-W. Sun, H. Pu, J.-H. Cheng y Q. Wei, "Advanced Techniques for Hyperspectral Imaging in the Food Industry: Principles and Recent Applications", *Annual Review of Food Science and Technology*, vol. 10, n.º Volume 10, 2019, págs. 197-220, 2019, ISSN: 1941-1421. DOI: https://doi.org/10.1146/annurev-food-032818-121155. [en línea]. Disponible en: https://www.annualreviews.org/content/journals/10.1146/annurev-food-032818-121155.

[15] Y. Liu, H. Pu y D.-W. Sun, "Hyperspectral imaging technique for evaluating food quality and safety during various processes: A review of recent applications", *Trends in Food Science & Technology*, vol. 69, págs. 25-35, 2017, ISSN: 0924-2244. DOI: https://doi.org/10.1016/j.tifs.2017.08.013. [en línea]. Disponible en: https://www.sciencedirect.com/science/article/pii/S0924224416305052.

[16] D. Wu, S. Wang, N. Wang et al., "Application of Time Series Hyperspectral Imaging (TS-HSI) for Determining Water Distribution Within Beef and Spectral Kinetic Analysis During Dehydration", *Food and Bioprocess Technology*, vol. 6, n.º 11, págs. 2943-2958, nov. de 2013, ISSN: 1935-5149. DOI: 10.1007/s11947-012-0928-0. [en línea]. Disponible en: https://doi.org/10.1007/s11947-012-0928-0.

[17] C. Rotaru, T. Graf y J. Zhang, "Color image segmentation in HSI space for automotive applications", *Journal of Real-Time Image Processing*, vol. 3, n.º 4, págs. 311-322, dic. de 2008, ISSN: 1861-8219. DOI: 10.1007/s11554-008-0078-9. [en línea]. Disponible en: https://doi.org/10.1007/s11554-008-0078-9.

[18] R. Sarić, V. D. Nguyen, T. Burge et al., "Applications of hyperspectral imaging in plant phenotyping", *Trends in Plant Science*, vol. 27, n.º 3, págs. 301-315, 2022, ISSN: 1360-1385. DOI: https://doi.org/10.1016/j.tplants.2021.12.003. [en línea]. Disponible en: https://www.sciencedirect.com/science/article/pii/S1360138521003460.

[19] A. Kumar, S. Saxena, S. Shrivastava, V. Bharti, U. Kumar y K. Dhama, "Hyperspectral Imaging (HSI): Applications in Animal and Dairy Sector", *Journal of Experimental Biology and Agricultural Sciences*, vol. 4, págs. 448-461, jul. de 2016. DOI: 10.18006/2016.4(4).448.461.

[20] G. Lu y B. Fei, "Medical hyperspectral imaging: a review", *Journal of Biomedical Optics*, vol. 19, n.º 1, pág. 010 901, 2014. DOI: 10.1117/1.JBO.19.1.010901. [en línea]. Disponible en: https://doi.org/10.1117/1.JBO.19.1.010901.

[21] A. Roman-Gonzalez y N. I. Vargas-Cuentas, "Análisis de imágenes hiperespectrales", *Revista Ingenieria & Desarrollo*, vol. Año 9, n.º N° 35, págs. 14-17, sep. de 2013. [en línea]. Disponible en: https://hal.science/hal-00935014.

[22] B. Fei, "Chapter 3.6 - Hyperspectral imaging in medical applications", en *Hyperspectral Imaging*, ép. Data Handling in Science and Technology, J. M. Amigo, ed., vol. 32, Elsevier, 2019, págs. 523-565. DOI: https://doi.org/10.1016/B978-0-444-63977-6.00021-3. [en línea]. Disponible en: https://www.sciencedirect.com/science/article/pii/B9780444639776000213.

[23] A. Esteva, B. Kuprel, R. A. Novoa et al., "Dermatologist-level classification of skin cancer with deep neural networks", *Nature*, vol. 542, n.º 7639, págs. 115-118, feb. de 2017.

[24] A. S. andMatthew J. Muckley, K. Sinha, F. Shamout et al., "COVID-19 Prognosis via Self-Supervised Representation Learning and Multi-Image Prediction", *CoRR*, vol. abs/2101.04909, 2021. arXiv: 2101.04909. [en línea]. Disponible en: https://arxiv.org/abs/2101.04909.

[25] MathWorks. "Máquinas de Vectores de Soporte (SVM)". (), [en línea]. Disponible en: https://es.mathworks.com/discovery/support-vector-machine.html [
Consulta: 07 jun. 2024].

[26] Y. Liu, Y. Wang y J. Zhang, "New Machine Learning Algorithm: Random Forest", en *Information Computing and Applications*, B. Liu, M. Ma y J. Chang, eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, págs. 246-252, ISBN: 978-3-642-34062-8.

[27] K. VijiyaKumar, B. Lavanya, I. Nirmala y S. S. Caroline, "Random Forest Algorithm for the Prediction of Diabetes", en *2019 IEEE International Conference on System, Computation, Automation and Networking (ICSCAN)*, 2019, págs. 1-5. DOI: 10.1109/ICSCAN.2019.8878802.

[28] S. Xuan, G. Liu, Z. Li, L. Zheng, S. Wang y C. Jiang, "Random forest for credit card fraud detection", en *2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC)*, 2018, págs. 1-6. DOI: 10.1109/ICNSC.2018.8361343.

[29] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms", *Pattern Recognition*, vol. 30, n.º 7, págs. 1145-1159, 1997, ISSN: 0031-3203. DOI: https://doi.org/10.1016/S0031-3203(96)00142-2. [en línea]. Disponible en: https://www.sciencedirect.com/science/article/pii/S0031320396001422.

[30] X. Wang, Y. Liu y H. Xin, "Bond strength prediction of concrete-encased steel structures using hybrid machine learning method", *Structures*, vol. 32, págs. 2279-2292, sep. de 2021. DOI: 10.1016/j.istruc.2021.04.018.

[31] D. Hendrycks y K. Gimpel, "Bridging Nonlinearities and Stochastic Regularizers with Gaussian Error Linear Units", *CoRR*, vol. abs/1606.08415, 2016. arXiv: 1606.08415. [en línea]. Disponible en: http://arxiv.org/abs/1606.08415.

[32] P. Ramachandran, B. Zoph y Q. V. Le, "Searching for Activation Functions", *CoRR*, vol. abs/1710.05941, 2017. arXiv: 1710.05941. [en línea]. Disponible en: http://arxiv.org/abs/1710.05941.

[33] A. Demirkaya, J. Chen y S. Oymak, "Exploring the Role of Loss Functions in Multiclass Classification", en *2020 54th Annual Conference on Information Sciences and Systems (CISS)*, 2020, págs. 1-5. DOI: 10.1109/CISS48834.2020.1570627167.

[34] T. Lin, P. Goyal, R. B. Girshick, K. He y P. Dollár, "Focal Loss for Dense Object Detection", *CoRR*, vol. abs/1708.02002, 2017. arXiv: 1708.02002. [en línea]. Disponible en: http://arxiv.org/abs/1708.02002.

[35] X. Wen, X. Yu, Y. Wang, C. Yang e Y. Sun, "A Hybrid 3D–2D Feature Hierarchy CNN with Focal Loss for Hyperspectral Image Classification", *Remote Sensing*, vol. 15, n.º 18, 2023, ISSN: 2072-4292. DOI: 10.3390/rs15184439. [en línea]. Disponible en: https://www.mdpi.com/2072-4292/15/18/4439.

[36] S. Ruder, "An overview of gradient descent optimization algorithms", *CoRR*, vol. abs/1609.04747, 2016. arXiv: 1609.04747. [en línea]. Disponible en: http://arxiv.org/abs/1609.04747.

[37] D. P. Kingma y J. Ba, *Adam: A Method for Stochastic Optimization*, 2017. arXiv: 1412.6980 [cs.LG].

[38] F. Pedregosa, G. Varoquaux, A. Gramfort et al., "Scikit-learn: Machine Learning in Python", *Journal of Machine Learning Research*, vol. 12, págs. 2825-2830, 2011.

[39] A. Paszke, S. Gross, F. Massa et al., *PyTorch: An Imperative Style, High-Performance Deep Learning Library*, 2019. arXiv: 1912.01703.

[40] A. Martín-Pérez, M. Villa, G. Olmeda et al., *SLIM Brain Database: A Multimodal Image Database of In-vivo Human Brains for Tumour Detection*, nov. de 2023. DOI: 10.21203/rs.3.rs-3629358/v1.

[41] D. U. Ozsahin, M. Taiwo Mustapha, A. S. Mubarak, Z. Said Ameen y B. Uzun, "Impact of feature scaling on machine learning models for the diagnosis of diabetes", en *2022 International Conference on Artificial Intelligence in Everything (AIE)*, 2022, págs. 87-94. DOI: 10.1109/AIE57029.2022.00024.

[42] V. N. G. Raju, K. P. Lakshmi, V. M. Jain, A. Kalidindi y V Padma, "Study the Influence of Normalization/Transformation process on the Accuracy of Supervised Classification", en *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*, 2020, págs. 729-735. DOI: 10.1109/ICSSIT48917.2020.9214160.

[43] E. Eşme, "Enhancing classification accuracy through feature extraction: a comparative study of discretization and clustering approaches on sensor-based datasets", *Knowledge and Information Systems*, vol. 66, ago. de 2023. DOI: 10.1007/s10115-023-01960-0.

[44] S. Khirirat, H. R. Feyzmahdavian y M. Johansson, "Mini-batch gradient descent: Faster convergence under data sparsity", en *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, 2017, págs. 2880-2887. DOI: 10.1109/CDC.2017.8264077.

[45] X. Glorot e Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks", *Journal of Machine Learning Research - Proceedings Track*, vol. 9, págs. 249-256, ene. de 2010.

[46] X. Glorot e Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks", en *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Y. W. Teh y M. Titterington, eds., ép. Proceedings of Machine Learning Research, vol. 9, Chia Laguna Resort, Sardinia, Italy: PMLR, mayo de

2010, págs. 249-256. [en línea]. Disponible en: https://proceedings.mlr.press/v9/glorot10a.html.

[47] S. Ioffe y C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift", *CoRR*, vol. abs/1502.03167, 2015. arXiv: 1502.03167. [en línea]. Disponible en: http://arxiv.org/abs/1502.03167.

[48] S. Santurkar, D. Tsipras, A. Ilyas y A. Madry, *How Does Batch Normalization Help Optimization?*, 2019. arXiv: 1805.11604 [stat.ML].

[49] J. Bjorck, C. P. Gomes y B. Selman, "Understanding Batch Normalization", *CoRR*, vol. abs/1806.02375, 2018. arXiv: 1806.02375. [en línea]. Disponible en: http://arxiv.org/abs/1806.02375.

[50] V. Aggarwal, N. Taneja y A. Garg, "Residual Learning and Batch Normalization for Improved Image Classification", *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 7, pág. 2003, mar. de 2019.

[51] I. Salehin y D.-K. Kang, "A Review on Dropout Regularization Approaches for Deep Neural Networks within the Scholarly Domain", *Electronics*, vol. 12, n.º 14, 2023, ISSN: 2079-9292. DOI: 10.3390/electronics12143106. [en línea]. Disponible en: https://www.mdpi.com/2079-9292/12/14/3106.

[52] D. Avila, *PyFPDF: A library for PDF document generation in Python*, Consultado: 23 de junio de 2024, 2024.

[53] N. Unidas. "Objetivos de Desarrollo Sostenible". (2024), [en línea]. Disponible en: https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/.

[54] K. Chui, W. Alhalabi, S. Pang et al., "Disease Diagnosis in Smart Healthcare: Innovation, Technologies and Applications", *Sustainability*, vol. 9, dic. de 2017. DOI: 10.3390/su9122309.

[55] R. A. A. Ahmed y H. Y. H. E. Al-Bagoury, "Artificial Intelligence in Healthcare Enhancements in Diagnosis, Telemedicine, Education, and Resource Management", *Journal of Contemporary Healthcare Analytics*, vol. 6, n.º 12, 1–12, dic. de 2022. [en línea]. Disponible en: https://publications.dlpress.org/index.php/jcha/article/view/55.

[56] N. N. Khanna, M. A. Maindarkar, V. Viswanathan et al., "Economics of Artificial Intelligence in Healthcare: Diagnosis vs. Treatment", *Healthcare*, vol. 10, n.º 12, 2022, ISSN: 2227-9032. DOI: 10.3390/healthcare10122493. [en línea]. Disponible en: https://www.mdpi.com/2227-9032/10/12/2493.

[57] N. Unidas. "Salud y Bienestar - Objetivos de Desarrollo Sostenible". (2024), [en línea]. Disponible en: https://www.un.org/sustainabledevelopment/es/health/.

[58] N. Unidas. "Educación de calidad - Objetivos de Desarrollo Sostenible". (2024), [en línea]. Disponible en: https://www.un.org/sustainabledevelopment/es/education/.

[59] N. Unidas. "Industria, innovación e infraestructuras - Objetivos de Desarrollo Sostenible". (2024), [en línea]. Disponible en: `https://www.un.org/sustainabledevelopment/es/infrastructure/`.

[60] N. Unidas. "Reducción de las desigualdades - Objetivos de Desarrollo Sostenible". (2024), [en línea]. Disponible en: `https://www.un.org/sustainabledevelopment/es/inequality/`.

[61] N. Unidas. "Producción y consumo responsables - Objetivos de Desarrollo Sostenible". (2024), [en línea]. Disponible en: `https://www.un.org/sustainabledevelopment/es/sustainable-consumption-production/`.

[62] N. Unidas. "Alianzas para lograr objetivos - Objetivos de Desarrollo Sostenible". (2024), [en línea]. Disponible en: `https://www.un.org/sustainabledevelopment/es/globalpartnerships/`.

Apéndice A

Anexo

En este capítulo se pretende mostrar las métricas de los resultados. En la primera sección se presentarán las matrices de confusión y las curvas ROC-AUC de cada paciente. Posteriormente, en la segunda sección se presenta el informe de resultados generado por la herramienta.

A.1. Métricas de clasificación para cada paciente

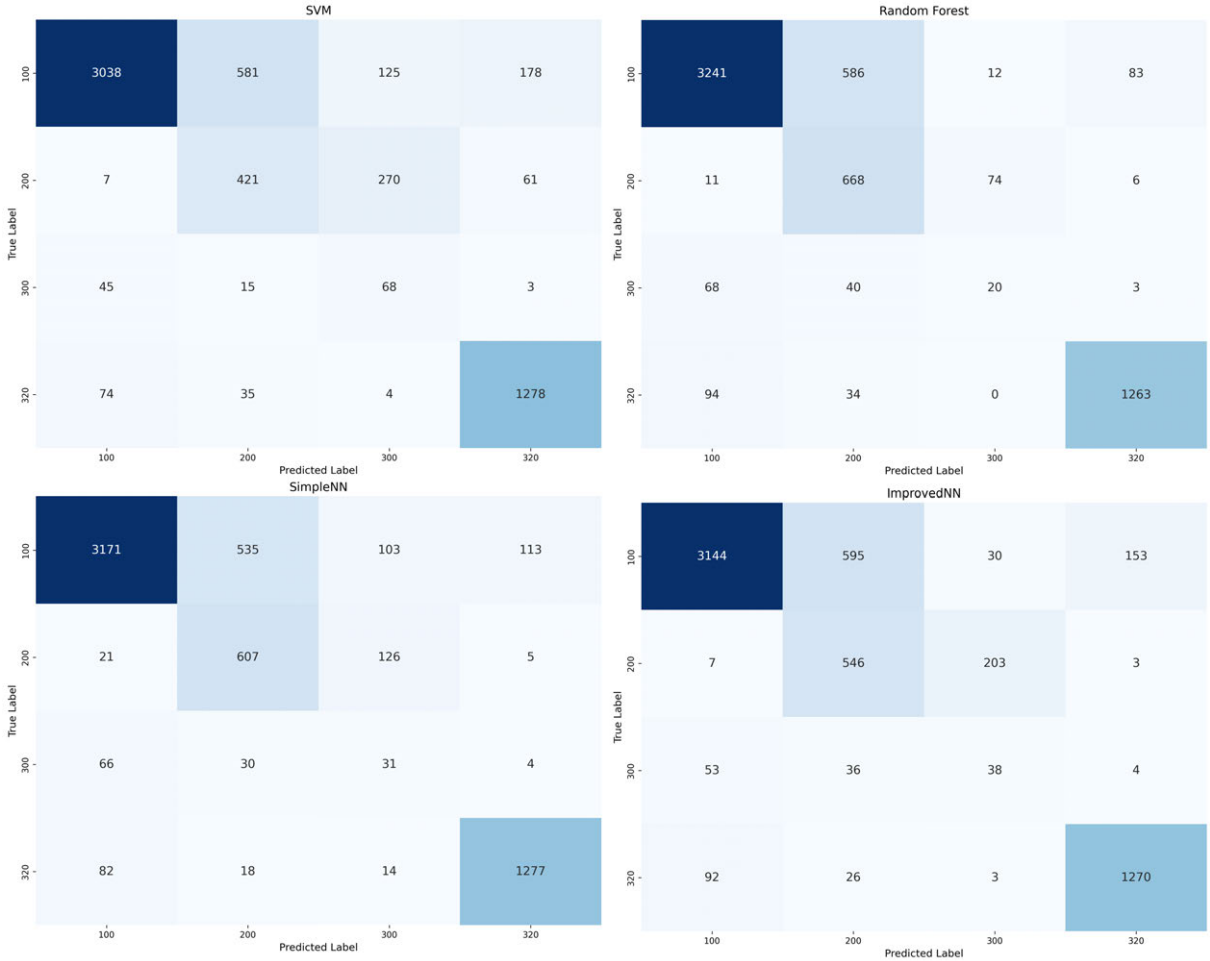


Figura A.1. Matrices de confusión para el paciente 142

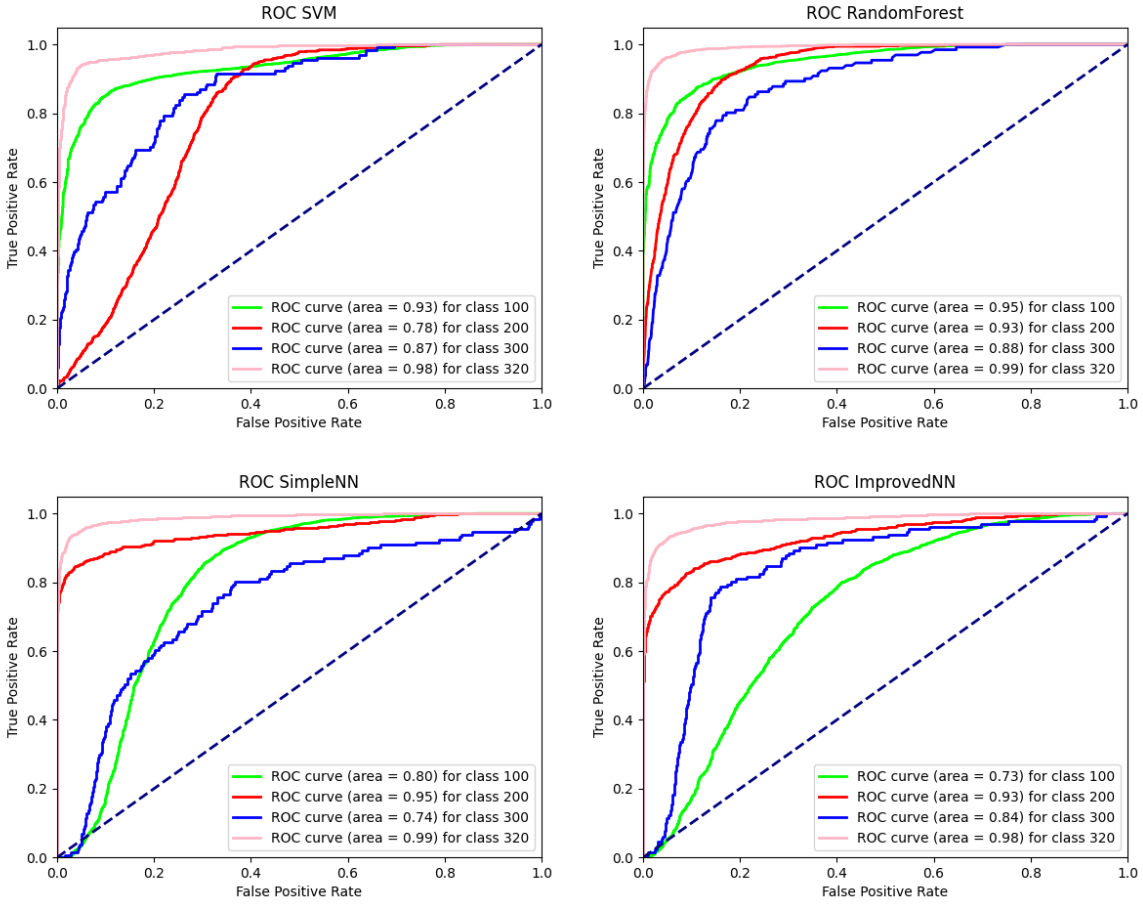


Figura A.2. Curvas ROC-AUC para el paciente 142

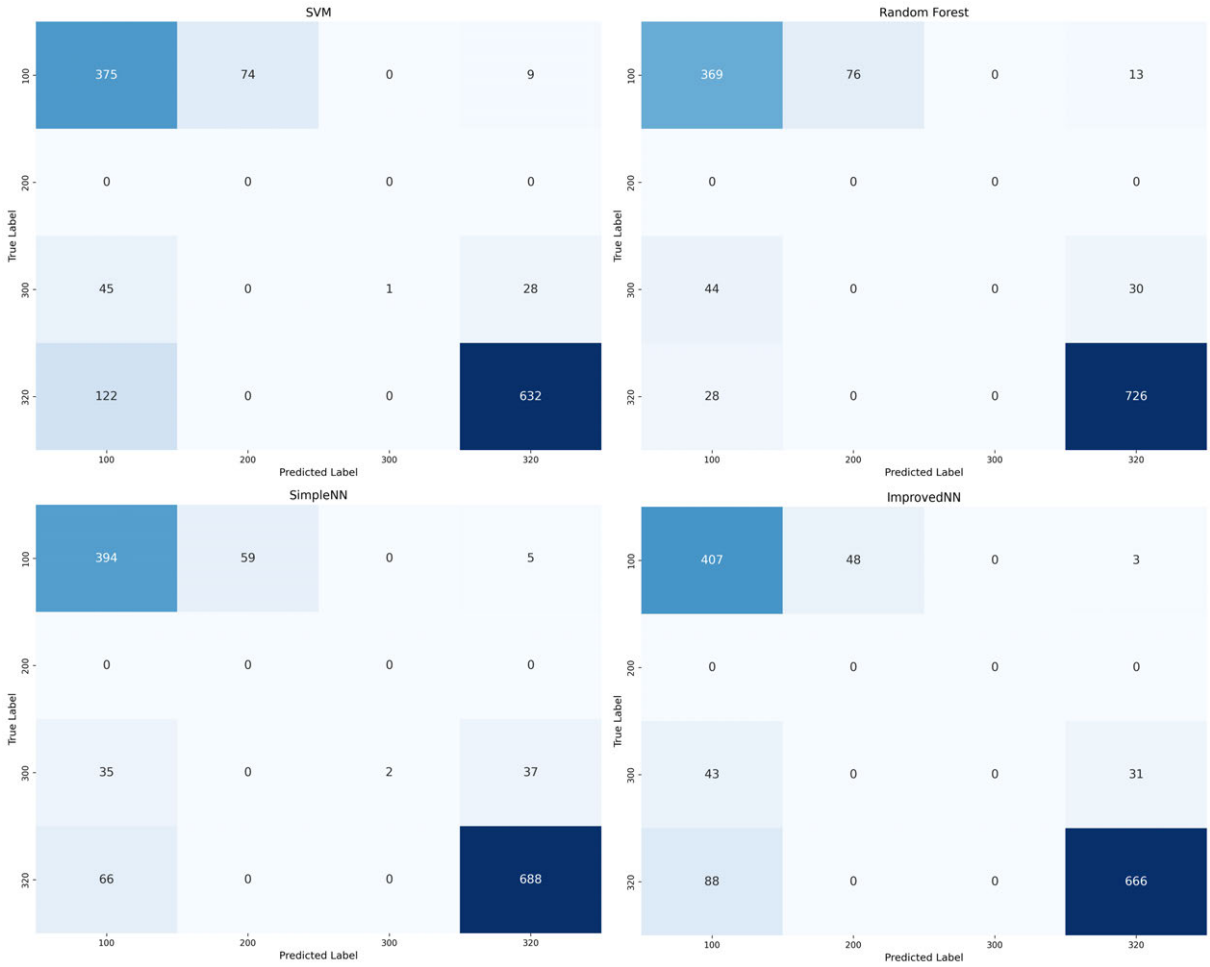


Figura A.3. Matrices de confusión para el paciente 187

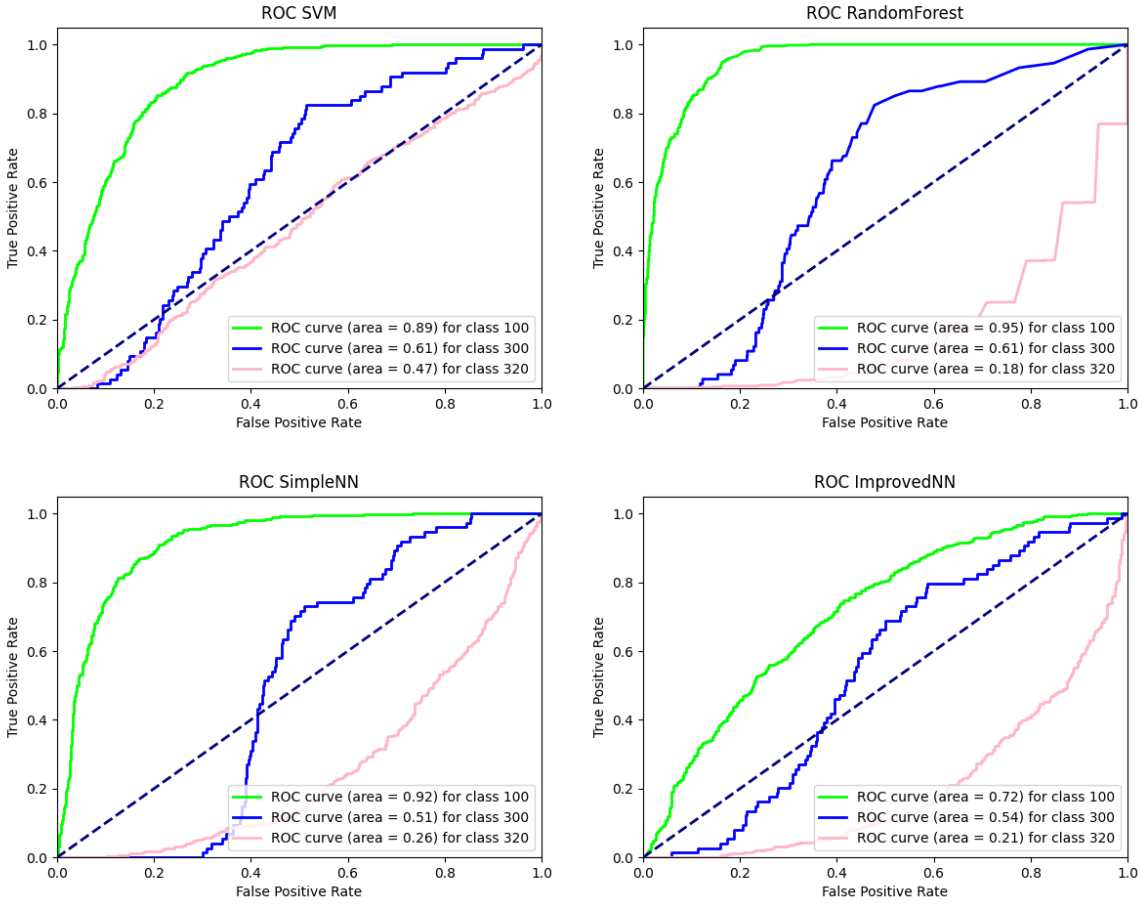


Figura A.4. Curvas ROC-AUC para el paciente 187

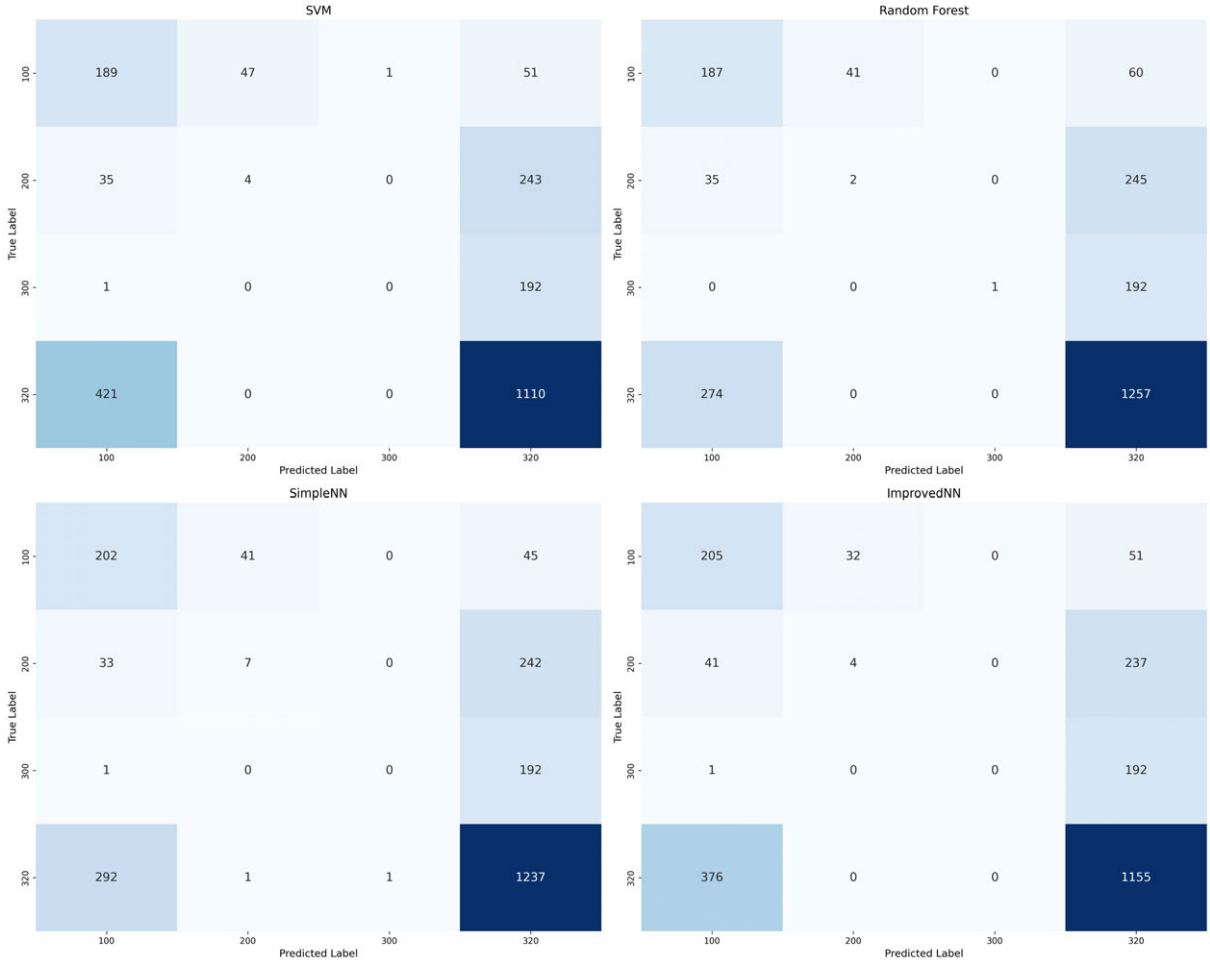


Figura A.5. Matrices de confusión para el paciente 193

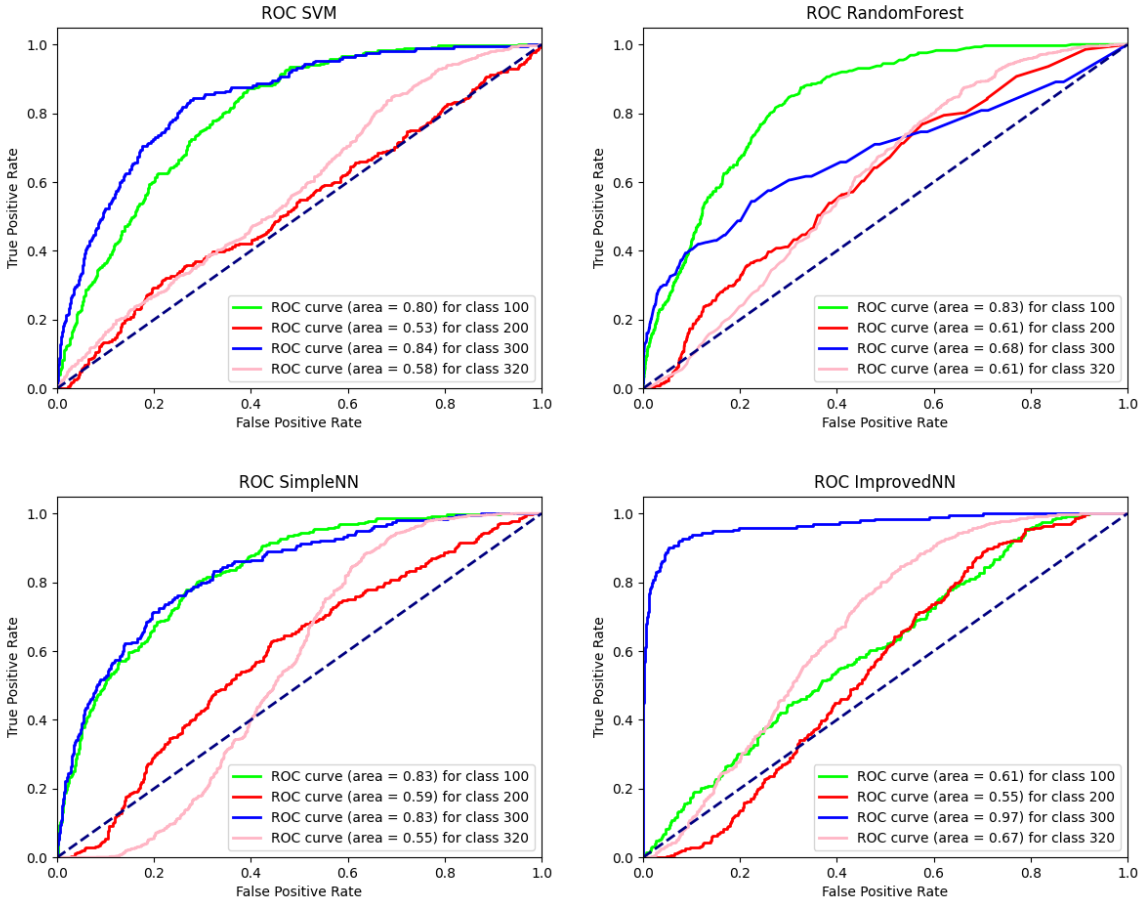


Figura A.6. Curvas ROC-AUC para el paciente 193

A.2. Informe final

La primera parte del informe consta de tablas con resultados de las métricas de cada modelo, estas métricas incluyen exactitud, precisión, recuperación, puntuación F1 y curva ROC-AUC. A continuación de estas tablas, se añade una tabla más con el mejor modelo, en la que se añade la columna con la puntuación del modelo. Cabe mencionar, que la puntuación se calcula mediante la ecuación 4.1 en la que se calcula una puntuación agregada ponderando las métricas. Posterior a las tablas, se presentan las imágenes de cada paciente, que componen los mapas de *ground truth* superpuestos con las imágenes RGB. Finalmente, se presentan los mapas de clasificación predichos por cada modelo.

Model Performance Report

Model Metrics

Model: RandomForest

Accuracy	Precision	Recall	F1 Score	ROC AUC	Score
0.7731	0.7806	0.7731	0.7571	0.7348	0.7583

Model: SimpleNN

Accuracy	Precision	Recall	F1 Score	ROC AUC	Score
0.7644	0.7700	0.7633	0.7513	0.7117	0.7454

Model: ImprovedNN

Accuracy	Precision	Recall	F1 Score	ROC AUC	Score
0.7449	0.7381	0.7440	0.7332	0.6863	0.7230

Model: SVM

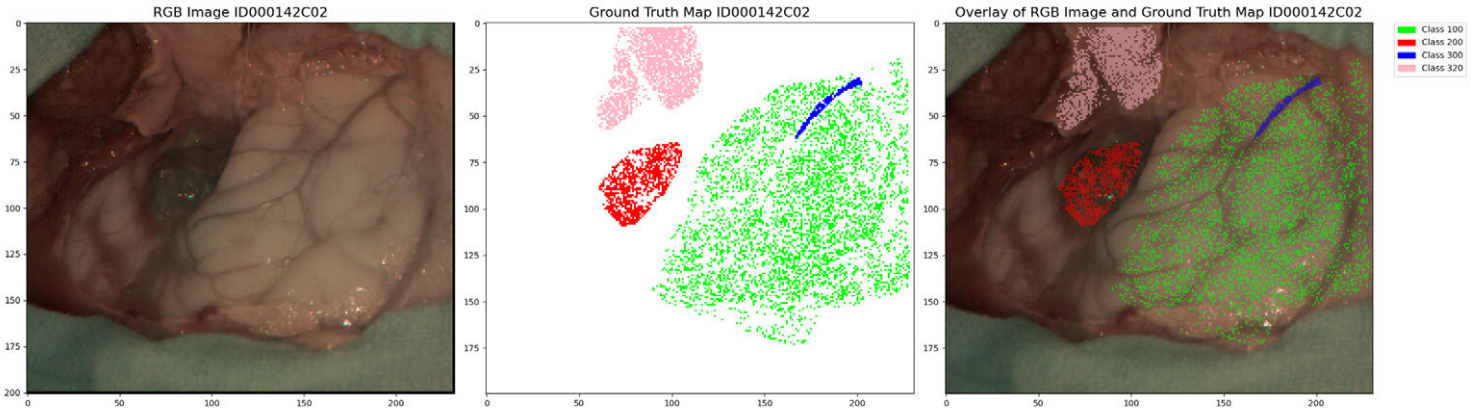
Accuracy	Precision	Recall	F1 Score	ROC AUC	Score
0.7088	0.7387	0.7084	0.7059	0.7458	0.7221

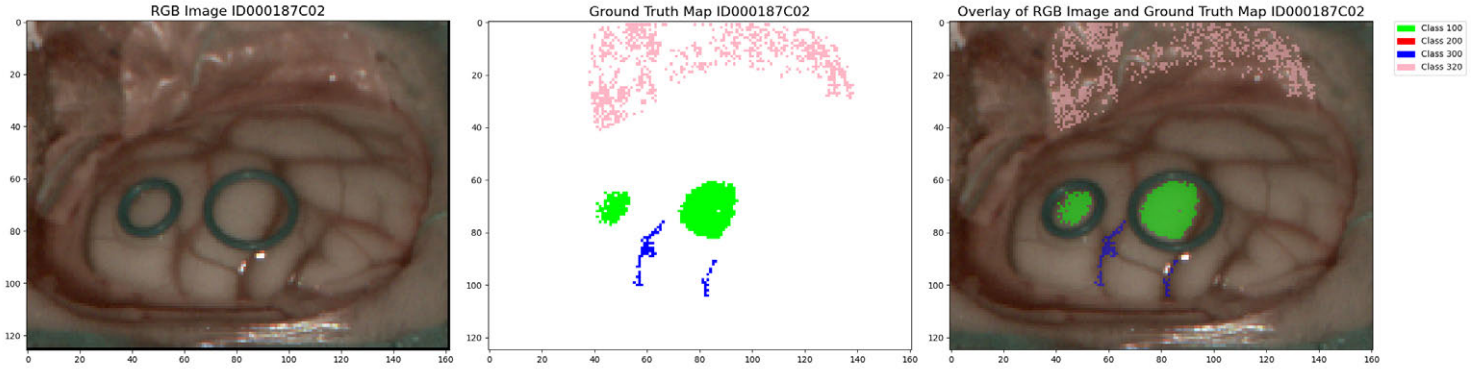
Best Model: RandomForest

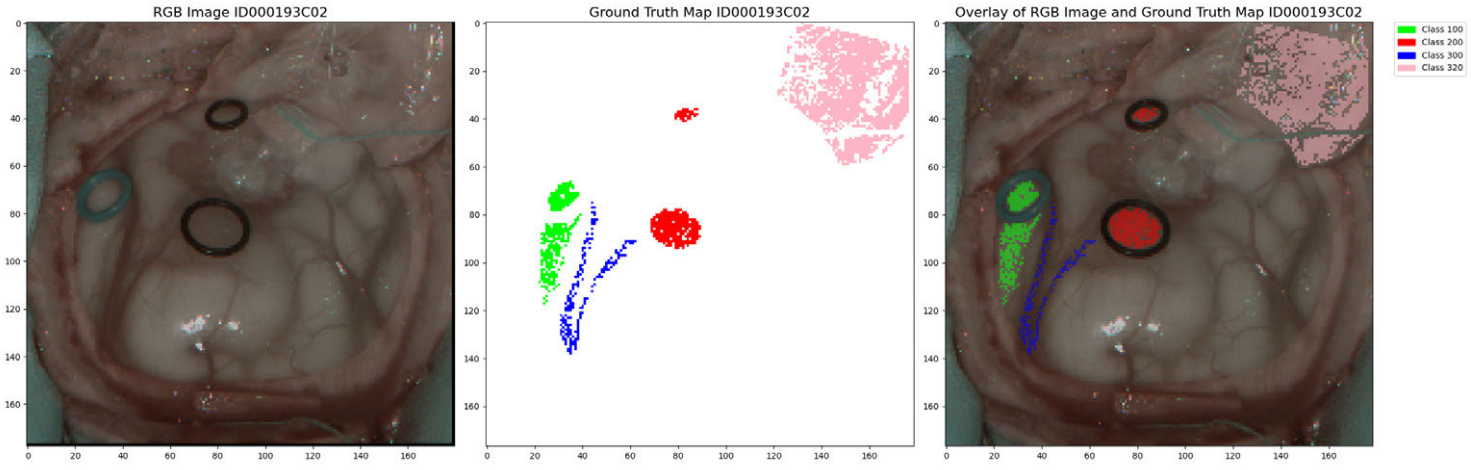
Accuracy	Precision	Recall	F1 Score	ROC AUC	Score
0.7731	0.7806	0.7731	0.7571	0.7348	0.7583

Model Performance Report

Combined Images

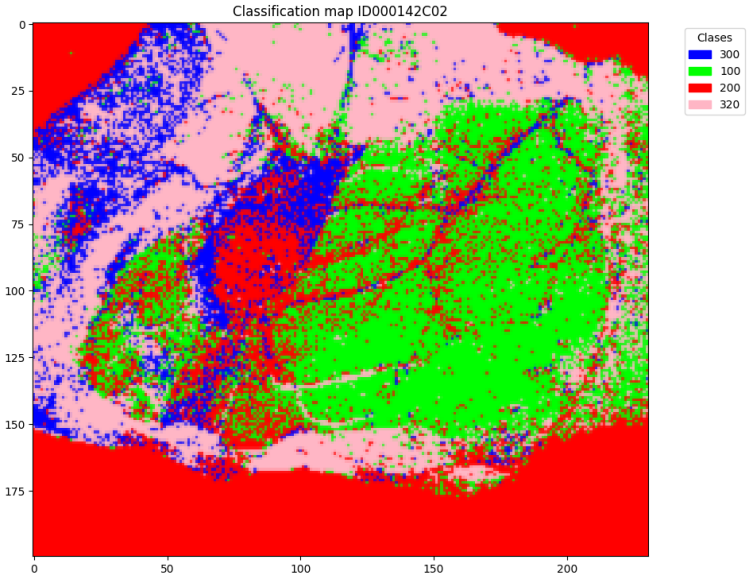


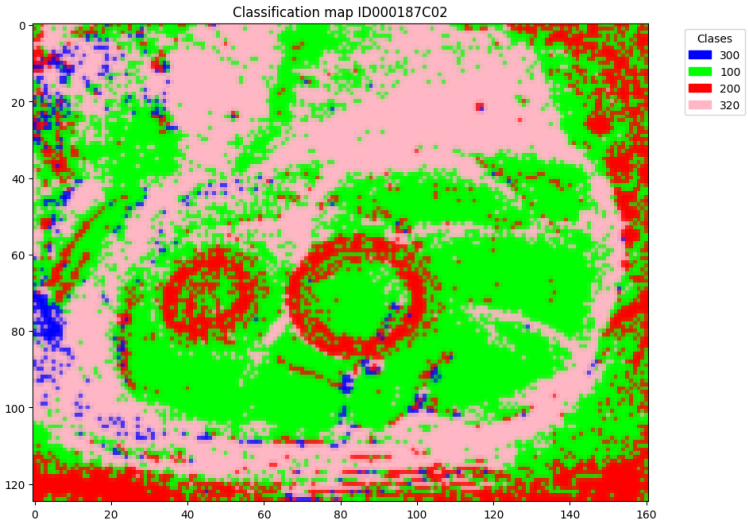




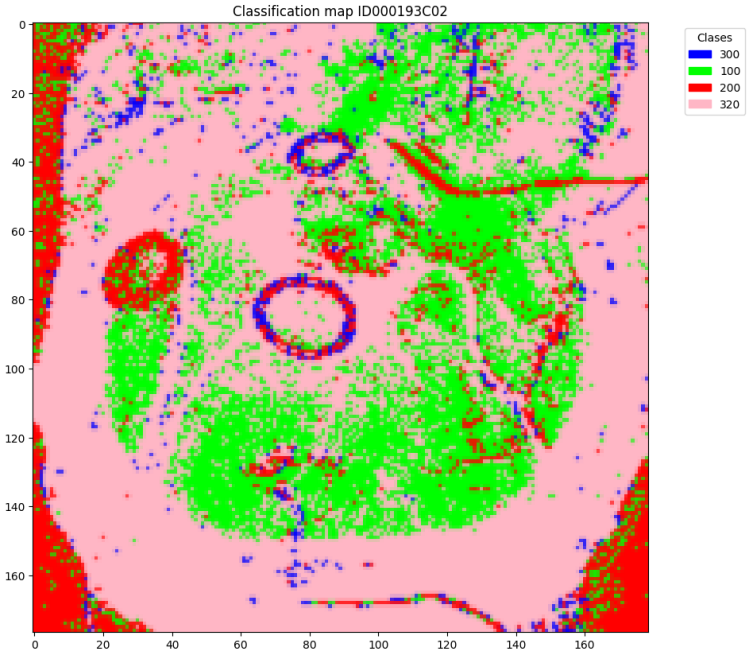
Model Performance Report

Predicted images for ImprovedNN model



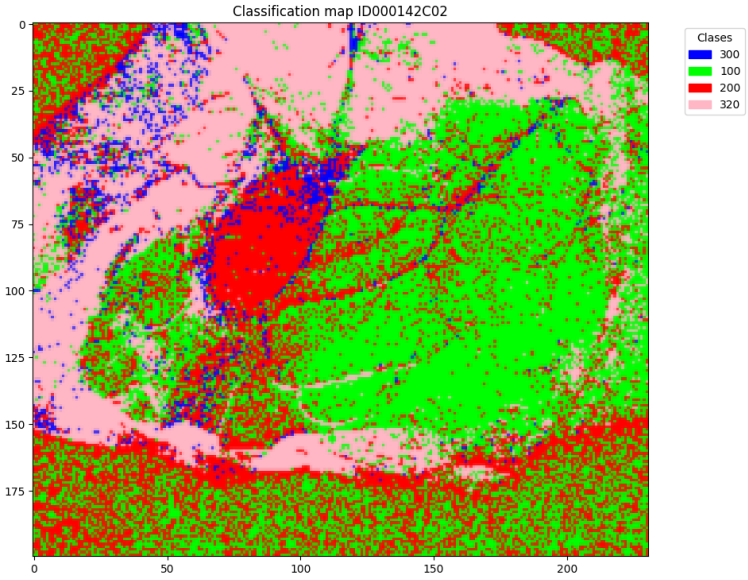


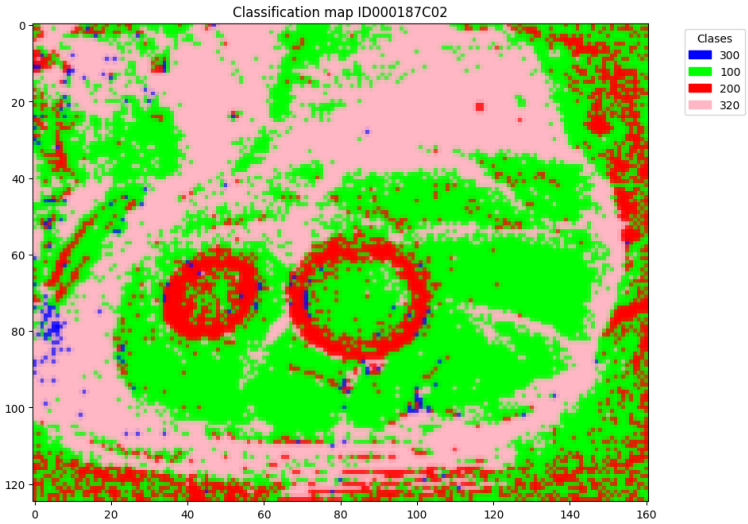
Model Performance Report



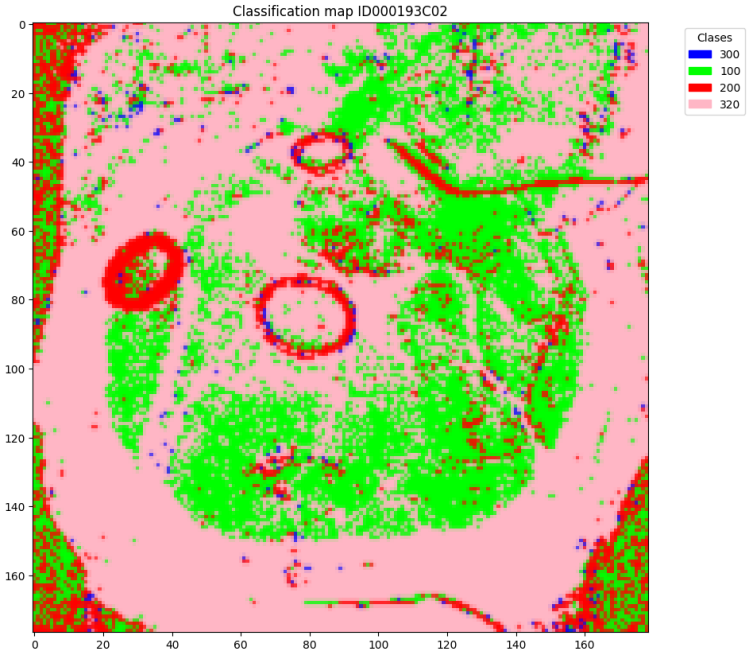
Model Performance Report

Predicted images for RandomForest model



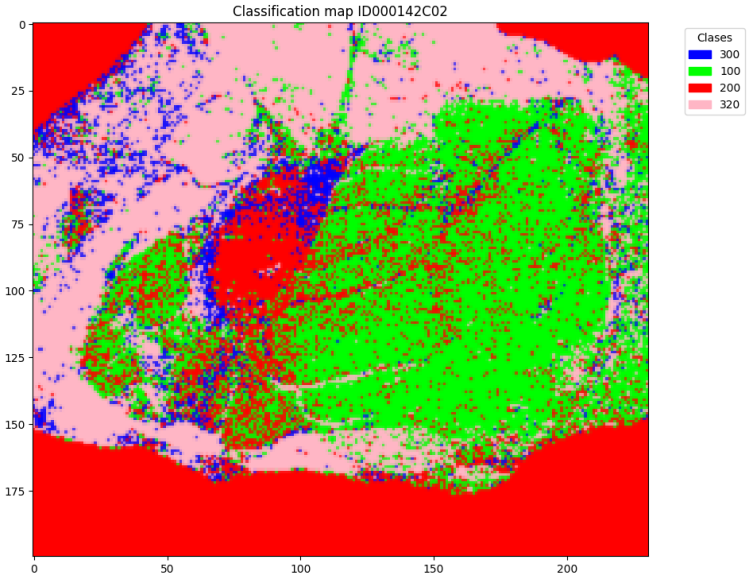


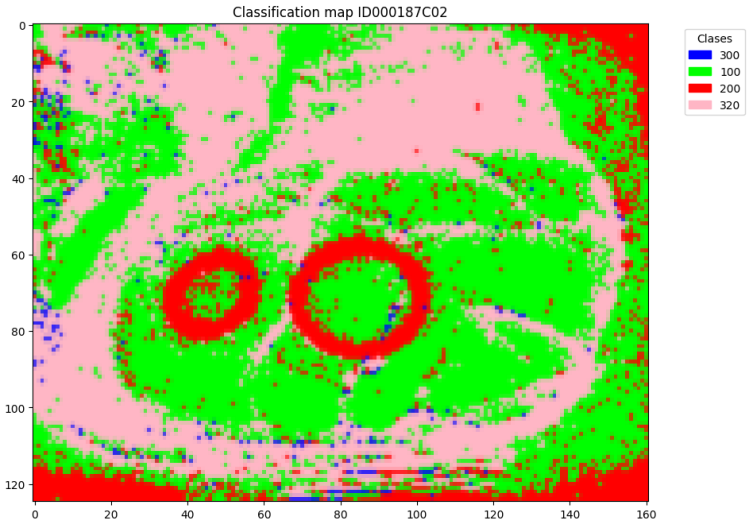
Model Performance Report



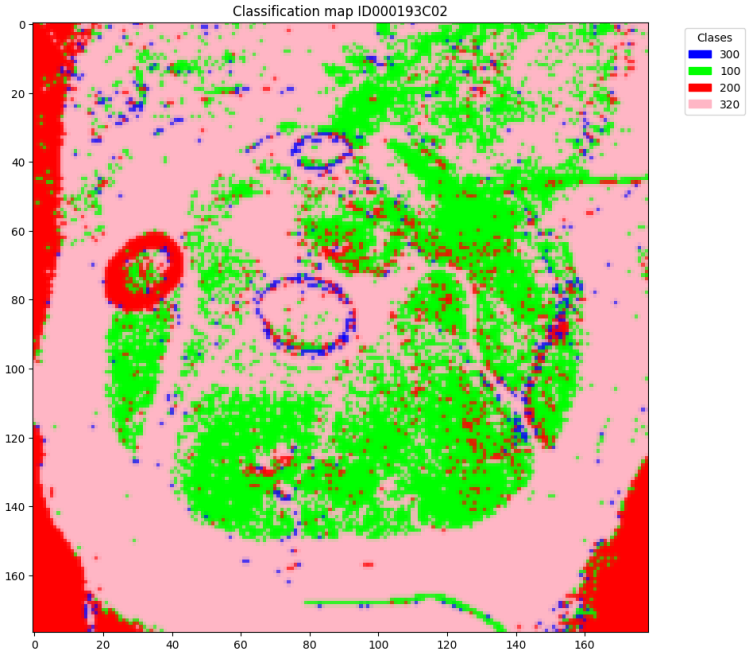
Model Performance Report

Predicted images for SimpleNN model



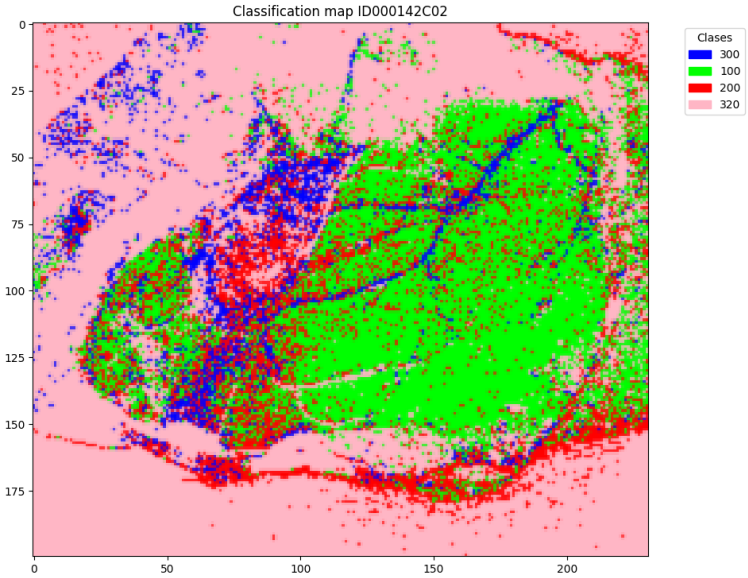


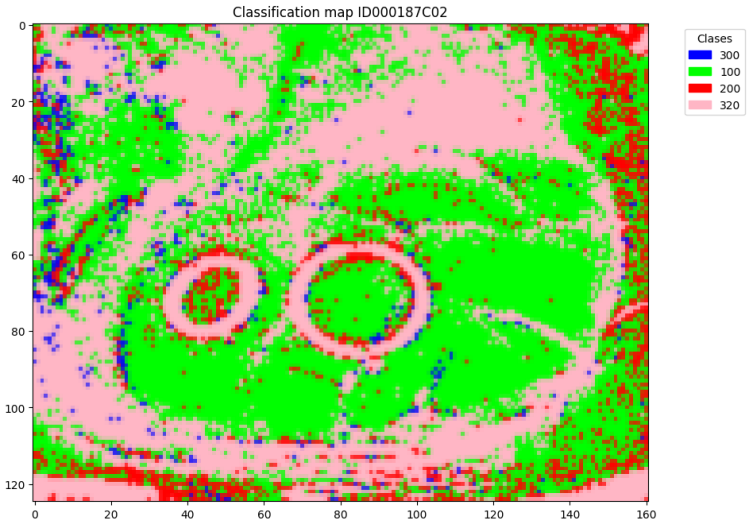
Model Performance Report



Model Performance Report

Predicted images for SVM model





Model Performance Report