Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

# Trabajo de Fin de Grado

Grado en Ingeniería Informática

---

# Aplicación de técnicas de IA para la extracción automática de características morfológicas de interés en retinografías

*Application of AI techniques for the automatic extraction of relevant morphological features in retinographies*

Marcos Jesús Santana Ramos

---

La Laguna, *09* de *julio* de 2024

D. **Rafael Arnay del Arco**, profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

**C E R T I F I C A (N)**

Que la presente memoria titulada:

*“Aplicación de técnicas de IA para la extracción automática de características morfológicas de interés en retinografías”*

ha sido realizada bajo su dirección por D. **Marcos Jesús Santana Ramos**.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 09 de julio de 2024

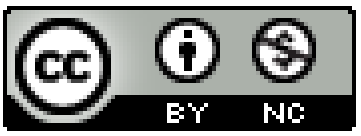
# Agradecimientos

Quisiera expresar mi más sincero agradecimiento a mi tutor, Rafael Arnay del Arco, por su invaluable guía, paciencia y apoyo continuo. Su experiencia y conocimientos han sido fundamentales para la realización de este trabajo.

Agradezco también a todos los profesores que me han impartido sus conocimientos a lo largo de estos años. Su dedicación y enseñanza han sido esenciales para mi formación académica y profesional.

Finalmente, quiero agradecer a mi familia y amigos por su apoyo, comprensión y estar siempre a mi lado en cada paso del camino. Sus palabras de aliento y su confianza en mí han sido una fuente constante de motivación.

# Licencia



**Resumen**

*Este Trabajo de Fin de Grado se centra en la automatización del proceso de extracción de características de interés en retinografías, utilizando técnicas de procesamiento de imágenes y aprendizaje automático desarrolladas en Python.*

*El proyecto desarrolla un sistema basado en una red neuronal convolucional (CNN) de arquitectura UNET, específicamente diseñada para la segmentación del disco y copa óptica en imágenes de retinografías. A partir de estas segmentaciones, se automatiza la extracción de 14 características morfológicas, que son útiles para un análisis clínico.*

*Además, estas características se utilizarán en un modelo de aprendizaje automático, concretamente un árbol de decisión, para clasificar las imágenes en categorías relacionadas con la presencia o ausencia de glaucoma.*

**Palabras clave:** retinografías, red neuronal convolucional (CNN), UNET, segmentación, disco y copa óptica, árbol de decisión, glaucoma.

**Abstract**

*This Final Degree Project focuses on automating the process of extracting relevant features in* retinographies*, using image processing and machine learning techniques developed in Python.*

*The project develops a system based on a convolutional neural network (CNN) with a UNET architecture, specifically designed for the segmentation of the optic disc and optic cup in* retinographies*. From these segmentations, the extraction of 14 morphological features is automated, which are useful for clinical analysis.*

*In addition, these features will be used in a machine learning model, specifically a decision tree, to classify the images into categories related to the presence or absence of glaucoma.*

**Keywords:** retinographies, convolutional neural networks (CNN), UNET, segmentation, optic disc and optic cup, decision tree, glaucoma.

# Índice general

# Índice de figuras

# Índice de tablas

# Capítulo 1 Introducción

El glaucoma es una enfermedad crónica y es la principal causa de ceguera irreversible en los países desarrollados, siendo una enfermedad difícil de detectar en sus primeras etapas debido a su naturaleza asintomática. Una vez diagnosticado, es muy importante identificar los factores de riesgo para la progresión de esta patología porque aquellos pacientes que tienen un mayor riesgo, deben estar sujetos a un protocolo de monitoreo más estricto con medidas terapéuticas más intensivas. Entre los factores más comúnmente identificados se encuentran la edad, la presión intraocular, el síndrome de pseudoexfoliación, el grosor corneal y las hemorragias del nervio óptico o de la papila.

Este proyecto se enfoca en la automatización de la extracción de características relevantes de retinografías para mejorar el estudio y diagnóstico del glaucoma. Estas características, que son fácilmente interpretables por sí mismas, ya que representan estructuras morfológicas de la retina, se utilizarán como entrada en un modelo de aprendizaje automático para clasificar las imágenes en dos categorías: “normal” y “glaucoma”.

En concreto, se plantea la extracción de catorce características morfológicas de dos estructuras de la retina, la copa y el disco, véase Figura 1.

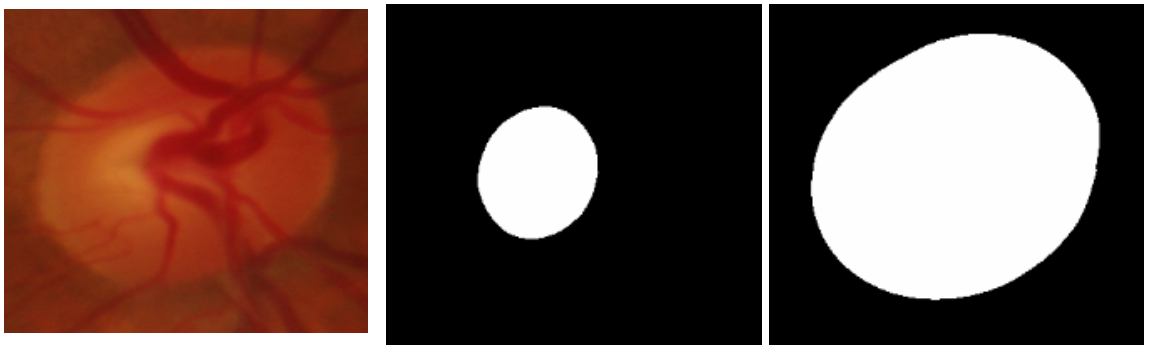


**Figura 1: Ejemplo de retinografía a la izquierda, en el centro y a la derecha se muestran máscaras binarias que representan la región de la copa y el disco, respectivamente.**

## 1.1 Objetivos

**Segmentación de imágenes mediante Redes Neuronales Convolucionales (Convolutional Neural Networks, CNN)**: Utilizar redes neuronales convolucionales para segmentar el disco y la copa óptica en retinografías. Esta fase es fundamental para identificar las áreas relevantes de la imagen que serán analizadas posteriormente.

**Extracción automática de características**: Una vez segmentadas las imágenes, el proyecto se centrará en la extracción automática de características morfológicas de la copa y el disco óptico. Esto incluye dimensiones, proporciones, y otras métricas relevantes que son esenciales para la evaluación clínica posterior del estado ocular.

**Clasificación mediante árboles de decisión**: Desarrollar y entrenar un modelo de árbol de decisión para categorizar las retinografías basándose en las características

extraídas. El objetivo es determinar de manera automatizada si una imagen indica la presencia de glaucoma, facilitando así un diagnóstico rápido y preciso.

## 1.2 Antecedentes y Estado Actual

En la disciplina de segmentación de imágenes médicas, los métodos convencionales han sido esenciales para la detección precisa de elementos críticos. Frecuentemente, se han utilizado técnicas de umbralización o segmentación basada en regiones para segmentar imágenes [1]. Estos métodos, si bien son útiles bajo ciertas condiciones, continúan evolucionando para generar representaciones de imagen más exactas y con menos interferencias.

Estas técnicas han sido fundamentales para analizar detalladamente las propiedades de la retina, facilitando así la identificación y delineación de áreas clave.

Con los progresos en tecnología de procesamiento de imágenes, se ha visto que la segmentación automática ofrece resultados prometedores en términos de precisión para la segmentación del disco óptico [2].

Particularmente, las redes neuronales convolucionales destinadas a la segmentación de discos ópticos y copas han probado ser excepcionalmente eficaces, convirtiéndose en herramientas valiosas para la identificación automática de glaucoma [3].

Este progreso indica un avance continuo hacia técnicas más avanzadas y automatizadas en la segmentación de imágenes médicas. Aunque estos sistemas pueden hacer predicciones muy precisas, el proceso interno por el cual llegan a esas predicciones no es fácilmente comprensible ni interpretable por los humanos, por eso se suele utilizar el término de “caja negra” para referirse a estos modelos. En campos como la medicina, es esencial tener mecanismos para explicar las decisiones tomadas por las redes neuronales. Recientemente, se ha establecido que las retinografías pueden ser categorizadas efectivamente utilizando características morfológicas derivadas de la copa y el disco óptico, logrando resultados comparables a aquellos obtenidos mediante técnicas de aprendizaje profundo [4].

Este proyecto busca automatizar la extracción de este tipo de características y usarlas en combinación con un modelo de árbol de decisión, para proporcionar una clasificación que pueda ser interpretable por especialistas médicos.

# Capítulo 2 Desarrollo

## 2.1 Conjunto de Datos

El proyecto utiliza el conjunto de datos RIM-ONE DL [5], diseñado para la evaluación y diagnóstico del glaucoma a través de técnicas de aprendizaje profundo. Este conjunto de datos se compone de retinografías recopiladas en tres hospitales españoles: Hospital Universitario de Canarias (HUC) en Tenerife, Hospital Universitario Miguel Servet (HUMS) en Zaragoza, y Hospital Clínico Universitario San Carlos (HCSC) en Madrid.

RIM-ONE DL incluye un total de 485 retinografías, con 313 imágenes de sujetos normales y 172 de pacientes diagnosticados con glaucoma. Este conjunto de datos está disponible en dos formatos de partición: una partición aleatoria que mezcla todas las imágenes para formar los conjuntos de entrenamiento y testeo, y una partición por hospital, donde las imágenes de HUC se utilizan para entrenamiento mientras que las de HUMS y HCSC se destinan a testeo. Esta estructura de datos facilita la validación y prueba de modelos de aprendizaje automático bajo diferentes escenarios y condiciones de evaluación.

Además de las imágenes, RIM-ONE DL proporciona segmentaciones de referencia del disco óptico y la copa, realizadas manualmente por expertos en glaucoma. Estas segmentaciones son esenciales para entrenar y evaluar modelos de segmentación automática, ofreciendo un estándar que ayude a mejorar la precisión y la eficacia de las técnicas de procesamiento de imágenes. En la Figura 1 se puede apreciar visualmente un ejemplo de retinografía y sus máscaras correspondientes de copa y disco óptico.

Adicionalmente, para los propósitos de este proyecto, se ha optado por utilizar las imágenes en una partición aleatoria en lugar de la partición por hospital. Las retinografías están organizadas en una estructura de carpetas denominada **RIM-ONE_DL_images**, que incluye un conjunto de entrenamiento (**training_set**) de 339 imágenes y un conjunto de prueba (**test_set**) de 146 imágenes. Cada uno de estos conjuntos está subdividido en dos carpetas adicionales que clasifican las imágenes en **glaucoma** para aquellas retinografías de pacientes con diagnóstico de glaucoma, y **normal** para imágenes de sujetos sanos.

En cuanto a las segmentaciones de referencia, estas se encuentran en una carpeta llamada **RIM-ONE_DL_reference_segmentations**, que contiene dos subcarpetas: una bajo la etiqueta **glaucoma**, almacenando las segmentaciones del disco y la copa para las imágenes diagnosticadas con glaucoma, y otra etiquetada como **normal**, con las segmentaciones correspondientes a las imágenes de sujetos sanos.

## 2.2 Segmentación de Imágenes

La segmentación de imágenes médicas es una tarea fundamental en el campo del análisis de imágenes biomédicas, especialmente en la oftalmología, donde la identificación precisa de estructuras anatómicas es crucial para el diagnóstico y seguimiento de enfermedades como el glaucoma. En este proyecto, se emplea la

arquitectura de red neuronal convolucional conocida como UNET [6], la cual es ampliamente reconocida por su eficacia en tareas de segmentación médica debido a su capacidad para trabajar con una cantidad limitada de datos y lograr una alta precisión.

### 2.2.1 Modelo UNET

El modelo UNET es una arquitectura de red neuronal convolucional especialmente diseñada para la segmentación precisa de imágenes biomédicas. Esta red se distingue por su estructura en forma de "U", que se divide en dos secciones principales: el codificador (o camino de contracción) y el decodificador (o camino expansivo), como se muestra en la Figura 2.

El codificador está compuesto por varias capas de convolución seguidas de operaciones de max pooling. En esta sección, las convoluciones se utilizan para extraer características progresivamente más complejas de la imagen, mientras que el max pooling reduce la resolución espacial de los mapas de características, ayudando a capturar el contexto más amplio de la imagen. Cada capa de convolución en el codificador aumenta la profundidad de los mapas de características, almacenando información esencial sobre diferentes aspectos y detalles de la imagen.

El decodificador utiliza operaciones de convolución transpuesta o 'up-sampling' para reconstruir la imagen desde las características codificadas, aumentando la resolución de los mapas de características paso a paso hacia la salida. En cada etapa del camino expansivo, se realizan operaciones de concatenación que combinan los mapas de características del codificador con los del decodificador a través de conexiones de salto. Estas conexiones permiten que las características contextuales capturadas durante la contracción influyan en la reconstrucción detallada durante la expansión, facilitando la segmentación precisa de elementos específicos en la imagen.

Las conexiones de salto juegan un papel fundamental al reintroducir información espacial y detallada en el decodificador. Estas conexiones copian directamente los mapas de características de las capas del codificador y los combinan con los mapas del decodificador, ayudando a restaurar los detalles y las estructuras precisas en la imagen final de segmentación.

La última capa del decodificador aplica una convolución para producir el mapa de segmentación final, que clasifica cada píxel de la imagen original en categorías de segmento. Este mapa tiene la misma resolución espacial que la imagen de entrada, lo que permite una delimitación exacta de las estructuras de interés.

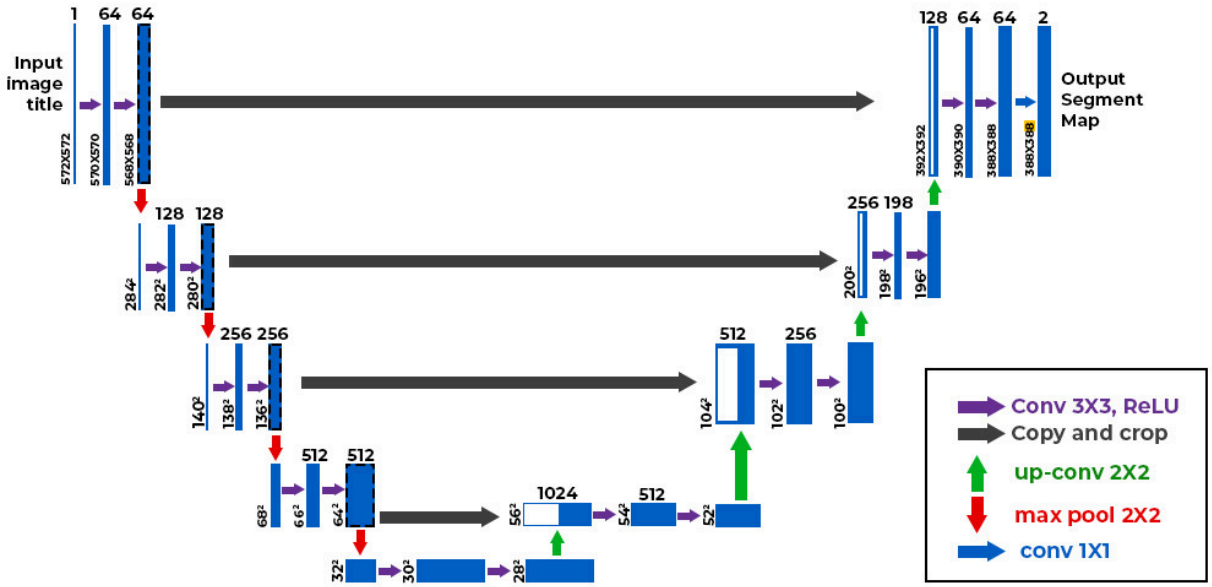



**Figura 2: Modelo UNET**

### 2.2.2 Implementación y Configuración del Modelo

Para llevar a cabo este proyecto, se ha diseñado un script utilizando Python [7], un lenguaje de programación de alto nivel que es ampliamente utilizado en el ámbito de la ciencia de datos y el aprendizaje automático debido a su sintaxis clara y su rica biblioteca de paquetes especializados.

El script se apoya en Keras [8], un framework de alto nivel para redes neuronales que opera como interfaz para TensorFlow, facilitando la creación y entrenamiento de modelos de aprendizaje profundo de manera más intuitiva y accesible. Keras permite a los usuarios construir rápidamente prototipos de modelos de aprendizaje profundo con pocas líneas de código, lo que lo hace ideal para experimentación rápida.

TensorFlow [9], por su parte, es una biblioteca de código abierto desarrollada por Google para cálculos numéricos y aprendizaje automático que proporciona una base robusta y flexible para construir y entrenar modelos avanzados de aprendizaje profundo.

Se establecieron las dimensiones iniciales de las imágenes en 224x224 píxeles, un tamaño que equilibra el detalle y la eficiencia computacional para modelos de aprendizaje profundo. Se eligió un tamaño del lote de 16 y un número de épocas de 30 para entrenar el modelo UNET en el caso del disco óptico y un tamaño del lote de 20 y un número de épocas de 50 para la copa .

- **Tamaño del lote (batch size)**: Representa el número de muestras de entrenamiento que se procesarán antes de que el modelo actualice los parámetros internos. Un batch size de 16 significa que el modelo toma 16 imágenes a la vez, procesándolas en conjunto antes de realizar una actualización de gradientes.
- **Época (epoch)**: Representa una iteración completa sobre todo el conjunto de datos de entrenamiento. Un valor de 30 épocas indica que el conjunto de datos se pasa por la red neuronal 30 veces. Cada época puede ayudar al modelo a aprender más profundamente de los datos, aunque más épocas también pueden llevar a un sobreajuste.

### 2.2.3 Construcción y Entrenamiento del Modelo

El proceso de construcción del modelo de segmentación UNET comienza con la definición de la entrada del modelo, especificando las dimensiones de las imágenes que será de

224x224 píxeles y 3 canales de color (RGB). A continuación, se detalla la arquitectura de la red, que está compuesta por un camino de contracción y un camino de expansión, típicos de la estructura UNET.

El camino de contracción está diseñado para capturar la contextualización de la imagen a través de capas sucesivas de convolución y pooling:

- **Primera Capa Convolutiva**: Consiste en dos capas convolucionales con 32 filtros de 3x3, seguidas de una operación de Dropout (15% para prevenir el sobreajuste) y una capa de MaxPooling para reducir las dimensiones espaciales.
- **Segunda a Cuarta Capa Convolutiva**: Cada una sigue un esquema similar con un aumento en el número de filtros (64, 128 y 256, respectivamente), Dropout y MaxPooling para seguir profundizando y reduciendo las dimensiones mientras aumenta la profundidad de los mapas de características.

El camino expansivo utiliza la información contextualizada y comprimida del codificador para reconstruir la imagen de segmentación a través de capas de convolución transpuesta y concatenaciones:

- **Capas de Convolución Transpuesta**: Se inicia con 256 filtros y se reduce progresivamente (256, 128, 64, 32), donde cada capa utiliza una convolución transpuesta para incrementar las dimensiones de los mapas de características.
- **Concatenación**: En cada etapa del camino expansivo, las características del camino de contracción correspondientes son concatenadas con las del decodificador para recuperar la información espacial perdida durante el proceso de contracción, crucial para una segmentación precisa.

La salida del modelo se genera a través de una capa convolucional con un filtro de tamaño 1x1 y una función de activación sigmoidal. Esta configuración permite clasificar cada píxel de la imagen en categorías binarias, identificando si pertenece o no a la región de interés (disco o copa óptica).

Se compilan dos instancias del modelo UNET, una para la segmentación de la copa y otra para el disco óptico. Ambos modelos se configuran con el optimizador **adam** y la función de pérdida **binary_crossentropy**, que son adecuados para tareas de clasificación binaria. La métrica de **accuracy** se utiliza para monitorear la precisión de la segmentación durante el entrenamiento.

A partir de este punto se replica para ambos modelos, tanto de disco óptico como el de la copa óptica, poniendo de ejemplo el del disco óptico:

El proceso de entrenamiento del modelo para la segmentación del disco óptico se realiza utilizando el conjunto de datos de entrenamiento. Se ajustan los parámetros del modelo durante 30 épocas con un tamaño de lote de 16, empleando el 20% de los datos de entrenamiento como conjunto de validación para monitorizar y evitar el sobreajuste.

### 2.2.4 Evaluación y Predicciones

Finalmente, se utiliza el modelo entrenado para realizar predicciones en el conjunto de testeo. Este paso es esencial para comprobar cómo el modelo se comporta con datos que no fueron utilizados durante la fase de entrenamiento.

Después de obtener las predicciones del modelo, es crucial convertirlas de probabilidades a segmentaciones binarias para evaluar su precisión. Las predicciones iniciales, en escala de grises, indican la probabilidad de que cada píxel pertenezca a la región de interés (como el disco óptico). Para convertir estas probabilidades en una máscara binaria, se emplea un proceso de umbralización.

La umbralización se realiza definiendo un valor de umbral que convierte las probabilidades en valores binarios (0 o 1). Los píxeles con probabilidades superiores al umbral se marcan como 1 (pertenecientes al disco o la copa), mientras que los inferiores se asignan a 0 (fondo).

Se puede apreciar en la Figura 3 como de esta manera se pasan de imágenes en el rango [0,1] a imágenes binarizadas útiles para análisis posteriores.

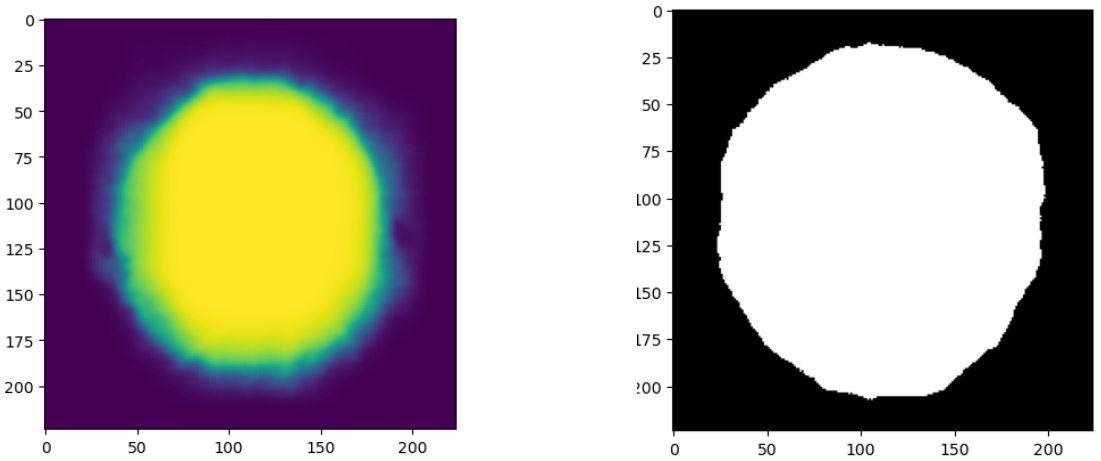



**Figura 3: Predicción en el rango [0,1] a la izquierda e imágen binarizada a la derecha**

Para cada imagen de prueba, se calcula la intersección sobre la unión (IoU) comparando la predicción binaria con la máscara de referencia correspondiente.

Para calcular esta métrica, primero se calcula el área de intersección entre la predicción binaria y la máscara de referencia. Esto se logra utilizando la operación lógica **AND** entre la predicción y la máscara. La función **np.sum** suma todos los valores True (1), que representan los píxeles donde tanto la predicción como la máscara coinciden (es decir, ambos marcan la región como parte del disco). De manera similar, se calcula la unión de los píxeles marcados como parte del disco o la copa en cualquiera de las dos imágenes (predicción o máscara). Aquí se usa la operación lógica **OR**, y nuevamente se suma todos los valores True.

Finalmente el valor IoU se calcula como el cociente de la intersección entre la unión. Esto proporciona una medida de qué tan bien la predicción coincide con la máscara de referencia en términos de área cubierta. En la Figura 4 se puede ver el código que implementa el cálculo de la IoU.

```python
# Calculo de la Union sobre la intersección para el disco
# Calcular IoU para cada muestra
iou_total = 0

# Calcular IoU para cada muestra
for i in range(len(X_test)):
    # Calcular la intersección y la unión
    intersection = np.sum(np.logical_and(binary_predictions[i].squeeze(), y_test_disc[i]))
    union = np.sum(np.logical_or(binary_predictions[i].squeeze(), y_test_disc[i]))

    # Calcular IoU
    iou = intersection / union
    iou_total += iou
```

**Figura 4: Cálculo de IoU**

Este cálculo de IoU se realiza como parte esencial del proceso de evaluación, proporcionando una métrica cuantitativa para medir la precisión de las segmentaciones realizadas por el modelo. El umbral óptimo para binarizar las imágenes de probabilidad se determina mediante el umbral que mejor resultado promedio obtenga en la métrica de intersección sobre la unión (IoU) en el conjunto de entrenamiento. La función **optimize_threshold_for_disc,** representada en la Figura 5, implementa este cálculo.

```python
def optimize_threshold_for_disc(X_train, y_train_disc, model_predictions):
    thresholds = np.arange(0.1, 1.0, 0.05)
    best_threshold = 0
    best_iou_average = 0

    for threshold in thresholds:
        binary_predictions = (model_predictions > threshold).astype(np.uint8)
        iou_total = 0

        for i in range(len(X_train)):
            intersection = np.sum(np.logical_and(binary_predictions[i].squeeze(), y_train_disc[i]))
            union = np.sum(np.logical_or(binary_predictions[i].squeeze(), y_train_disc[i]))
            iou = intersection / union
            iou_total += iou
        iou_average = iou_total / len(X_train)
        print(f"Threshold: {threshold:.2f}, IoU promedio: {iou_average:.3f}")

        # Keep the best threshold
        if iou_average > best_iou_average:
            best_iou_average = iou_average
            best_threshold = threshold

    return best_threshold, best_iou_average

# Usar la función para encontrar el mejor umbral
best_threshold, best_iou_average = optimize_threshold_for_disc(X_train, y_train_disc, preds_train)
print(f"Mejor umbral: {best_threshold:.2f}, Mejor IoU promedio: {best_iou_average:.3f}")
```

**Figura 5: Función optimize_threshold_for_disc para el cálculo del umbral que optimiza la IoU promedio en el conjunto de entrenamiento**

Una vez seleccionado el umbral óptimo para la métrica IoU, se procede a aplicar este umbral a las predicciones del conjunto de test para convertir las probabilidades a valores binarios claros y almacenar las imágenes umbralizadas en un directorio específico para su uso posterior.

Además de las métricas de IoU detalladas anteriormente, que ofrecen una evaluación de la precisión de segmentación, el modelo UNET también se evalúa

utilizando las métricas estándar de pérdida (**loss**) y precisión (**accuracy**) a través del método **evaluate** de Keras. Esta evaluación dual es fundamental para obtener una comprensión completa del rendimiento del modelo. Mientras que la pérdida proporciona una medida del error general del modelo en el conjunto de testeo, indicando qué tan cerca están las predicciones del modelo de los valores reales, la precisión mide cuántas de las etiquetas predichas por el modelo fueron clasificadas correctamente, ofreciendo una visión directa sobre su eficacia en la detección correcta de la presencia o ausencia de características específicas del disco óptico. En este caso, la función de pérdida utilizada es **binary_crossentropy**, que es particularmente adecuada para problemas de clasificación binaria. Esta función calcula la pérdida de entropía cruzada entre las etiquetas verdaderas y las predicciones del modelo, permitiendo así una evaluación eficaz del rendimiento del modelo en términos de su capacidad para clasificar correctamente cada píxel de las imágenes segmentadas.

En el contexto de la segmentación de imágenes médicas, es crucial aplicar métricas que reflejen con precisión la calidad de la segmentación más allá de las métricas generales de pérdida y precisión. Aquí, la métrica de Intersección sobre la Unión (IoU) es especialmente valiosa. La IoU proporciona una evaluación detallada de la capacidad del modelo para capturar las características anatómicas críticas, comparando cuán bien el área predicha por el modelo se superpone con la área real marcada en las imágenes de referencia. A diferencia de la precisión simple, la IoU toma en cuenta tanto los aciertos como los errores, ofreciendo una medida más completa de la efectividad del modelo. Esto la hace más adecuada para tareas de segmentación donde la precisión en la identificación de estructuras es crucial para diagnósticos médicos precisos.

## 2.3 Extracción de características

Antes de detallar el código y las funciones específicas empleadas para la extracción de características, es relevante mencionar algunas librerías externas que apoyan la implementación de este estudio:

- **cv2** (OpenCV) [10]: Fundamental para el procesamiento de imágenes, abarcando desde la carga y modificación hasta la manipulación avanzada, incluyendo la detección de bordes y otras operaciones geométricas.
- **numpy** [11]: Ofrece soporte para matrices y grandes arrays multidimensionales, junto con una colección de funciones matemáticas que operan eficientemente sobre estos datos.
- **csv** [12]: Facilita la lectura y escritura en formato CSV, permitiendo el almacenamiento y análisis de datos recolectados de manera tabular.

Después de implementar los modelos de segmentación para el disco y la copa óptica, el proceso continúa con la extracción de características morfológicas de estas estructuras segmentadas. Este paso es crucial para el análisis clínico posterior, permitiendo la cuantificación precisa de elementos críticos de la anatomía ocular que son relevantes.

### 2.3.1 Configuración y Preparación de Datos

En esta fase del proyecto, se establece la estructura inicial para la extracción de características morfológicas de las retinografías. El script desarrollado en Python utiliza argumentos de línea de comando para especificar las ubicaciones de los datos, facilitando así la configuración flexible y reproducible de los parámetros de entrada.

El script se configura para recibir los nombres de cuatro carpetas a través de

argumentos de línea de comando:

- **images_folder_name**: Nombre de la carpeta que contiene las retinografías a analizar.
- **segmented_disc_folder_name**: Nombre de la carpeta que contiene las imágenes segmentadas de los discos ópticos.
- **segmented_cup_folder_name**: Nombre de la carpeta que contiene las imágenes segmentadas de las copas ópticas.
- **results_folder_name**: Nombre de la carpeta donde se guardarán los resultados procesados.

### 2.3.2 Cálculo y Descripción de Características Morfológicas

Este apartado se centra en el procesamiento de las imágenes segmentadas del disco y la copa óptica para extraer medidas cuantitativas que son cruciales en la evaluación y diagnóstico del glaucoma.

- **Forma del disco óptico (sOD)**: Es la relación entre la anchura y la altura del disco óptico. Un disco óptico con una forma más circular tendrá una relación cercana a 1, mientras que una forma más elíptica mostrará una relación mayor o menor que 1 dependiendo de su orientación.
- **Forma de la copa óptica (sOC)**: Similar al sOD, esta medida evalúa la relación entre la anchura y la altura de la copa óptica. Las variaciones en la forma de la copa pueden ser un indicador temprano de glaucoma, incluso antes de que se manifiesten otros síntomas.
- **Área del disco óptico (aOD)**: Es la medida del área total del disco óptico dentro de la retina, que incluye tanto la copa como el anillo neurorretiniano.
- **Área de la Copa Óptica (aOC)**: Se refiere al área interna de la copa dentro del disco óptico. El aumento de esta área en el tiempo puede indicar progresión del glaucoma.

Utilizando la función **process_images**, cuyo código se puede ver en la Figura 6, se obtienen las coordenadas y dimensiones de las cajas delimitadoras (bounding boxes) con la función **boundingRrect** que encierran el disco y la copa óptica y a las cuales previamente se les ha obtenido el contorno con la función **findContours**, ambas de **OpenCV**.

```python
def process_images(image_path):
    # Carga una imagen, encuentra el mayor contorno y devuelve la imagen junto con e
    img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    if img is None:
        return None, None, None, None, None

    contours, _ = cv2.findContours(img, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    if not contours:
        return None, None, None, None, None

    max_contour = max(contours, key=cv2.contourArea)
    x, y, w, h = cv2.boundingRect(max_contour)

    return img, x, y, w, h
```

**Figura 6: Función process_images**

Para el cálculo de **sOD** y **sOC** simplemente se divide el ancho entre el alto para el disco (**w_disc/h_disc**) y lo mismo para la copa (**w_cup** / **h_cup),** mientras que para **aOD** y **aOC** se hace uso de la función **countNonZero** de **OpenCV** que cuenta el número de

píxeles distintos de 0 de dicha imagen. La Figura 7 muestra el código que implementa este cálculo.

```
# Calcula la forma del disco óptico sOD
optical_disc_shape = w_disc / h_disc
shape_OD.append(optical_disc_shape)
# Calcular el área utilizando la función countNonZero de OpenCV aOD
area_disc = cv2.countNonZero(img_disc)
areas_OD.append(area_disc)
# Calcula la longitud horizontal HC y vertical VC de la caja delimitadora
diameter_HC.append(w_cup)
diameter_VC.append(h_cup)

# Calcula la forma de la copa óptica sOC
optical_cup_shape = w_cup / h_cup
shape_OC.append(optical_cup_shape)

# Calcular el área utilizando la función countNonZero de OpenCV aOC
area_cup = cv2.countNonZero(img_cup)
areas_OC.append(area_cup)
```

**Figura 7: cálculo de sOD, sOC, aOD y aOC**

- **Tamaño del borde neurorretiniano (aNR)**

El Tamaño del Borde Neurorretiniano (**aNR**) se refiere a la diferencia de área del disco óptico (**aOD**) y la copa óptica ( **aOC**). Un borde neurorretiniano más delgado sugiere una mayor pérdida de tejido neural y es un marcador significativo de riesgo y progresión del glaucoma.

- **Tamaño de la copa óptica en relación con el tamaño del disco óptico. (aCDR)**

La Relación Copa-Disco (aCDR) es una medida que compara el área de la copa óptica (aOC) con el área total del disco óptico (aOD). Un **aCDR** elevado suele ser indicativo de glaucoma, especialmente cuando hay un aumento significativo en la proporción de la copa respecto al disco.

El aCDR se calcula dividiendo el área de la copa óptica entre el área del disco óptico.

- **Relaciones Vertical y Horizontal Copa-Disco (vCDR y hCDR)**

Las relaciones Vertical y Horizontal Copa-Disco (vCDR y hCDR) son métricas que comparan la altura y la anchura de la copa óptica con la del disco óptico, respectivamente.

Las relaciones **vCDR** y **hCDR** se calculan utilizando las dimensiones de las cajas delimitadoras mínimas del disco y la copa óptica. La relación vertical Copa-Disco (**vCDR**) se obtiene dividiendo la altura de la copa por la altura del disco, mientras que la relación horizontal Copa-Disco (**hCDR**) se calcula dividiendo la anchura de la copa por la anchura del disco.

- **Forma del borde neurorretiniano (reglas ISNT)**

Las reglas ISNT (Inferior-Superior-Nasal-Temporal) son utilizadas para evaluar la

forma del borde neurorretiniano en la evaluación del glaucoma, específicamente mirando la configuración del disco óptico y la copa desde diferentes ángulos. Estas reglas ayudan a identificar patrones anómalos en la relación espacial entre la copa y el disco que pueden indicar la presencia de glaucoma. En la Figura 8, se representan las distancias ISNT.

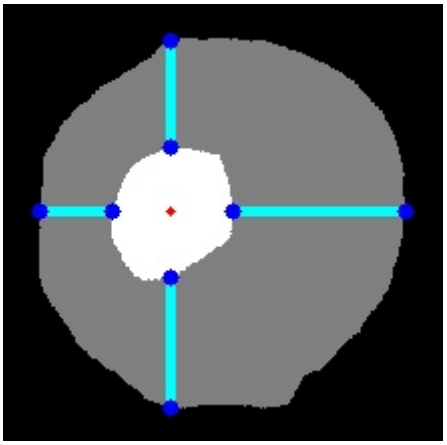


**Figura 8: Ejemplo de ISNT**

El cálculo de la forma del borde neurorretiniano según las reglas **ISNT** se efectúa midiendo las distancias entre los bordes de la copa y del disco óptico en direcciones específicas: 0 grados (Nasal), 90 grados (Inferior), 180 grados (Temporal) y 270 grados (Superior). Este procedimiento se realiza mediante la detección de puntos clave en los bordes de las imágenes segmentadas de la copa y el disco. Para identificar estos puntos, se emplea la función **detect_edge_points**, mostrada en la Figura 9, la cual parte desde el centro de la copa y avanza píxel a píxel hacia las direcciones indicadas hasta detectar un cambio de color. Este cambio indica la transición entre la copa y el disco, permitiendo determinar con precisión los puntos de borde. Posteriormente, se calculan las distancias euclídeas entre los puntos correspondientes de la copa y del disco, proporcionando así las métricas **ISNT** necesarias para la evaluación.

```python
def detect_edge_points(image, center, directions):
    #Detecta los puntos de borde en direcciones especificadas desde el centro de la imagen.
    edge_points = {}
    high, width = image.shape
    for angle in directions:
        # Calculamos el vector de dirección
        radians = np.deg2rad(angle)
        dx = int(np.cos(radians))
        dy = int(np.sin(radians))

        # Iniciar en el centro
        x, y = center

        # Moverse en la dirección hasta encontrar un cambio de color o salir de la imagen
        while 0 <= x < width and 0 <= y < high:
            pixel_actual = image[y, x]
            if angle in [0, 180]:  # Movimiento horizontal
                x += dx
            else:  # Movimiento vertical
                y += dy
            # Si se llega al borde de la imagen sin encontrar un cambio de color
            if not (0 <= x < width and 0 <= y < high):
                # Ajustar x, y para que estén dentro de los límites de la imagen
                x = max(0, min(x, width - 1))
                y = max(0, min(y, high - 1))
                edge_points[angle] = (x, y)
                break

            # Verificar cambio de color
            if image[y, x] != pixel_actual:
                edge_points[angle] = (x, y)
                break

    return edge_points
```

**Figura 9: Función detect_edge_points para el cálculo de la regla ISNT**

- **Ancho del borde más estrecho del anillo neurorretiniano (Wmin.rimNR) y Relación borde-disco (wRDR)**

**Wmin.rimNR** mide la menor distancia entre los bordes del disco y la copa óptica, representando el ancho más estrecho del anillo neurorretiniano. Esta métrica es fundamental para identificar áreas donde el tejido neurorretiniano está potencialmente comprometido, una señal indicativa de daño en casos de glaucoma.

**wRDR** representa la relación entre el ancho mínimo del borde neurorretiniano (**Wmin.rimNR**) y la longitud total del disco en el mismo eje, proporcionando un coeficiente que refleja la proporción del disco óptico afectada por la copa.

Para calcular ambos valores, se implementa un proceso que identifica la distancia mínima entre los contornos de la copa y el disco óptico. Utilizando la función **detect_and_calculate_minimum_distance**, representada en la Figura 10, se explora cada ángulo desde el centro de la copa en un rango completo de 360 grados. Por cada ángulo, se determina el primer punto de cambio de color desde el centro hacia afuera tanto en la imagen del disco como en la de la copa, indicando los límites de cada estructura. En la Figura 11 se muestra un ejemplo visual de estos valores.

```python
def detect_and_calculate_minimum_distance(img_disc, img_cup, center):
    #Calcula la distancia mínima entre puntos correspondientes en imágenes de disco y copa, iniciando des
    min_distance = float('inf')
    best_pair = None
    diameter_points = None
    high, width = img_disc.shape

    for angle in range(360):
        radians = np.deg2rad(angle)
        dx = np.cos(radians)
        dy = -np.sin(radians)

        # Rastrear en la dirección original para ambos discos y copas
        point_disc_1 = trace_until_color_change(img_disc, center, dx, dy, width, high)
        point_cup = trace_until_color_change(img_cup, center, dx, dy, width, high)

        if point_disc_1 and point_cup:
            distance = np.linalg.norm(np.array(point_disc_1) - np.array(point_cup))

            if distance < min_distance:
                min_distance = distance
                best_pair = (point_disc_1, point_cup)  # Guardar los puntos del disco y la copa

                # Si encontramos una nueva mínima distancia, calcular los puntos de diámetro en el disco
                opposite_disc_point = trace_until_color_change(img_disc, center, -dx, -dy, width, high)
                if opposite_disc_point:
                    diameter_points = (point_disc_1, opposite_disc_point)

    return min_distance, best_pair, diameter_points
```

**Figura 10: Función detect_and_calculate_minimum_distance**

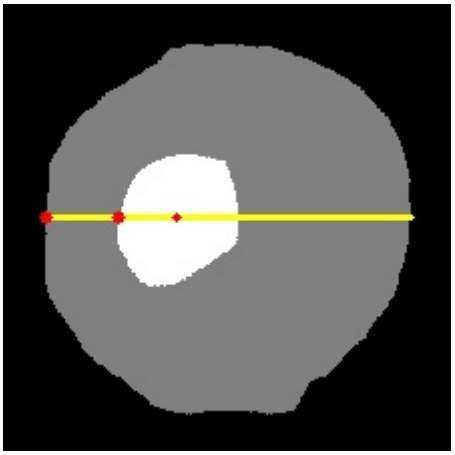


**Figura 11: Ejemplo de Wmin.rimNR y wRDR**

La función **trace_until_color_change** es clave en este proceso: se parte del centro y avanza en una dirección determinada por los componentes de dirección **dx** y **dy** (derivados del ángulo actual) hasta que detecta un cambio de color en la imagen, que señala el borde de la copa o del disco. Dicha función se puede apreciar en la Figura 12.

```python
def trace_until_color_change(image, center, dx, dy, width, high):
    #función que se desplaza desde el centro en las direcciones dx y dy
    x, y = center
    center_color = image[int(y), int(x)]
    while 0 <= x < width and 0 <= y < high:
        if image[int(y), int(x)] != center_color:
            return (int(x), int(y))
        x += dx
        y += dy
    return None
```

**Figura 12: Función trace_until_color_change**

La distancia euclídea entre los puntos de borde correspondientes de la copa y del disco se calcula en cada paso; la mínima de estas distancias se registra como el **Wmin.rimNR**. Para el cálculo de **wRDR,** al final de la función **detect_and_calculate_minimum_distance** cuando se tiene la mínima distancia entre dos puntos de la copa y el disco, se realiza otra llamada a la función **trace_until_color_change** pero esta vez con las direcciones dx y dy en negativo con la intención de que calcule en el mismo eje el extremo contrario del disco dando así el par de puntos correspondientes a la longitud total del disco en el mismo eje.

Finalmente, para calcular **wRDR** solo es necesario calcular la distancia euclídea de este par de puntos y dividir el mínimo ancho de anillo neurorretiniano entre la longitud del disco en el mismo eje.

### 2.3.3 Combinación y Análisis Visual de Imágenes

Para facilitar la interpretación visual de las mediciones realizadas y mejorar la comprensión de las características estructurales del disco y la copa óptica, se ha desarrollado un conjunto de funciones que combinan visualmente las imágenes del disco y la copa, añadiendo marcadores gráficos para resaltar puntos clave.

**Funciones de Combinación de Imágenes**:

- **combine_images_simple**: Esta función básica combina dos imágenes, del disco y la copa, con una opacidad ajustada para permitir la observación de ambas estructuras simultáneamente. Asegura que ambas imágenes sean del mismo tamaño y las convierte a color si son en escala de grises, aplicando una transparencia que facilita la comparación directa entre la copa y el disco. Función descrita en la Figura 13.

```python
def combine_images_simple(image_disc, image_cup):
    # Combina dos imágenes con transparencia, asegurándose de que sean del mismo tamaño
    if image_disc is None or image_cup is None:
        return None

    # Asegurarse de que ambas imágenes tengan el mismo tamaño
    if image_disc.shape != image_cup.shape:
        image_cup = cv2.resize(image_cup, (image_disc.shape[1], image_disc.shape[0]))

    # Convertir a color si son imágenes en escala de grises
    if len(image_disc.shape) == 2:
        image_disc = cv2.cvtColor(image_disc, cv2.COLOR_GRAY2BGR)
    if len(image_cup.shape) == 2:
        image_cup = cv2.cvtColor(image_cup, cv2.COLOR_GRAY2BGR)

    # Combinar imágenes con una transparencia especificada
    alpha = 0.5  # Define la opacidad de la superposición
    combined_image = cv2.addWeighted(image_disc, 1 - alpha, image_cup, alpha, 0)

    return combined_image
```

**Figura 13: Función combine_images_simple**

- **combine_images_ISNT**: Específicamente diseñada para resaltar los puntos según las reglas ISNT (Nasal, Inferior, Temporal, Superior), esta función, mostrada en la Figura 14, no solo combina las imágenes, sino que también marca y conecta puntos clave en el disco y la copa, proporcionando una representación gráfica clara de la relación geométrica entre estas estructuras.

```python
def combine_images_ISNT(image_disc, image_cup, disc_edge_points, cup_edge_points, center_cup):
    #Función para crear la imagen que representa los valores ISNT
    # Asegurarse de que ambas imágenes sean del mismo tamaño
    if image_disc.shape != image_cup.shape:
        image_cup = cv2.resize(image_cup, (image_disc.shape[1], image_disc.shape[0]))

    # Convertir a color si son imágenes en escala de grises
    if len(image_disc.shape) == 2:
        image_disc = cv2.cvtColor(image_disc, cv2.COLOR_GRAY2BGR)
    if len(image_cup.shape) == 2:
        image_cup = cv2.cvtColor(image_cup, cv2.COLOR_GRAY2BGR)

    # Combinar imágenes con opacidad
    alpha = 0.5  # Opacidad de la imagen de la copa
    combined_image = cv2.addWeighted(image_disc, 1 - alpha, image_cup, alpha, 0)

    # Dibujar los puntos en ambas imágenes
    for angle, pt in cup_edge_points.items():
        if angle in disc_edge_points:
            # Dibujar líneas entre los puntos de la copa y del disco
            cv2.line(combined_image, tuple(pt), tuple(disc_edge_points[angle]), (255, 255, 0), 3)
        cv2.circle(combined_image, tuple(pt), 4, (255, 0, 0), -1)  # Puntos verdes en la copa
    for angle, pt in disc_edge_points.items():
        cv2.circle(combined_image, tuple(pt), 4, (255, 0, 0), -1)  # Puntos rojos en el disco
    # Dibujar el centro de la copa
    cv2.circle(combined_image, tuple(center_cup), 2, (0, 0,255), -1)  # Punto cian para marcar el

    return combined_image
```

**Figura 14: Función combine_images_ISNT**

- **combine_images_MinANR**: Centrada en destacar el ancho del borde neurorretiniano más estrecho y el diámetro neurorretiniano. Utiliza la opacidad para superponer imágenes del disco y la copa, mientras que dibuja y resalta el mínimo diámetro encontrado, proporcionando una perspectiva visual directa del estado del anillo neurorretiniano. La función se puede apreciar en la Figura 15.

```python
def combine_images_MinANR(image_disc, image_cup, diameter_points, final_disc_diameter, center_cup):
    # Combina imágenes de discos y copas, destacando el mínimo diámetro y el centro de la copa para análisis.
    # Asegurarse de que ambas imágenes sean del mismo tamaño
    if image_disc.shape != image_cup.shape:
        image_cup = cv2.resize(image_cup, (image_disc.shape[1], image_disc.shape[0]))

    # Convertir a color si son imágenes en escala de grises
    if len(image_disc.shape) == 2:
        image_disc = cv2.cvtColor(image_disc, cv2.COLOR_GRAY2BGR)
    if len(image_cup.shape) == 2:
        image_cup = cv2.cvtColor(image_cup, cv2.COLOR_GRAY2BGR)

    # Combinar imágenes con opacidad
    alpha = 0.5  # Opacidad de la imagen de la copa
    combined_image = cv2.addWeighted(image_disc, 1 - alpha, image_cup, alpha, 0)

    # Dibujar puntos del diámetro más pequeño y línea
    if diameter_points:
        cv2.line(combined_image, tuple(final_disc_diameter[0]), tuple(final_disc_diameter[1]), (0, 255, 255), 2)
        cv2.circle(combined_image, tuple(diameter_points[0]), 3, (0, 0, 255), -1)  # Punto azul en el disco
        cv2.circle(combined_image, tuple(diameter_points[1]), 3, (0, 0, 255), -1)  # Punto azul en la copa
    # Dibujar el centro de la copa
    cv2.circle(combined_image, tuple(center_cup), 2, (0, 0,255), -1)  # Punto cian para marcar el centro del disco

    return combined_image
```

**Figura 15: Función combine_images_MinANR**

### 2.3.4 Almacenamiento de Resultados

Se crea un archivo CSV, **datos.csv**, en el directorio de resultados, donde se almacenan las métricas calculadas para cada imagen. Este archivo incluye columnas para el nombre del archivo, las catorce características morfológicas descritas, junto con la etiqueta que clasifica cada imagen (“normal” o “glaucoma”).

El proceso inicia escribiendo los nombres de las columnas y luego se añaden los datos de cada imagen. Se verifica que los valores de las características cuyo cálculo dependían de encontrar un centro válido para el disco o copa sean válidos (not a number, NaN) antes de su inclusión en el archivo. Esto asegura que los datos almacenados sean coherentes y completos.

Para facilitar la revisión rápida de los resultados y su comparación, se implementa una función, **show_table**, que muestra los datos en formato de tabla directamente en la consola.

## 2.4 Clasificación mediante árbol de decisión

Un árbol de decisión es un modelo de aprendizaje supervisado que se utiliza para clasificar datos o predecir resultados basándose en reglas claras y simples, estructuradas de manera jerárquica y secuencial. Estos modelos son especialmente valorados por su

transparencia y facilidad de interpretación, ya que cada decisión tomada por el modelo puede ser trazada y comprendida a través de una serie de preguntas binarias, lo que resulta en un árbol de bifurcaciones donde cada nodo representa una decisión basada en una característica de los datos.

El modelo fue configurado para clasificar imágenes de retinografías en categorías relacionadas con la presencia o ausencia de glaucoma haciendo uso de las características extraídas anteriormente.

Para entrenar el modelo se usaron las características extraídas del conjunto de entrenamiento de RIMONE-DL, así como sus correspondientes etiquetas (“normal” y “glaucoma”).

La configuración del modelo de árbol de decisión se realizó mediante la herramienta **GridSearchCV** [13] de la biblioteca **scikit-learn**. Esta herramienta permite una búsqueda exhaustiva de los mejores parámetros de un modelo a través de una validación cruzada estructurada. La validación cruzada es una técnica de evaluación de modelos que busca mejorar la fiabilidad de las estimaciones de rendimiento, dividiendo los datos en múltiples subconjuntos o "pliegues". En cada ciclo de validación cruzada, un subconjunto diferente se reserva como conjunto de testeo, mientras que los restantes se utilizan para entrenar el modelo. Este proceso se repite hasta que cada subconjunto ha servido tanto de entrenamiento como de testeo.

Los parámetros más influyentes en un árbol de decisión son la profundidad máxima del árbol (**max_depth**) y el número mínimo de muestras requeridas en una hoja (**min_samples_leaf**). Estos parámetros son determinantes en el control de la complejidad del modelo y su capacidad para generalizar a nuevos datos:

- **max_depth**: Controla la máxima longitud entre la raíz del árbol y cualquier hoja final. Un árbol profundo puede captar detalles muy finos de los datos de entrenamiento, pero corre el riesgo de sobreajuste. Un árbol menos profundo puede no capturar toda la estructura subyacente de los datos, afectando el rendimiento del modelo.
- **min_samples_leaf**: Este parámetro establece el número mínimo de muestras que debe tener una hoja. Valores mayores en este parámetro pueden ayudar a mejorar la generalización del modelo al evitar divisiones que resulten en hojas con muy pocos datos.

**GridSearchCV** fue configurado con un rango de valores posibles para **max_depth** y **min_samples_leaf**, evaluando la precisión del árbol de decisión bajo diferentes combinaciones de estos parámetros mediante validación cruzada de 5 pliegues. Este enfoque garantiza que la evaluación de cada configuración sea robusta y representativa del rendimiento general del modelo (véase Figura 16):

```
dt = DecisionTreeClassifier(random_state=42) # Inicializa el árbol de decisión
param_grid=[{"max_depth":[None,3,4,5,10],"min_samples_leaf":[1,2,3,4,5,8,10,15]}]
dt_GS = GridSearchCV(dt, param_grid, cv=5,
                           scoring='accuracy',
                           return_train_score=True)
dt_GS.fit(X_train, y_train)

print("Best parameters ",dt_GS.best_params_) # Muestra los mejores parámetros encontrados
print("Score :",dt_GS.best_score_) # Muestra el mejor puntaje de precisión
```

**Figura 16: Entrenamiento del árbol de decisión con GridSearchCV**

Una vez completada la búsqueda, **GridSearchCV** identifica la configuración óptima del modelo. Estos resultados indican la configuración que maximiza la precisión del

modelo en datos no vistos, asegurando que el árbol de decisión está bien ajustado para ofrecer un rendimiento equilibrado entre precisión y capacidad de generalización.

### 2.4.1 Selección del umbral de decisión del Modelo

Una vez entrenado el modelo de árbol de decisión utilizando **GridSearchCV** para optimizar sus parámetros, la siguiente fase crucial es la selección del umbral de decisión del mismo.

Para evaluar la efectividad del modelo de árbol de decisión, se utilizan dos herramientas estadísticas importantes: la curva ROC (Receiver Operating Characteristic) [14] y la curva de precisión-recall. Ambas curvas son vitales para medir la capacidad predictiva del modelo bajo diferentes umbrales de clasificación.

- **Curva ROC**: Esta curva es una representación gráfica que muestra el rendimiento de un modelo de clasificación en todos los umbrales de clasificación. Representa la Tasa de Verdaderos Positivos (Sensibilidad) contra la Tasa de Falsos Positivos (1-Especificidad) para diferentes puntos de corte. Un área bajo la curva ROC (AUC) cercana a 1 indica un modelo muy bueno, mientras que un AUC cercano a 0.5 sugiere un rendimiento no mejor que el azar.
- **Curva de Precisión-Recall**: Esta curva muestra la relación entre la precisión del modelo (proporción de verdaderos positivos entre todas las clasificaciones positivas realizadas) y el recall (proporción de verdaderos positivos entre todos los positivos reales). Es especialmente útil en situaciones donde las clases están muy desbalanceadas, como puede ser el caso en la detección médica de enfermedades, donde el número de casos positivos (enfermedades) es mucho menor que el número de negativos (sanos).

El análisis de los resultados del modelo de árbol de decisión se realiza utilizando las curvas ROC y de precisión-recall, junto con varias métricas de evaluación. Estas métricas incluyen la precisión, recall, F1-score y precisión balanceada, que proporcionan una visión integral del rendimiento del modelo..

Se utilizan funciones para calcular los umbrales óptimos que maximizan las métricas de precisión y recall. Las funciones **cutoff_youdens_j** y **cutoff_f1_score** se utilizan para encontrar los umbrales que maximizan el índice de Youden y el F1-score, respectivamente.

- **Índice de Youden**: El índice de Youden se calcula como $J = \text{Sensibilidad} + \text{Especificidad} - 1$. Este índice proporciona un balance entre la sensibilidad (tasa de verdaderos positivos) y la especificidad (tasa de verdaderos negativos), maximizando ambos simultáneamente [15].Esto se ilustra en la Figura 17.

```python
#Esta función calcula el umbral óptimo utilizando el índice de Youden,
#que maximiza la diferencia entre la tasa verdadera positiva y la tasa falsa positiva.
def cutoff_youdens_j(fpr,tpr,thresholds):
    j_scores = tpr-fpr
    j_ordered = sorted(zip(j_scores,thresholds,tpr,fpr))
    return [j_ordered[-1][1],j_ordered[-1][2],j_ordered[-1][3] ]
```

**Figura 17: Función cutoff_youdens_jl**

- **F1-score**: El F1-score es la media armónica de la precisión y el recall, y se utiliza para encontrar un balance entre ambos [16]. Se calcula como F1= 2 × ( (Precisión x Recall) / (Precisión + Recall) ). Esta función se detalla en la Figura 18.

```python
#Calcula el umbral que maximiza el puntaje F1, que es el promedio armónico de la precisión y el recall.
def cutoff_f1_score(precisions, recalls, thresholds):
    np.seterr(divide='ignore', invalid='ignore')
    f1_scores = 2* precisions*recalls/(precisions+recalls)
    np.seterr(divide='warn', invalid='warn')
    f1_score_ordered = sorted(zip(f1_scores,thresholds,precisions,recalls))
    f1_score_ordered_fix = list(filter(lambda x: not(np.isnan(x[0])), f1_score_ordered))
    #for e in f1_score_ordered_fix:
     # print(e)
    return [f1_score_ordered_fix[-1][1], f1_score_ordered_fix[-1][2],f1_score_ordered_fix[-1][3]]
```

**Figura 18: Función cutoff_f1_score**

Después de determinar los umbrales óptimos, el código evalúa el modelo utilizando estos umbrales y calcula varias métricas de rendimiento:

- **Curvas ROC y de Precisión-Recall**: Se calculan y representan las curvas ROC y de precisión-recall para el modelo de árbol de decisión.
- **Umbrales Óptimos**: Utiliza las funciones **cutoff_youdens_j** y **cutoff_f1_score** para determinar los umbrales que maximizan el índice de Youden y el F1-score.
- **Evaluación de Métricas**: Para cada umbral, se calculan las métricas de evaluación como precisión, recall, F1-score, exactitud y el índice de Youden.
- **Matrices de Confusión**: Se generan y representan matrices de confusión para visualizar el rendimiento del modelo en términos de verdaderos positivos, falsos positivos, verdaderos negativos y falsos negativos.

Una vez configurado y optimizado el modelo de árbol de decisión con los datos de validación, es crucial evaluar su rendimiento utilizando un conjunto de datos de testeo independiente. Esta evaluación permite verificar la capacidad del modelo para generalizar a nuevos datos no vistos durante el entrenamiento.

El modelo se evaluó utilizando las métricas de precisión, el recall, el F1-score y el índice de Youden en el conjunto de testeo eligiendo el umbral que maximiza las métricas en el conjunto de validación.

# Capítulo 3 Resultados

Este capítulo presenta los resultados obtenidos en las diferentes etapas del proyecto, abarcando la segmentación de imágenes, la extracción de características y la clasificación utilizando un modelo de árbol de decisión.

## 3.1 Segmentación de Imágenes

Para esta tarea, se utilizó un modelo UNET, tanto para segmentar el disco óptico como la copa óptica. A continuación se presentan los resultados visuales y métricas de evaluación obtenidas mediante este enfoque.

Como se explicó en el capítulo de desarrollo, las predicciones obtenidas fueron sometidas a un proceso de umbralización para generar imágenes binarizadas, que posteriormente se utilizaron para extraer características. En el caso del disco óptico, el valor de umbral de 0.35 fue el que proporcionó los mejores resultados, obteniendo una métrica IoU media en el conjunto de testeo de 0.89 y una precisión de 0.93.

A continuación, se presenta un ejemplo de resultado visual en la Figura 19, que muestra la retinografía original, su máscara original y la imagen segmentada. En este caso, se obtuvo un valor de IoU de 0.96, siendo uno de los mejores resultados.

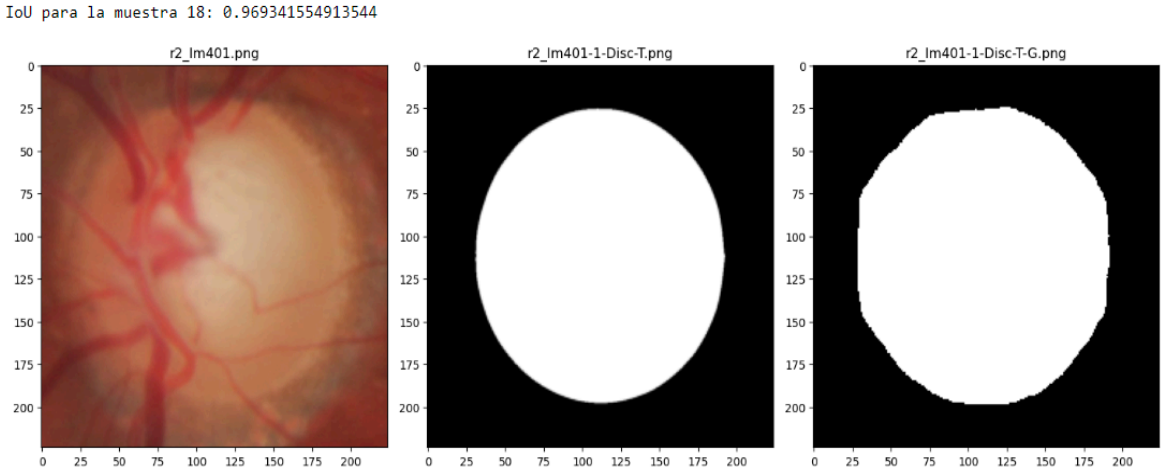



**Figura 19: Representación de imagen original, máscara original y máscara segmentada del disco óptico**

Para la copa óptica, el valor de umbral de 0.50 proporcionó una métrica IoU promedio de 0.56 en el conjunto de testeo y una precisión de 0.93. En la Figura 20, se muestra una representación visual de la segmentación de la copa, donde se puede apreciar la comparación entre la imagen original, su máscara original y la imagen segmentada.

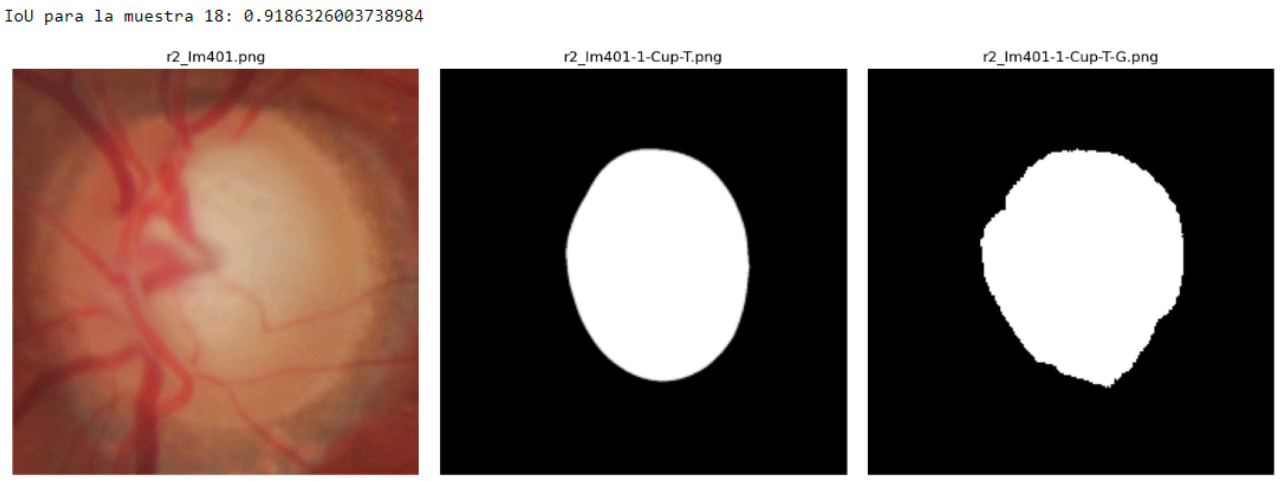



**Figura 20: Representación de imagen original, máscara original y máscara segmentada de la copa óptica**

Finalmente, las imágenes segmentadas de la copa y del disco que se van a utilizar en el proceso de extracción de características han sido almacenadas en los directorios de resultados **mascaras_copa_UNET** y **mascaras_disco_UNET**.

## 3.2 Extracción de características

Los resultados de la extracción de las 14 características se almacenaron en un archivo CSV, donde cada fila representa una imagen y contiene las 14 características separadas por comas y un valor denominado etiqueta con los valores glaucoma o normal. Este archivo es fundamental, ya que posteriormente se utiliza para la clasificación de las retinografías mediante el modelo de árbol de decisión.

Para un mejor análisis y estructura de los datos, la carpeta generada al ejecutar el script de extracción de características contiene, además del archivo CSV, una carpeta individual para cada imagen. Dentro de cada una de estas carpetas, se encuentra un archivo TXT con las 14 características individuales de la imagen correspondiente y la etiqueta glaucoma o normal (ver Tabla 1) y otra carpeta de imágenes que contiene una representación visual de las características (ver Figura 21).

| Características | | | | | |
|---|---|---|---|---|---|
| aOD | 19824 | aOC | 5673 | Etiqueta | normal |
| sOD | 0.96875 | sOC | 1.2345679012345678 | | |
| aNR | 14151 | aCDR | 0.2861682808716707 | | |
| N_ISNT | 28.0 | vCDR | 0.50625 | | |
| I_ISNT | 43.0 | hCDR | 0.6451612903225806 | | |
| T_ISNT | 25.0 | Wmin_rim | 24.331050121192877 | | |
| S_ISNT | 38.0 | wRDR | 0.1571053562892618 | | |

**Tabla 1: Ejemplo de resultados representados de una imágen**

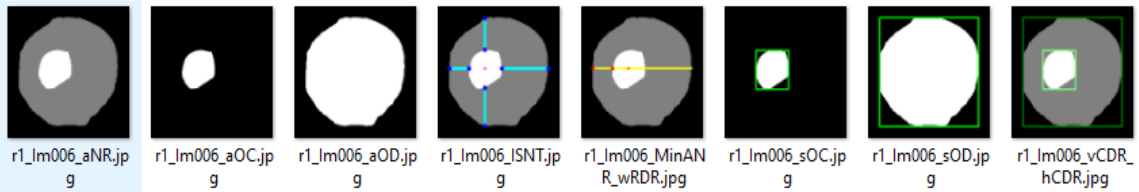



**Figura 21: Contenido del directorio con las imágenes de las características**

Al ejecutar el script, los datos del CSV también se muestran en pantalla en forma de tabla, lo que permite una visualización inmediata de cada valor.

Se quiso cuantificar la diferencia entre las características obtenidas con las máscaras originales de RIM-ONE en el conjunto de test con respecto a las obtenidas con las máscaras generadas automáticamente con el modelo UNet. Para ello, se usaron dos métricas de error: el error cuadrático medio (**RMSE**) y el coeficiente de determinación ($\mathbf{R^2}$). Los resultados se muestran en la Tabla 2:

El RMSE se calcula como la raíz cuadrada del promedio de los cuadrados de las diferencias entre los valores predichos y los valores originales. Esta métrica proporciona una medida de la magnitud de los errores en las predicciones del modelo.

El coeficiente de determinación ($R^2$) se calcula como uno menos la proporción de la suma de los cuadrados de los residuos (diferencias entre los valores originales y predichos) respecto a la suma total de los cuadrados (diferencias entre los valores originales y su media). Esta métrica indica qué tan bien los valores predichos explican la variabilidad de los valores originales.

| Característica | RMSE | $R^2$ | Rango originales | Rango segmentadas |
|---|---|---|---|---|
| aOD | 2047.6764 | 0.2301 | [13640, 26020] | [18369, 24475] |
| sOD | 0.0727 | 0.1070 | [0.6835, 1.1933] | [0.8941, 1.0178] |
| aNR | 3155.6610 | 0.4446 | [3059, 25332] | [9575, 21535] |
| ISNT_N | 20.5828 | -0.3069 | [5.0, 93.0] | [13.0, 86.0] |
| ISNT_I | 14.9346 | 0.4629 | [4.0, 91.0] | [12.0, 81.0] |
| ISNT_T | 18.0939 | 0.1257 | [5.0, 94.0] | [23.0, 95.0] |
| ISNT_S | 12.4441 | 0.4924 | [2.0, 84.0] | [14.0, 84.0] |
| aOC | 2888.5582 | 0.5260 | [165, 17131] | [580, 13175] |
| sOC | 0.2976 | -0.7210 | [0.4429, 2.0] | [0.3158, 1.76] |
| aCDR | 0.1323 | 0.5205 | [0.0065, 0.8485] | [0.0273, 0.5651] |
| vCDR | 0.1495 | 0.4968 | [0.0842, 0.9756] | [0.1667, 0.8284] |
| hCDR | 0.1627 | 0.2350 | [0.0581, 0.9286] | [0.1118, 0.7546] |
| Wmin.rimNR | 13.0642 | 0.1995 | [0.0, 74.3303] | [6.4031, 54.4243] |
| wRDR | 0.0877 | 0.1496 | [0.0, 0.4501] | [0.0420, 0.3174] |

**Tabla 2: Diferencias entre las características obtenidas con las máscaras originales en el conjunto de test de RIMONE-DL y con las máscaras generadas automáticamente con UNet.**

Los errores cometidos en las características de la copa óptica son mayores que los del disco óptico. Esto se debe a que la segmentación de la copa es más difícil y presenta una mayor variabilidad, lo que se refleja en mayores valores de RMSE y menores valores de $R^2$ para las características de la copa. Por ejemplo, la forma de la copa óptica (**sOC**) tiene un RMSE bajo pero un $R^2$ negativo, indicando que, aunque las predicciones son cercanas, no siguen bien la variabilidad de los datos.

Aunque el modelo UNET implementado ofrece predicciones razonables para algunas características, hay áreas donde las predicciones presentan errores significativos. Estos resultados sugieren la necesidad de mejoras adicionales en el modelo para aumentar la precisión y fiabilidad de las segmentaciones.

## 3.3 Clasificación con árbol de decisión

La clasificación de las retinografías se realizó utilizando un modelo de árbol de decisión, optimizado y evaluado con las características extraídas de las imágenes segmentadas. Esta sección presenta los resultados obtenidos, incluyendo el árbol de decisión generado y las métricas de rendimiento.

El modelo de árbol de decisión se entrenó y optimizó utilizando **GridSearchCV** para encontrar los mejores parámetros. Los parámetros óptimos encontrados fueron **max_depth**: 3 y **min_samples_leaf**: 4, con un puntaje de **precisión** de 0.8879. Una vez entrenado, se generó el árbol de decisión final, que se muestra en la Figura 22.

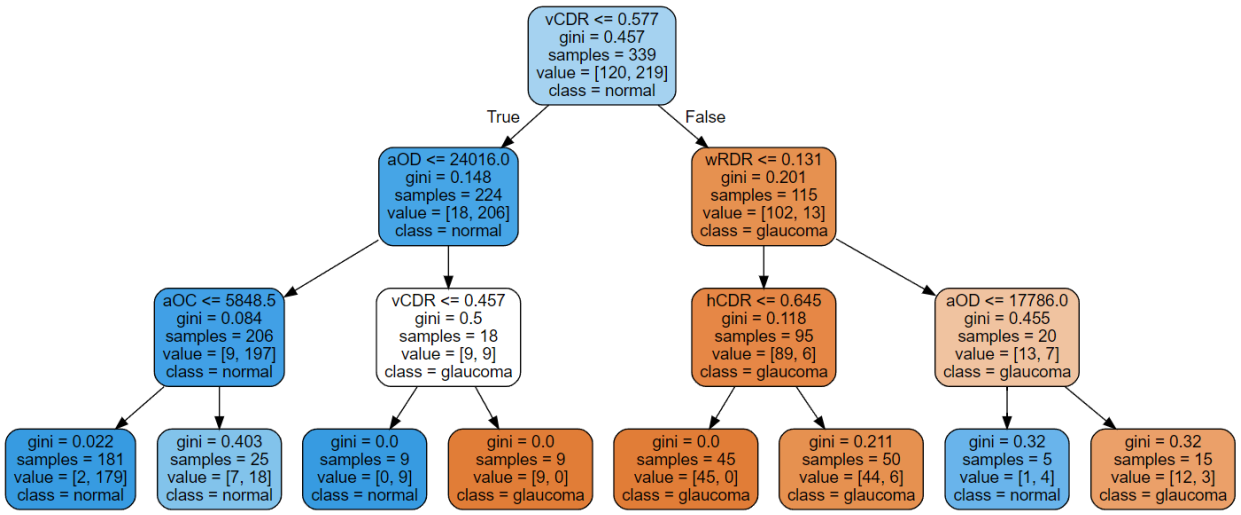



**Figura 22: Árbol de decisión**

**Este árbol de decisión se interpreta de la siguiente manera:**

1. **Nodo Raíz (vCDR <= 0.577):**

- Gini: 0.457
- Muestras: 339
- Valores: [120, 219] (120 casos de glaucoma, 219 normales)
- Clase Predicha: Normal
- Este nodo indica que si la relación vertical copa/disco (vCDR) es menor o igual a 0.577, el modelo clasifica la imagen como normal. De lo contrario, se mueve al subárbol derecho.

2. **Subnodo Izquierdo (aOD <= 24016.0):**

- Gini: 0.148
- Muestras: 224
- Valores: [18, 206]
- Clase Predicha: Normal
- Este nodo indica que, para las imágenes con vCDR <= 0.577, si el área del disco óptico (aOD) es menor o igual a 24016, el modelo sigue clasificando la imagen como normal.

3. **Subnodo Derecho (wRDR <= 0.131):**

- Gini: 0.201
- Muestras: 115
- Valores: [102, 13]
- Clase Predicha: Glaucoma
- Para las imágenes con vCDR > 0.577, si la relación del radio del borde del disco (wRDR) es menor o igual a 0.131, el modelo clasifica la imagen como glaucoma. De lo contrario, se mueve al siguiente nodo.

4. **Subnodo del Subnodo Izquierdo (aOC <= 5848.5):**

- Gini: 0.084
- Muestras: 206
- Valores: [9, 197]
- Clase Predicha: Normal
- Para las imágenes con aOD <= 24016.0, si el área de la copa óptica (aOC) es menor o igual a 5848.5, el modelo sigue clasificando la imagen como normal.

5. **Subnodo del Subnodo Derecho (hCDR <= 0.645):**

- Gini: 0.118
- Muestras: 95
- Valores: [89, 6]
- Clase Predicha: Glaucoma
- Para las imágenes con wRDR > 0.131, si la relación horizontal copa/disco (hCDR) es menor o igual a 0.645, el modelo sigue clasificando la imagen como glaucoma.

6. **Hojas del Árbol:**

- Cada hoja muestra la pureza del nodo (Gini), el número de muestras, la distribución de las clases, y la clase predicha.
- Por ejemplo, una hoja con Gini = 0.0, muestras = 9, valores = [0, 9], y clase = glaucoma, indica que todas las 9 muestras en ese nodo se clasifican correctamente como glaucoma.

**Errores de Clasificación**

El árbol de decisión también permite observar cuántas imágenes fueron clasificadas incorrectamente:

- En el nodo raíz, 120 imágenes de glaucoma y 219 imágenes normales se clasificaron en función de vCDR <= 0.577.
- En el subnodo izquierdo (aOD <= 24016.0), de las 224 imágenes, 18 casos de glaucoma fueron predichos incorrectamente como normales.
- En el subnodo derecho (wRDR <= 0.131), de las 115 imágenes, 13 casos normales fueron predichos incorrectamente como glaucoma.

**Importancia de las Características**

El modelo de árbol de decisión también permitió identificar las características más importantes para la clasificación. Las importancias de las características fueron las siguientes, como se muestra en la Tabla 3.

| Características | | | |
|---|---|---|---|
| aOD | 0.0773 | aOC | 0.02561 |
| sOD | 0.0 | sOC | 0.0 |
| aNR | 14151 | aCDR | 0.0 |
| N_ISNT | 0.0 | vCDR | 0.8696 |
| I_ISNT | 0.0 | hCDR | 0.0054 |
| T_ISNT | 0.0 | Wmin_rim | 0.0 |
| S_ISNT | 0.0 | wRDR | 0.10219 |

**Tabla 3: Importancia de las Características**

De estas características, la relación vertical copa/disco (**vCDR**) fue la más importante, con una importancia de 0.8696, seguida por el área del disco óptico (**aOD**) y el área de la copa óptica (**aOC**).

### 3.3.1 Curvas ROC y de Precisión-Recall

Las curvas ROC (Receiver Operating Characteristic) y de precisión-recall se generaron para evaluar el rendimiento del modelo en diferentes umbrales de clasificación. Estas curvas proporcionan una visión clara de la capacidad del modelo para distinguir entre casos de glaucoma y no glaucoma.

La curva ROC muestra que el modelo tiene un área bajo la curva (AUC) de 0.92. Esto indica que el modelo tiene una alta capacidad para distinguir entre casos positivos (glaucoma) y negativos (normales), como se aprecia en la Figura 23.

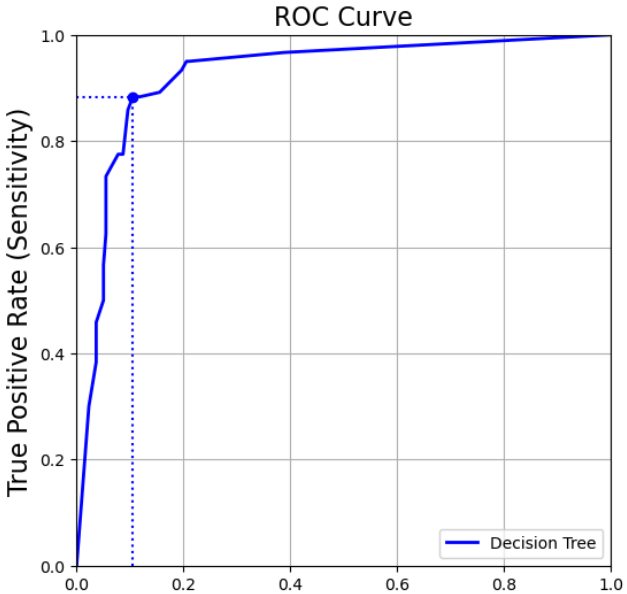



**Figura 23: Curva ROC**

La curva de precisión-recall muestra que el modelo mantiene una alta precisión y

recall en diferentes umbrales, lo que indica que el modelo es eficaz para identificar correctamente los casos de glaucoma, como se muestra en la Figura 24.

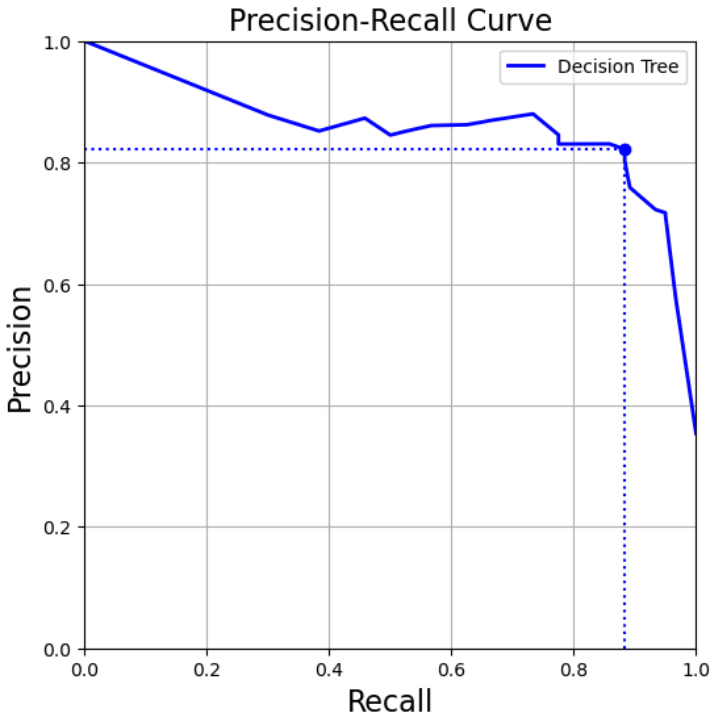



**Figura 24: Curva Precisión-Recall**

### Matriz de Confusión

La matriz de confusión proporciona una visión detallada del rendimiento del modelo en términos de verdaderos positivos (casos positivos predichos como positivos), verdaderos negativos (casos negativos predichos como negativos), falsos positivos (casos negativos predichos como positivos) y falsos negativos (casos positivos predichos como negativos). La matriz de confusión muestra en las filas los casos reales y en las columnas los casos predichos.

### Evaluación de Umbrales con las predicciones de la validación cruzada

Además del umbral de decisión a 0.5 por defecto, el modelo se evaluó utilizando diferentes umbrales para maximizar el F1-score y el índice Youden, en ambos casos obteniendo un umbral de 0.3.

- AUC: 0.92
- Umbral de 0.50:
  - Precisión (Accuracy): 0.89
  - Precisión (Precision): 0.83
  - Recall: 0.86
  - Especificidad: 0.86
  - F1-score: 0.84
  - Índice de Youden: 0.72

| Matriz de Confusión | glaucoma | normal |
|---|---|---|

| | | |
|---|---|---|
| **glaucoma** | 103 (TP) | 17 (FN) |
| **normal** | 21 (FP) | 198 (TN) |

**Tabla 4: Matriz de confusión con umbral 0.5**

- Umbral de 0.30:
  - Precisión (Accuracy): 0.89
  - Precisión (Precision): 0.82
  - Recall: 0.88
  - Especificidad: 0.88
  - F1-score: 0.85
  - Índice de Youden: 0.77

| Matriz de Confusión | glaucoma | normal |
|---|---|---|
| **glaucoma** | 106 (TP) | 14 (FN) |
| **normal** | 23 (FP) | 196 (TN) |

**Tabla 5: Matriz de confusión con umbral 0.3**

**Ventajas:**

- Este umbral es menos estricto, lo que significa que está más inclinado a clasificar un caso como glaucoma, reduciendo el número de falsos negativos (individuos con glaucoma diagnosticados incorrectamente como sanos) a costa de aumentar el número de falsos positivos.

En resumen, la elección del umbral depende del contexto clínico y de los objetivos específicos del diagnóstico. Si es más importante minimizar los falsos negativos y asegurar que menos casos de glaucoma se pasen por alto, el umbral de 0.30 es más adecuado. Sin embargo, si el objetivo es maximizar la precisión del diagnóstico y reducir los falsos positivos, el umbral de 0.50 es preferible.El umbral de decisión a 0.3 ofrece un mejor índice y youden y F1-score en las predicciones de validación y, por tanto, es el que se usará en el conjunto de testeo.

**Evaluación de Umbrales en el Conjunto de testeo**

El modelo de árbol de decisión se evaluó en el conjunto de testeo utilizando el umbral 0.30 obteniéndose los siguientes resultados:

- Precisión (Accuracy): 0.84
- Precisión (Precision): 0.76
- Recall: 0.81
- Especificidad: 0.81
- F1-score: 0.79
- Índice de Youden: 0.62

# Capítulo 4 Conclusiones y líneas futuras

## 4.1 Conclusiones

En este trabajo, se desarrolló y evaluó un sistema para la detección de glaucoma a partir de imágenes de retinografías. Se implementaron varias etapas, incluyendo la segmentación del disco óptico y la copa óptica mediante modelos de redes neuronales convolucionales, la extracción de características relevantes y la clasificación utilizando un modelo de árbol de decisión.

Los principales hallazgos y conclusiones de este trabajo son los siguientes:

- **Segmentación**: Se utilizó un modelo UNET para segmentar tanto el disco óptico como la copa óptica. Los resultados indicaron que, aunque el modelo UNET proporciona una segmentación adecuada, existen oportunidades de mejora, especialmente en la segmentación de la copa óptica donde estructuras como los vasos sanguíneos pueden interferir con la precisión de la segmentación.
- **Extracción de Características**: Se extrajeron 14 características clave de las imágenes segmentadas, las cuales se almacenaron en un archivo CSV para su posterior análisis y clasificación. La comparación con las máscaras originales mostró que la segmentación automática tiene un rendimiento aceptable, aunque hay margen para mejorar la precisión de las características extraídas.
- **Clasificación**: Se utilizó un modelo de árbol de decisión optimizado para clasificar las retinografías en casos de glaucoma y normales. Los resultados mostraron una precisión general del 84% en el conjunto de prueba, con un área bajo la curva (AUC) de 0.85. El modelo presenta un equilibrio entre precisión y recall, pero es necesario ajustar los umbrales de decisión según el contexto clínico para optimizar la detección y reducir tanto los falsos positivos como los falsos negativos.

Este trabajo demuestra el potencial de los enfoques automáticos basados en aprendizaje automático para la detección de glaucoma, aunque también resalta la necesidad de continuar mejorando los modelos y técnicas utilizadas.

## 4.2 Líneas Futuras

Para mejorar y ampliar el trabajo realizado, se identificaron varias líneas futuras de investigación y desarrollo:

1. **Segmentación de Vasos Sanguíneos**: Incorporar la segmentación de los vasos sanguíneos en las retinografías podría proporcionar características adicionales que

son relevantes para el análisis del glaucoma. Esta información podría mejorar la precisión de la detección y proporcionar una visión más completa del estado de la retina.

2. **Análisis con otros Modelos de CNN:** Realizar más análisis utilizando diferentes modelos de redes neuronales convolucionales (CNN) distintos a UNET para la segmentación del disco y la copa óptica.
3. **Desarrollo de un Aplicativo Web**: Desarrollar un aplicativo web para la extracción de características de retinografías que haga el uso del script más sencillo e intuitivo. Este aplicativo permitiría a los usuarios cargar imágenes de retinografías directamente desde directorios.

Estas líneas futuras no solo mejorarán la precisión y la utilidad del sistema desarrollado, sino que también facilitarán su adopción y uso en entornos clínicos y de investigación. Continuar con esta investigación contribuirá significativamente al campo de la detección automática de glaucoma y a la mejora de los métodos de diagnóstico ocular.

# Capítulo 5 Summary and Conclusions

## 5.1 Conclusions

In this work, a system for glaucoma detection from retinal images was developed and evaluated. Several stages were implemented, including the segmentation of the optic disc and optic cup using convolutional neural network models, the extraction of relevant features, and classification using a decision tree model.

The main findings and conclusions of this work are as follows:

- **Segmentation**: A UNET model was used to segment both the optic disc and the optic cup. The results indicated that while the UNET model provides adequate segmentation, there are opportunities for improvement, especially in the segmentation of the optic cup where structures like blood vessels can interfere with segmentation accuracy.
- **Feature Extraction**: Fourteen key features were extracted from the segmented images and stored in a CSV file for further analysis and classification. Comparison with the original masks showed that the automatic segmentation has acceptable performance, although there is room to improve the accuracy of the extracted features.
- **Classification**: An optimized decision tree model was used to classify the retinal images into glaucoma and normal cases. The results showed an overall accuracy of 84% on the test set, with an area under the curve (AUC) of 0.85. The model presents a balance between precision and recall, but it is necessary to adjust the decision thresholds according to the clinical context to optimize detection and reduce both false positives and false negatives.

This work demonstrates the potential of automated machine learning-based approaches for glaucoma detection, while also highlighting the need for continued improvement of the models and techniques used.

## 5.2 Future Work

To improve and expand the work carried out, several future research and development lines have been identified:

1. **Segmentation of Blood Vessels**: Incorporating the segmentation of blood vessels in the fundus images could provide additional relevant features for glaucoma analysis. This information could improve detection accuracy and provide a more

comprehensive view of the retina's condition.

2. **Analysis with Other CNN Models**: Conduct further analysis using different convolutional neural network (CNN) models other than UNET for the segmentation of the optic disc and optic cup. Models such as DeepLab, SegNet, or Mask R-CNN might offer more precise and robust segmentations.
3. **Development of a Web Application**: Develop a web application for retinal image feature extraction that makes the use of the script simpler and more intuitive. This application would allow users to upload fundus images, process them automatically, and obtain the relevant features efficiently and accessibly.

These future work lines will not only improve the accuracy and utility of the developed system but also facilitate its adoption and use in clinical and research environments. Continuing this research will significantly contribute to the field of automatic glaucoma detection and the improvement of ocular diagnostic methods.

# Capítulo 6 Presupuesto

A continuación se presenta el presupuesto estimado para el desarrollo del sistema de detección de glaucoma, considerando aproximadamente 300 horas a una tarifa de 30 euros por hora.

| Tarea | Horas | Tarifa por Hora (EUR) | Coste (EUR) |
|---|---|---|---|
| Desarrollo | 210 | 30 | 6.300 |
| Investigación Médica | 5 | 30 | 150 |
| Investigación de Tecnologías | 15 | 30 | 450 |
| Redacción | 40 | 30 | 1.200 |
| Pruebas y Validación | 30 | 30 | 900 |
| **Total** | **300** | **30** | **9.000** |

**Tabla 6: Presupuesto final**

# Capítulo 7 Anexo I

El código correspondiente al desarrollo del trabajo de fin de grado, incluyendo la segmentación de imágenes, extracción de características y clasificación con el modelo de árbol de decisión, se encuentra disponible en el siguiente enlace a GitHub:

Enlace al repositorio de GitHub

# Bibliografía


[1] Sandra Jardim, João António, Carlos Mora 2023 Image thresholding approaches for medical image segmentation - short literature review, pp 1485-92

[2] R. Geetha Ramani, J. Jeslin Shanthamalar 2020 Improved image processing techniques for optic disc segmentation in retinal fundus images, Biomedical Signal Processing and Control

[3] H.N. Veena, A. Muruganandham, T. Senthil Kumaran 2022 A novel optic disc and optic cup segmentation technique to diagnose glaucoma using deep learning convolutional neural network over retinal fundus images

[4] J Sigut, FJ Fumero Batista, R Arnay, J Estévez, T Díaz-Alemán 2023 Interpretable surrogate models to approximate the predictions of convolutional neural networks in glaucoma diagnosis Machine Learning: Science and Technology

[5] FUMERO BATISTA, Francisco José et al. RIM-ONE DL: A Unified Retinal Image Database for Assessing Glaucoma Using Deep Learning. Image Analysis & Stereology, v. 39, n. 3, p. 161-167, nov. 2020. ISSN 1854-5165. Available at: https://www.ias-iss.org/ojs/IAS/article/view/2346.

[6] Ronneberger, O.; Fischer, P.; Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. In Medical Image Computing and Computer-Assisted Intervention—MICCAI 2015; Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F., Eds.; Springer: Cham, Switzerland, 2015; pp. 234–241.

[7] Python. https://www.python.org/

[8] Keras: Deep Learning for humans. https://keras.io/

[9] TensorFlow, Comienza a usar TensorFlow. https://www.tensorflow.org/?hl=es-419

[10] OpenCV. https://pypi.org/project/opencv-python/

[11] Numpy. https://numpy.org/

[12] csv: CSV File Reading and Writing. https://docs.python.org/es/3/library/csv.html

[13] GridSearchCV. GridSearchCV — scikit-learn 1.5.0 documentation https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

[14] CERDA, Jaime y CIFUENTES, Lorena.Uso de curvas ROC en investigación clínica: Aspectos teórico-prácticos. Rev. chil. infectol. [online]. 2012, vol.29, n.2, pp.138-141. ISSN 0716-1018. http://dx.doi.org/10.4067/S0716-10182012000200003.

[15] Daniel Berrar, Performance Measures for Binary Classification, Editor(s): Shoba Ranganathan, Michael Gribskov, Kenta Nakai, Christian Schönbach, Encyclopedia of Bioinformatics and Computational Biology,Academic Press, 2019,Pages 546-560,ISBN 9780128114322, https://doi.org/10.1016/B978-0-12-809633-8.20351-8.

[16] Chase Lipton, Z., Elkan, C., and Narayanaswamy, B., “Thresholding Classifiers to Maximize F1 Score”, 2014. doi:10.48550/arXiv.1402.1892.