

Sistema de Predicción Ocular

Bootcamp BigData, Machine Learning & IA XV
KeepCoding

AUTORES:

- David Sotelo
- Javier Luque
- Leticia Cabañas
- Miguel Ángel Pardo
- Nauzet Fernández
- Sara Cárcamo
- Sofía Gabián

Índice

1. Resumen ejecutivo del MVP

- 1.1 Problema
- 1.2 Solución / Producto
- 1.3 Proceso de elección de la solución a desarrollar
- 1.4 Resultados
- 1.5 Manual de usuario

2. Metodología y Fases del Proyecto

- 2.1 Metodología de trabajo
- 2.2 Procesos y flujos de trabajo
- 2.3 Empleo de competencias adquiridas en el Bootcamp

3. Ideación y Diseño

- 3.1 Exploración y análisis de información sobre la solución a desarrollar
- 3.2 Recopilación: fuentes y material empleado
- 3.3 Análisis y elección de datos

4. Prototipado

- 4.1 Estructura del pipeline
- 4.2 EDA de metadatos
- 4.3 EDA de imágenes
- 4.4 Preprocesado de metadatos
- 4.5 Preprocesado de imágenes

5. MVP Técnico (Modelado y Evaluación)

- 5.1 Entrenamiento y optimización
- 5.2 Mejora de la propuesta de valor del producto (RAG)

6. MVP Desplegado

1. Resumen ejecutivo del MVP

1.1 Problema

El proyecto se enmarca en el área sanitaria, uno de los sectores con mayor impacto directo en la calidad de vida de la población. Este ámbito se caracteriza además por haber adoptado de forma temprana una cultura orientada al dato, donde conviven múltiples formatos de información (tabulares, imágenes, registros clínicos, entre otros).

Partiendo de este contexto, hemos identificado y alineado problemáticas comunes que afectan a distintos agentes del sistema sanitario y de la sociedad en general. El objetivo ha sido diseñar un producto capaz de integrar soluciones transversales, ofreciendo respuestas específicas y adaptadas a las necesidades de cada uno de estos actores.

Actualmente observamos tendencias sistemáticas a nivel global que plantean retos como los siguientes:

Principales (según OMS) orígenes de ceguera y baja visión:

- **DMAE:** un meta-análisis global estima 196 M de personas con DMAE en 2020, proyectadas a 288 M en 2040; ~1,85 M con ceguera por DMAE en 2020. La prevalencia de ceguera por DMAE ≥50 años en 2020 se situó en 0,10 % a nivel global.
- **Retinopatía diabética (RD):** con la diabetes en aumento, la RD es causa líder de pérdida visual en población laboral. Los programas de tele-cribado con IA ya muestran sensibilidad y especificidad > 0,9 para RD “referible”.
- **Catarata:** en 2020, entre mayores de 50 años, la prevalencia edad-estandarizada de ceguera por catarata fue 0,84% y de discapacidad visual moderada-severa (MSVI) 1,01 %; se estimaron 15,2 M de personas ciegas y 78,8 M con MSVI por catarata.
- **Miopía:** proyección clásica a 2050: ~50 % de la población mundial miope (~4,76 G) y ~10 % con alta miopía (~0,94 G), con mayor riesgo de degeneración macular miópica.

Tras revisar los principales orígenes de ceguera y baja visión a nivel global, se observa que patologías como la DMAE, la retinopatía diabética, la catarata y la miopía tienen un impacto creciente en la población mundial, estrechamente relacionado con el envejecimiento y el aumento de la diabetes.

Si trasladamos esta realidad al contexto nacional y regional, los datos reflejan de manera clara cómo España reproduce y amplifica estas tendencias:

- En Galicia, más de una cuarta parte de la población (26,3 %) tiene más de 65 años.
- En Castilla y León, la proporción es del 20,4 %.
- Además, en 2024 se estima que 4,7 millones de adultos en el rango de 20 a 79 años padecen diabetes, una de las principales causas subyacentes de pérdida visual.

A esta situación se suman dificultades operativas propias del sistema sanitario, especialmente marcadas en zonas rurales:

- Escasez de especialistas.
- Alta dispersión geográfica, con estructuras sanitarias muy fragmentadas.
- Ejemplos: en Galicia existen 313 municipios con núcleos poblacionales pequeños y dispersos, mientras que en Castilla y León se contabilizan 3.601 consultorios locales de Atención Primaria frente a sólo 248 centros de salud.

En conjunto, esta combinación de tendencias globales y limitaciones locales refuerza la necesidad de soluciones innovadoras que permitan detectar precozmente las enfermedades oculares, optimizar los recursos disponibles y garantizar el acceso equitativo a la atención sanitaria.

1.2 Proceso de elección de la solución a desarrollar

Tras haber identificado la magnitud del problema, tanto desde una perspectiva global como a nivel local, el siguiente paso consistió en valorar qué tipo de solución podía responder de forma efectiva a estos retos. La prioridad fue analizar si era posible diseñar una herramienta que, además de detectar de manera ágil las principales enfermedades oculares, lograra superar las barreras estructurales de nuestro sistema sanitario, como la dispersión geográfica y la escasez de especialistas en áreas rurales.

Para ello nos apoyamos en los conocimientos y técnicas adquiridas durante el bootcamp, aplicándolos al diseño de un MVP orientado a optimizar el uso de los recursos sanitarios y mejorar la accesibilidad a la atención oftalmológica.

En este contexto, se definieron los siguientes criterios de evaluación para guiar la elección y priorización de la solución propuesta:

Criterios de evaluación empleados:

- Impacto en salud: altísima base de población mayor (Galicia 26,3 %) → más DMAE y catarata; gran bolsa de diabetes en España (4,7 M) → RD frecuente. Objetivo clínico claro: detectar antes y priorizar bien.
- Viabilidad operativa: red muy densa de consultorios locales (CyL) y centros comarcales (Galicia) que pueden capturar retinografías y apoyarse en IA + e-interconsulta (ya recogido en protocolos SERGAS). Esto reduce desplazamientos y tiempos.
- Eficiencia del sistema: la cirugía de catarata ya mueve grandes volúmenes en el SNS; filtrar/ordenar la demanda con IA mejora el uso del quirófano y libera consultas.
- Trazabilidad y gobierno clínico: el seguimiento cuantitativo en retina (p. ej., microaneurismas en RD, drusas/atrofia en DMAE) permite medir resultados y ajustar tratamientos en comarcas envejecidas.

Una vez establecidos estos criterios, resultaba fundamental visualizar cómo se podría materializar la solución en un escenario real de uso. Para ello, planteamos un ejemplo de flujo operativo (user journey) que refleja de manera simplificada los pasos principales que seguiría el sistema desde la captura inicial de la imagen hasta la derivación priorizada al especialista:

1. Captura local (consultorio/centro de salud/unidad móvil).
2. Análisis IA en tiempo casi real con marcado CE previsto para el software sanitario (traje: no referible /referible / urgente).
3. Informe estructurado a la HCE (Ianus/Medora) con justificación explicable.
4. e-Interconsulta automática para casos referibles; listado priorizado para Oftalmología.

1.3 Solución / Producto

A partir del análisis realizado y del diseño del flujo operativo, dimos el paso hacia la construcción de un prototipo funcional. Con los conocimientos adquiridos en el bootcamp desarrollamos un MVP orientado a la detección temprana de las principales enfermedades oculares con mayor impacto sanitario: DMAE, retinopatía diabética (RD), catarata y miopía.

- DMAE
- Retinopatía diabética (RD)
- Catarata
- Miopía

Para evidenciar la capacidad real de la solución, se definieron varios casos de uso principales, que ilustran cómo el MVP aporta valor en distintos niveles del sistema sanitario:

- Soporte a personal sanitario (AP y enfermería): lectura automática de retinografías con informe estructurado y trazabilidad en la HCE, + e-interconsulta al oftalmólogo solo en casos referibles.
- Trazabilidad y seguimiento: cuantificación longitudinal (p. ej., microaneurismas, exudados, drusas) para medir respuesta a tratamiento y riesgo de progresión.
- Cobertura en dispersión geográfica: cámaras portátiles en consultorios locales (CyL) y centros comarcales/rurales (Galicia); la IA hace el pre-triaje local y remite solo lo necesario. La tele-oftalmología en RD está evaluada positivamente en España (efectiva y segura).
- Cribado rápido de casos críticos: priorización automática (semáforo) para DMAE húmeda, EMD y RD referible. (Evidencia de DL con performance clínica alta para decisiones de derivación).
- Reducción de costes: menos derivaciones innecesarias y viajes; descarga de consultas hospitalarias (más cirugía de catarata, menos primeras visitas “banales”). Evaluaciones HTA nacionales ya apoyan telecribado RD por eficiencia.
- Atención a domicilio / telemedicina: para crónicos con movilidad reducida, o integración con dispositivos móviles/OCT portátil cuando aplique (vía programas de hospitalización a domicilio o campañas comarcales).

1.4 Resultados

El desarrollo del MVP ha permitido demostrar la viabilidad de un sistema capaz de **detectar automáticamente enfermedades oculares relevantes** (DMAE, retinopatía diabética, catarata y miopía) a partir de imágenes de fondo de ojo combinadas con metadatos clínicos básicos. Se ha desarrollado un prototipo funcional reproducible y trazable, que integra todo el flujo de trabajo: preparación de datos, entrenamiento del modelo multimodal y despliegue de un pipeline de predicción.

Los resultados obtenidos evidencian que el modelo ofrece un rendimiento consistente y con capacidad de generalización, especialmente en patologías prevalentes como miopía y catarata, y sienta las bases para seguir mejorando la sensibilidad en casos más complejos como DMAE y retinopatía diabética. El valor reside en haber construido un sistema robusto y extensible que puede adaptarse a escenarios clínicos reales.

De forma complementaria, se implementó un módulo de **razonamiento clínico basado en evidencia (RAG)**, que enriquece las predicciones con información procedente de guías oficiales y literatura médica. Este componente amplía la propuesta de valor del MVP, aportando explicabilidad y soporte a la toma de decisiones en contextos de telemedicina y cribado poblacional.

En conjunto, el proyecto demuestra que es posible combinar visión por computador, integración multimodal y técnicas de RAG para generar una herramienta innovadora, con impacto potencial en la accesibilidad a la atención oftalmológica y en la eficiencia del sistema sanitario.

1.5 Manual de usuario

Requisitos mínimos.

SO: Windows, Linux
CPU: Intel Core i5
RAM: 8 GB
Memoria: 256 GB

Datos de entrada.

La aplicación permite introducir dos datos del paciente (edad y sexo) además de la imagen del fondo de retina. Para indicar la edad del paciente debe usar los botones de '-' y '+'. Sólo se puede seleccionar una edad comprendida entre 1 y 120.

A screenshot showing a horizontal slider input for age. The slider has a light gray background with a white rectangular track. In the center of the track is a dark gray rectangular thumb with the number '45' in white. To the right of the thumb are two small square buttons with black outlines: one with a minus sign '-' and one with a plus sign '+'. A red rectangle highlights the area around the plus sign button.

Para seleccionar el género del paciente debe usarse el menú desplegable que aparece a la derecha al pulsar la flecha.

A screenshot showing a dropdown menu for gender selection. The menu is a light gray rectangular box with a thin black border. Inside, the word 'Masculino' is centered in a white rectangular area. To the right of this area is a small dark gray square button with a white downward-pointing arrow. A red rectangle highlights the entire dropdown menu.

La imagen del fondo de retina debe ser arrastrada a la zona indicada.

A screenshot showing a file upload interface. At the top left, the text 'Elige una imagen en formato jpg' is displayed. Below it is a large rectangular input field with a red border. On the left side of this field is a blue cloud icon with an upward arrow. In the center, the text 'Drag and drop file here' is written in bold, and below it, 'Limit 200MB per file • JPG, JPEG' is shown in a smaller font. To the right of the input field is a smaller, rounded rectangular button labeled 'Browse files' in blue text. A red rectangle highlights the entire input field area.

Otra opción para subir la imagen es abrir un navegador de ficheros pulsando el botón de la derecha.

A screenshot showing another file upload interface. At the top left, the text 'Elige una imagen en formato jpg' is displayed. Below it is a large rectangular input field. On the left side is a blue cloud icon with an upward arrow. In the center, the text 'Drag and drop file here' is written in bold, and below it, 'Limit 200MB per file • JPG, JPEG' is shown in a smaller font. To the right of the input field is a smaller, rounded rectangular button labeled 'Browse files' in blue text. A red rectangle highlights the 'Browse files' button.

La imagen debe tener formato JPG o JPEG, con un tamaño máximo de 200MB.

Predicción.

Cuando los datos han sido introducidos en el formulario, se debe usar el botón 'Predecir' para realizar la predicción de enfermedades en base a la información disponible.

Información del Paciente

Edad
45

Género
Masculino

Image

Elige una imagen en formato jpg

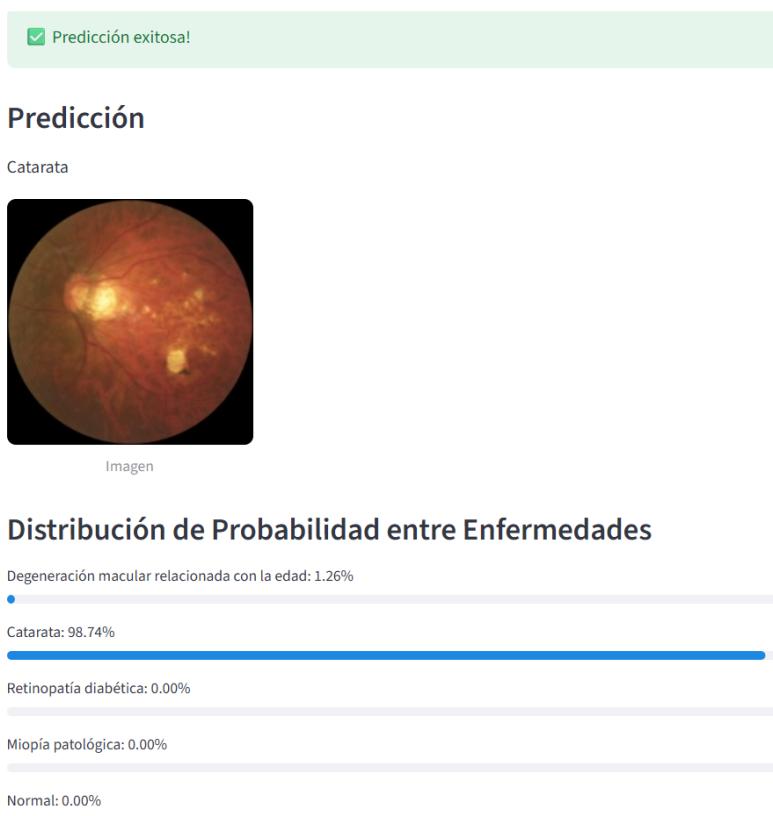
Drag and drop file here
Limit 200MB per file • JPG, JPEG

Browse files

19_right.jpg 18.5KB

Predecir

Si el proceso de predicción se realiza satisfactoriamente aparecerá a continuación la enfermedad más probable que presenta el paciente o si el estado es normal. Se muestra la imagen que se ha utilizado para la predicción y un resumen de las probabilidades calculadas para cada patología y para el resultado normal.



Información clínica adicional.

Para obtener información adicional sobre el resultado debe usar el botón 'Obtener información'.

Información Adicional sobre la condición predicha

 Obtener información

 Exportar Reporte

Aparecerá a continuación información clínica sobre el resultado basada en publicaciones científicas.

Información Adicional sobre la condición predicha

 Obtener información

Información obtenida correctamente.

◆ Diagnóstico diferencial

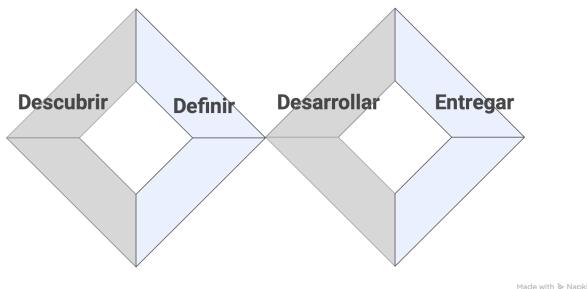
- Nombre: Catarata
 - Probabilidad estimada: Alta
 - Razonamiento: La paciente tiene 55 años y presenta un diagnóstico de catarata, lo cual es común en esta franja etaria.
 - Nivel de evidencia: Moderado
 - Confianza: Alta
- ◆ Pruebas diagnósticas recomendadas
- Prueba: Examen oftalmológico completo
 - Prioridad: Alta
- ◆ Hallazgos esperados: Evaluación de agudeza visual, examen de biomicroscopía, medición de presión intraocular, y exploración del fondo de ojo.
- Contraindicaciones: Ninguna relevante para esta prueba.
- Referencias: Guías de Práctica Clínica de la American Academy of Ophthalmology y NICE.
- ◆ Opciones de tratamiento
- Opción: Cirugía de catarata (facoemulsificación o extracción extracapsular)
 - Indicación: Cuando la catarata afecta significativamente la calidad de vida o la agudeza visual.

Si se necesita toda esta información en formato PDF debe usar el botón de 'Exportar Reporte'

2. Metodología y etapas del proyecto

2.1 Metodología de trabajo

Para garantizar que el desarrollo del proyecto siguiera una ruta ordenada y orientada a resultados, adoptamos como marco de trabajo el **modelo de doble diamante del British Design Council**. Se trata de una metodología ampliamente utilizada en procesos de innovación y diseño, cuyo valor principal radica en combinar fases de exploración amplia con momentos de focalización y concreción.



En nuestro caso, este enfoque no fue solo una guía conceptual, sino también un mecanismo práctico de gestión, ya que cada fase se corresponde con apartados específicos de la memoria, donde se describen con detalle los procesos realizados y los resultados obtenidos.

1) Descubrir

En esta primera etapa se busca entender el problema en toda su amplitud, analizando tanto el contexto clínico como los actores implicados. Aquí se sentaron las bases de la investigación inicial que más tarde permitió acotar el alcance del producto.

- Entender los problemas a resolver.
- Investigación clínica: qué enfermedades predecir.
- Evaluación de stakeholders (necesidades, restricciones, impacto).
- Recolección de datos.

El resultado de esta fase se recoge en el Capítulo 3. Ideación y Diseño, donde se detallan los análisis exploratorios y la definición del enfoque.

2) Definir

El objetivo de esta fase es concretar qué problemas podemos resolver con los datos disponibles y alinear los requisitos técnicos y clínicos con los objetivos del producto. Aquí se delimitan los casos de uso viables y se priorizan las soluciones de mayor impacto.

Ideación y diseño

- Exploración y análisis de información sobre la solución a desarrollar.
- Recopilación: fuentes y material empleado.
- Análisis & Elección (trade-offs y priorización).

Este trabajo también se desarrolla en detalle en el Capítulo 3. Ideación y Diseño (secciones 3.1 a 3.4).

3) Desarrollar

En esta fase se exploran alternativas técnicas y se toman decisiones para avanzar hacia un prototipo viable. El objetivo es transformar los requisitos definidos en artefactos tangibles que preparan el terreno para el modelo final.

Prototipado

- EDA.
- Preprocesado.
- Diseño de arquitectura.

MVP técnico

- Entrenamiento y optimización.
- Mejora de la propuesta de valor del producto (RAG).

Los resultados de esta fase se documentan en el Capítulo 4. Prototipado y en el Capítulo 5. MVP Técnico (Modelado y Evaluación).

4) Entregar

La fase de entrega representa el momento en que el prototipo se convierte en un producto funcional desplegable, accesible y utilizable en un entorno real. Aquí se asegura que el MVP esté disponible para pruebas, validaciones y uso en demo.

MVP Desplegado

- Desarrollo del entorno (FastAPI & Streamlit).
- Pruebas, test y mejoras.

Esta etapa corresponde al Capítulo 6. MVP Desplegado (Producto y Manual).

2.2 Procesos y flujos de trabajo

Principios operativos

Transversalidad y aprendizaje continuo. Todos los miembros participaron en varias áreas (datos, modelado, evaluación y documentación), priorizando la experimentación compartida y la transferencia de conocimiento.

Colaboración abierta. No se establecieron jerarquías funcionales ni roles formales; la coordinación y el reparto se acordaron por consenso en reuniones periódicas.

Trazabilidad. Decisiones técnicas, vías alternativas para mejorar resultados y registros de las acciones tomadas quedaron documentadas en notebooks y resúmenes accesibles.

Modos de trabajo

La elección del modo de trabajo se basó en la importancia y la incertidumbre de la tarea, la posibilidad de descomposición en subproblemas y la necesidad de coordinación; con ello buscamos equilibrar rapidez de decisión, calidad técnica y eficiencia.

Trabajo conjunto. Para tareas de alta importancia, elevada incertidumbre o gran complejidad se trabajó en modo conjunto para acelerar la convergencia, compartir conocimiento y tomar decisiones colectivas.

Células de trabajo (minigrupos). Cuando el problema admitía descomposición, se formaron subgrupos paralelos (p. ej., EDA de metadatos & EDA de imágenes) para avanzar en paralelo y facilitar la integración posterior.

Trabajo individual. Para actividades acotadas, de bajo riesgo y con alcance claro, se asignaron tareas individuales que permitieron ciclos rápidos de experimentación y entrega.

Gestión del proyecto y comunicación

- Repositorio: GitHub como fuente única del código y notebooks; Google Drive para bases documentales (actas, resúmenes y materiales clínicos).
- Comunicación: grupo de WhatsApp y reuniones presenciales/virtuales.
- Actas y decisiones: grabación con OBS (audio/vídeo) y resúmenes automatizados con NotebookLM; los enlaces a los resúmenes se compartían en Drive para acceso de todo el equipo.

Cadencia y tipos de reuniones

Establecimos una cadencia regular de reuniones con objetivos claros para mantener el progreso y la alineación del equipo. Las sesiones se articulaban en tres tipos principales:

- Seguimiento y planificación. Objetivo: priorizar, repartir carga y eliminar bloqueos.

- Toma de decisiones y organización del flujo. Objetivo: consensuar cambios de diseño.
- Sesiones colaborativas (hands-on). Objetivo: construir soluciones o componentes del pipeline, exploración y resolución de problemas.

Flujo de desarrollo extremo a extremo

1. Descubrir y acotar.
2. Exploración de datos (EDA).
3. Diseñar pipeline.
4. Desarrollar, optimizar y validar.
5. Documentar y decidir.

Herramientas

- GitHub: control de versiones y coordinación del código/notebooks.
- Google Drive: repositorio de documentación, actas y resúmenes.
- OBS + NotebookLM: registro de reuniones compartida con los miembros del proyecto.

2.3 Empleo de competencias adquiridas en el Bootcamp

Desde el inicio del proyecto, el equipo tuvo como motivación principal consolidar las competencias adquiridas durante el Bootcamp, procurando que el mayor número posible de ellas se aplicaran en el desarrollo real de un producto. La experimentación estuvo presente en cada etapa, sirviendo como herramienta para contrastar ideas y reforzar el aprendizaje práctico.

Con este propósito, establecimos un compromiso colectivo para definir las áreas de actuación prioritarias, siempre en función de los plazos marcados (deadlines) y de los elementos fundamentales del desarrollo.

Gracias a esta organización, logramos integrar un alto porcentaje de competencias relevantes, asegurando que el proyecto fuese a la vez un ejercicio formativo completo y una solución tecnológica consistente.

1) Ideación y Diseño

- Estudio de calidad de imágenes: tamaños, brillo/contraste, imágenes corruptas.
- Estrategia multimodal: CNN (imágenes) + MLP (metadatos).
- Transfer learning (ResNet18) para acelerar convergencia.

- Incorporación de RAG para mejorar explicabilidad clínica.
- Módulos: Álgebra & Estadística, Deep Learning, NLP, IA.

2) Prototipado

- EDA imágenes: histogramas RGB, detección de imágenes oscuras, contraste, distribución por enfermedad.
- EDA metadatos: análisis de edad, sexo, combinaciones de patologías, tests estadísticos (t-test, chi2).
- Preprocesado (metadatos): binarización de sexo, tokenización y vectorización de keywords diagnósticas.
- Preprocesado (imágenes): resize 224×224, crop cuadrado, normalización con parámetros de ImageNet, data augmentation (flip, rotación).
- Experimentación con arquitecturas diferentes.
- Módulos: Álgebra & Estadística, Machine Learning, Deep Learning.

3) MVP Técnico (Modelado y Evaluación)

- Entrenamiento y optimización.
- Mejora de la propuesta de valor del producto (RAG).
- Módulos: Deep Learning, Despliegue de Algoritmos, NLP, IA.

4) MVP Desplegado

- Desarrollo del entorno (FastAPI & Streamlit).
- Módulos: Despliegue de Algoritmos, IA.

3. Ideación y Diseño

3.1 Exploración y análisis de información sobre la solución a desarrollar

Tras la fase de mapeo (en la que el equipo delimitó el alcance del proyecto y estableció el marco de trabajo) se inició un proceso sistemático de exploración orientado a validar la viabilidad del producto. El eje central de esta etapa fue el análisis de los datos disponibles, ya que solo con fuentes adecuadas en volumen, calidad y relevancia clínica podía garantizarse un desarrollo sólido.

En este contexto, definimos un conjunto de criterios de selección que guiaron la búsqueda y evaluación de datasets, y emprendimos un trabajo inicial de validación para comprobar la calidad, fiabilidad y aplicabilidad de la información recopilada.

Criterios para la recolección de información

- Cobertura clínica: datasets e información que abordasen las dolencias definidas provisionalmente como objetivo.
- Volumen y multimodalidad: cantidad razonable de imágenes y datos tabulares; además, un nivel de complejidad que permitiera validar las capacidades del equipo.
- Fiabilidad del dato: fuentes confiables y de calidad, condición crítica al tratarse de un ámbito sanitario.

Trabajo realizado (validación de calidad y fuentes)

- Verificación de datos reales y calidad/volumen del dataset.
- Localización y análisis de diversas fuentes de contrastada calidad.
- Revisión de los datos en Drive y de las fuentes evaluadas para su posible inclusión.

3.2 Recopilación: fuentes y material empleado

Los datasets identificados se organizaron en función de su fuente de origen y de las dolencias representadas, lo que facilitó un análisis estructurado y comparativo. Esta clasificación permitió evaluar de forma ágil su pertinencia respecto a los requisitos del producto y priorizar aquellas colecciones de datos con mayor potencial de integración en el proyecto.

1) Retinopatía diabética (DR) y edema macular (DME)

- Competiciones de Kaggle: Diabetic Retinopathy Detection (~88 GB, >35.000 imágenes) y APTOS 2019 (~10 GB, ~3.600 imágenes).

- Datasets clínicos especializados: Messidor/Messidor-2 (\approx 3.000 imágenes), DIARETDB1 (130 imágenes con DR), IDRiD (subconjuntos de grading, segmentación y localización con anotaciones detalladas).

Constituyen la mayor fuente de datos con anotaciones estandarizadas para RD/DME.

2) Glaucoma

- Colecciones dedicadas: RIM-ONE DL (485 imágenes), RIGA (750 imágenes con múltiples anotaciones manuales), ORIGA-light (650 imágenes).
- Incluyen localización de estructuras clave (disco óptico, nervio óptico), útiles para segmentación y clasificación.

3) Bases multi-patología

- RFMiD (860 imágenes, 51 enfermedades).
- ODIR-5K (6,7 GB, 3.500 pacientes, 7.000 imágenes, 8 etiquetas: N, D, G, C, A, H, M, O).
- Chākṣu IMAGE (1.345 imágenes, población india, centrado en glaucoma y nervio óptico).

Aportan diversidad más allá de la RD y permiten entrenar modelos más generalistas.

4) Imágenes OCT (Optical Coherence Tomography)

- OCTDL, Eye OCT datasets y OLIVES (~50 GB combinados).
- Incluyen AMD, DME, oclusiones venosas y alteraciones vitreomacular.

Se valoraron pero quedaron descartados, ya que el proyecto se centra en imágenes 2D (fundus).

5) Bases longitudinales / con seguimiento

- RODREP (1.120 imágenes + mosaicos intra/inter-visita).
- OLIVES incluye también seguimiento temporal con biomarcadores.

Útiles para estudios de progresión de enfermedad, aunque no prioritarios para el objetivo inicial.

3.3 Análisis y elección de datos.

A partir de la lista inicial de candidatos, llevamos a cabo un análisis comparativo detallado con el fin de identificar cuáles datasets se ajustaban mejor a los criterios previamente definidos. Este proceso incluyó también la justificación de las exclusiones, señalando de

manera explícita las limitaciones que impedían su aprovechamiento dentro del alcance del proyecto.

A) Foco limitado de patología

- Kaggle DR / APTOS → solo RD (grading 0–4): muy útiles para afinar RD, pero no cubren Catarata/Miopía/AMD.
- Messidor → centrado en RD (excelente para conteo de lesiones), no multi-patología.
- Messidor-2 → imágenes sin etiquetas clínicas detalladas, mantiene enfoque RD-centric.

B) Mismatch de modalidad / dominio

- OCTDL (Mendeley) → OCT, no fundus color; ideal para AMD/DME estructural, pero cambiaría por completo el pipeline (3D/volumétrico).
- OLIVES (Zenodo 7105232) → fundus NIR + OCT + longitudinal y orientado a DR/DME; gran valor clínico, pero no cubre bien Catarata/Miopía/AMD como clases generales y añade complejidad multimodal.

C) Tamaño/estructura poco aprovechables para el use-case

- RFMiD (y 2.0) → multi-enfermedad, pero muy pequeño (≈ 860 imágenes); la escasez por clase limita un clasificador robusto para las 4 dianas.
- UCI Debrecen → tabular derivado de Messidor (sin imágenes); útil para ML clásico, no para un pipeline de visión profunda.

Elección del dataset principal

Tras el análisis comparativo, el equipo concluyó de forma consensuada que el dataset ODIR-5K era el que mejor satisfacía los criterios definidos, al combinar una cobertura clínica adecuada, un volumen representativo y un alto nivel de relevancia científica.

1) Cobertura clínica alineada (en un único dataset)

- Incluye las 4 dianas (AMD, RD, Catarata, Miopía) como clases explícitas.
- Multi-label por paciente → refleja comorbilidad real (p. ej., RD + Catarata) y permite una única cabeza de salida (sigmoides) para todas las patologías.

2) Representatividad y “realismo” de la muestra

- ~5.000 pacientes, ambos ojos, procedentes de múltiples centros/cámaras → variabilidad real de iluminación, óptica y calidad.

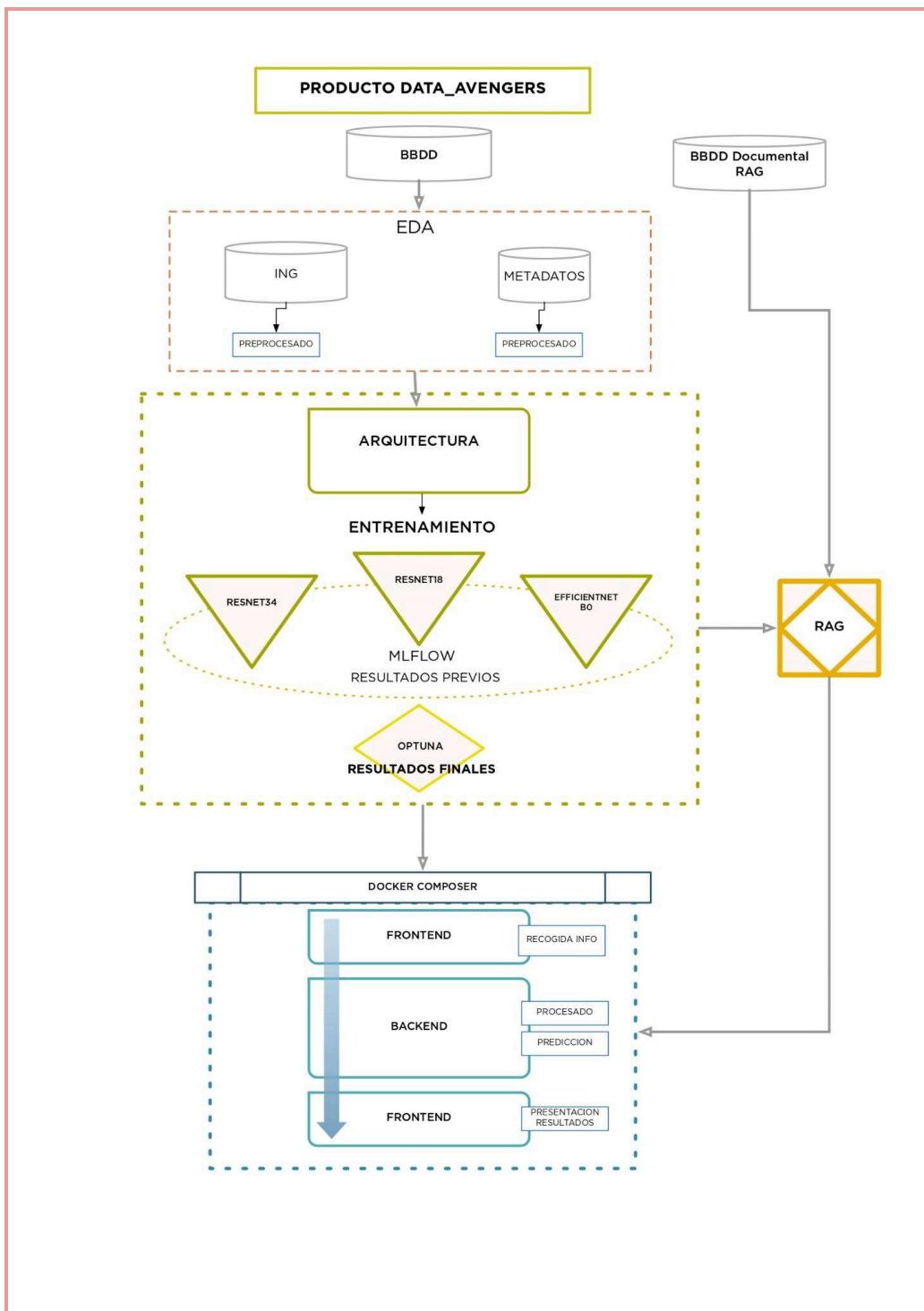
- Desbalance natural de clases (como en clínica) → entrenamiento con class weights/focal loss y calibración de umbrales por patología, acorde a producción.
- Metadatos (edad, sexo, keywords clínicas) y pareado binocular (L/R) → posibilitan explotar asimetrías y consistencia entre ojos.

3) Benchmark ampliamente usado (relevancia científica)

- ODIR-5K se ha consolidado como estándar multi-label en la literatura reciente; existen baselines y comparativas públicas.
- Validar sobre ODIR facilita la reproducibilidad, la defensa de resultados y la transferencia a entornos reales.

4. Prototipado

4.1 Estructura del pipeline



4.2 EDA de Metadatos

Objetivo y encaje en el proyecto

Este EDA sobre metadatos clínicos y administrativos de ODIR-5K comprueba la coherencia del dataset, describe su composición demográfica y la distribución de patologías, e identifica posibles fuentes de sesgo que puedan condicionar el modelado posterior. Se coordinó con el EDA de imágenes para que las decisiones de preprocesado, particionado y evaluación sean consistentes en todo el pipeline.

Librerías.

pandas: para la manipulación de datos tabulares.

numpy: para operaciones numéricas.

matplotlib.pyplot: para visualización básica.

seaborn: para visualizaciones estadísticas más estilizadas.

re: para trabajar con expresiones regulares.

scipy.stats.ttest_ind: para realizar pruebas t de comparación de medias.

scipy.stats.chi2_contingency: para pruebas de independencia entre variables categóricas.

1) Ingesta, sanidad y tipado de variables

Qué se hizo

- Cargamos `full_df.csv` y verificamos dimensiones (6.392 filas, 19 columnas), tipos y ausencia de nulos/NaNs.
- Revisamos duplicidad de identificadores y consistencia relacional entre columnas.
- Examinamos cada columna relevante:
 - a) Etiquetas binarias de clases diagnósticas 'N, D, G, C, A, H, M, O' (0/1).
 - b) Vínculo por ojo: nombre de fichero para 'Left-Fundus' y 'Right-Fundus' asociados al 'ID' de Paciente.

Motivación

- Toda inferencia estadística y cualquier split train/val/test dependen de una tabla limpia y consistente.

Resultados

- Estructura coherente, sin valores nulos y con esquema estable para análisis posteriores.

Decisión

- Mantener ambos ojos como referencias (rutas/nombres)

2) Caracterización demográfica (edad y sexo)

Qué se hizo

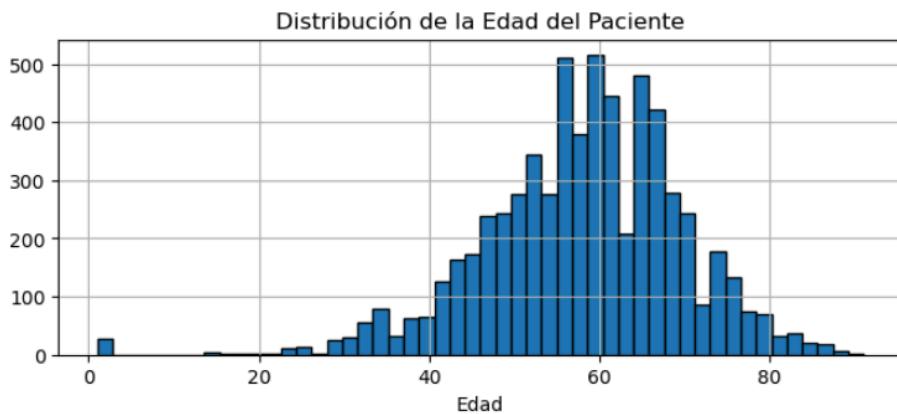
- Análisis de estimadores estadísticos de edad (media, mediana, moda, rango) y su distribución.
- Conteos por sexo y visualización de proporciones.
- Observación de casos extremos (hay 28 pacientes de 1 año en las notas)

Motivación

- La distribución de edad y sexo puede condicionar sesgos del modelo; conviene controlarlas en splits y métricas.

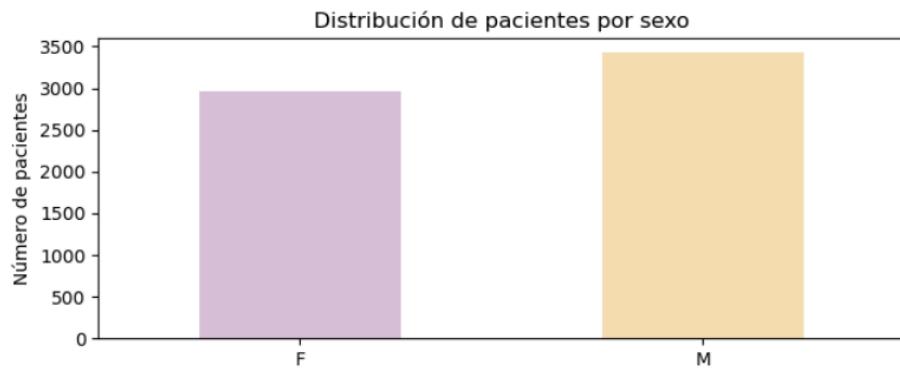
Resultados

- Distribución de edades unimodal ligeramente sesgada a la izquierda.
- Mayoría de pacientes en el rango 50–70 años; minoría pediátrica.



Métrica	Valor (años)
Min	1
Max	91
Media	57.8
Mediana	59
Moda	56

- Distribución de sexo con ligera sobrerepresentación masculina.



Decisión

- No descartar de entrada los extremos; marcarlos como casos límite para auditoría manual y análisis de sensibilidad.
- En el split, estratificar por paciente preservando proporciones por sexo y rangos de edad

3) Análisis de patologías anotadas (N, D, G, C, A, H, M, O)

Qué se hizo

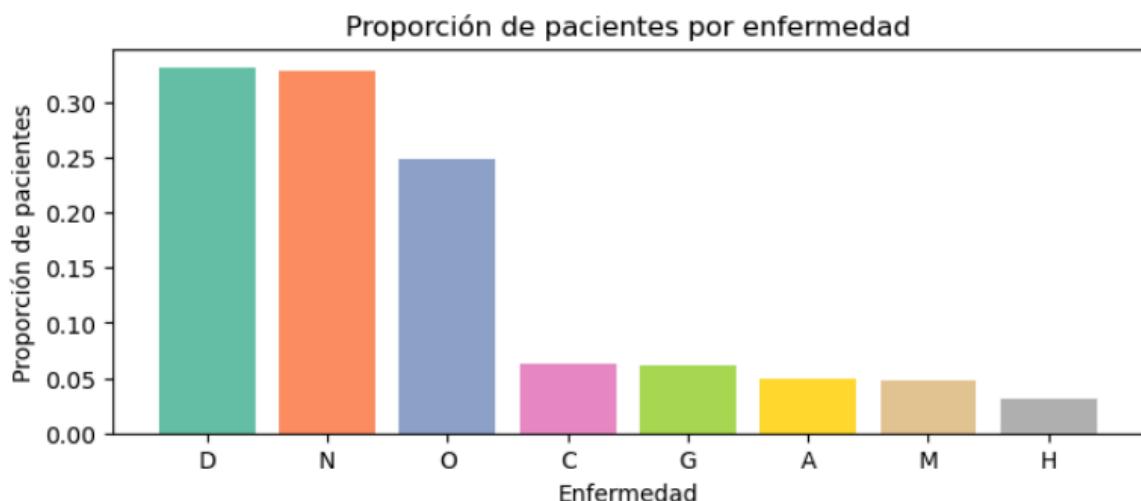
- Recuentos y proporciones por clase a partir de columnas de patologías.
- Revisión de la etiqueta O como agregado mixto de enfermedades.

Motivación

- Conocer la prevalencia por clase para elegir métricas, anticipar descompensación entre clases y definir muestreo/pérdida.

Resultados

- Dominan Normal ($\approx 33\%$) y Diabetes ($\approx 32\%$); resto $< 7\%$ cada una; O $\approx 28\%$.



Decisión

- Manejar el desbalanceo

4) Distribución de sexo por enfermedad.

Qué se hizo

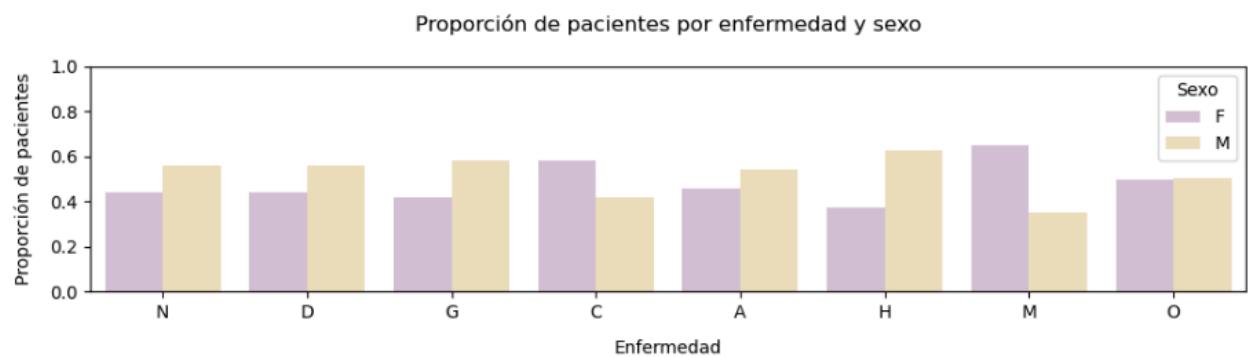
- Para cada enfermedad (sin 'N'), tabla de contingencia 'Patient Sex x presencia(1)/ausencia(0)' y prueba chi2 ('chi2_contingency').

Motivación

- Detectar asociaciones entre sexo y patología para controlar posibles sesgos.

Resultados

- Género masculino tiene mayor representación en la condición normal y en la mayoría de las enfermedades. Hay que tener en cuenta que de partida el número de pacientes de género masculino es mayor.



- Las enfermedades de cataratas (C) y miopía patológica (M), así como la presencia de otras enfermedades (O) muestran una asociación significativa con el sexo del paciente, siendo más común este tipo de enfermedades en el grupo de mujeres que en el de hombres. (p-valores, $\alpha=0.05$)

Enfermedad	p-valor	Significancia
0	1.591738e-02	*
1	8.015363e-02	
2	1.306149e-06	***
3	8.518602e-01	
4	1.108809e-02	*
5	7.533737e-11	***
6	2.993816e-03	**

Decisión

- Valorar el equilibrar sexo por clase en los splits y el batching cuando sea viable.
- Reportar métricas por subgrupos (h/m) y revisar error análisis estratificado.

5) Distribución de edades para cada enfermedad

Qué se hizo

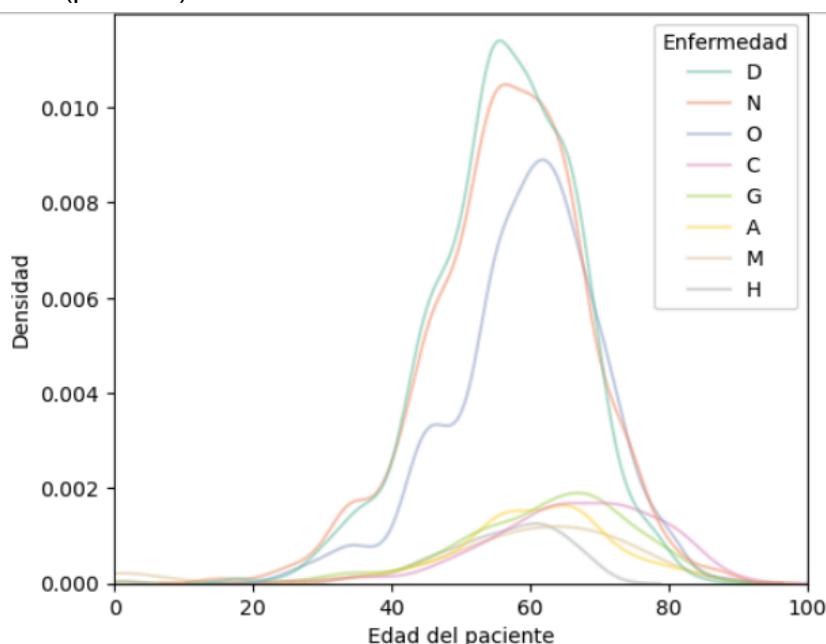
- Para cada enfermedad se compararon las edades de los pacientes con la patología (valor=1) frente a aquellos sin la patología (valor=0).
- Se aplicó la función `ttest_ind` de SciPy con el argumento `equal_var=False`. Se obtuvieron para cada enfermedad el estadístico t, su p-valor y un nivel de significancia (***, **, *, ns).

Motivación

Analizar si existen diferencias relevantes en la distribución de edades de los pacientes según presenten o no una determinada enfermedad. Esto permite localizar patrones de edad asociados a patologías, relevantes para prevenir sesgos en el entrenamiento del modelo.

Resultados

- Diferencias muy significativas ($p < 0.001$) en D, G, C, A y O.
- H también significativa ($p \approx 0.003$).
- M no significativa ($p \approx 0.22$).



Decisión

Incluir la edad como variable de control, asegurando que la distribución de edades se mantenga equilibrada en los splits train/val/test.

6) Análisis de diagnósticos textuales (keywords izquierda/derecha)

Qué se hizo

- Parsing robusto de `Left/Right-Diagnostic Keywords`: lowercasing, estandarización de separadores y strip.
- Búsqueda de incidencias técnicas (`keywords`):
 - `low image quality`
 - `anterior segment image`
 - `image offset`
 - `no fundus image`
 - `lens dust`

Motivación

- Identificar ruido sistemático/artefactos para decidir filtros o preprocesados selectivos en variables categóricas antes del entrenamiento.

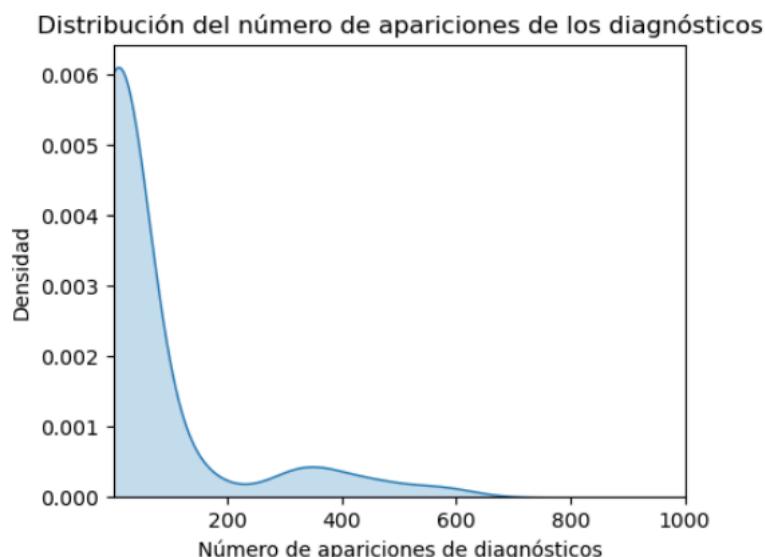
Resultados

- Se encuentran un gran número de diagnósticos distintos.

Número total de diagnósticos distintos: 94	
Top 10 diagnósticos más frecuentes:	

normal fundus	5678
moderate non proliferative retinopathy	1879
mild nonproliferative retinopathy	1063
cataract	594
glaucoma	535
pathological myopia	457
dry age-related macular degeneration	449
hypertensive retinopathy	382
macular epiretinal membrane	376
drusen	338

- Distribución del número de apariciones de cada término con larga cola: ~75.5% aparece <40 veces



- Aparecen diferentes anotaciones de incidencias en las imágenes.

lens dust	289
low image quality	19
anterior segment image	2
image offset	1
no fundus image	1

Decisión

- No usar estas cadenas como labels (ruidosas y dispersas).

7) Estudio de convergencia de patologías por paciente

Qué se hizo

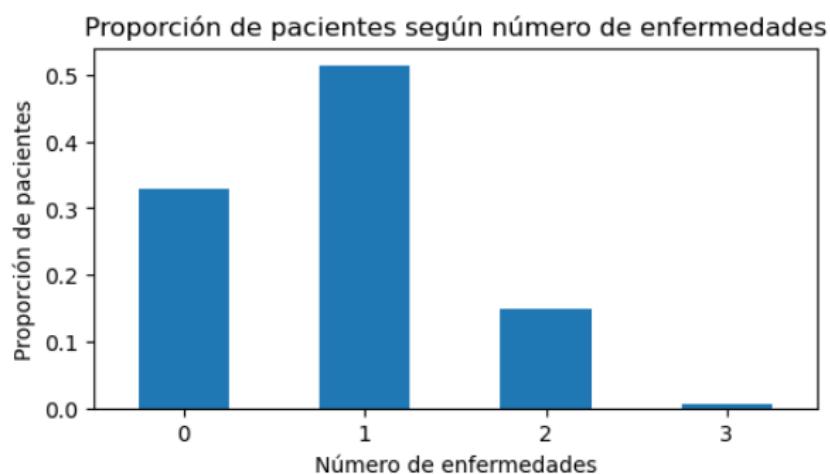
- Conteo por fila del no de patologías ('D...O') y distribución de 0,1,2,3.

Motivación

- Entender convergencia (mismo paciente con distintas clases en particiones diferentes).

Resultados

- 33% sin patología (N=1), 51% con 1, 16% con 2 y <1% con 3; ningún caso >3.



Decisión

- Revisar coherencia ojo-izquierdo/derecho por sujeto para evitar leakage.

4.3 EDA de imágenes

Objetivo

El objetivo de este análisis exploratorio es evaluar la estructura y la calidad visual de las imágenes de fondo de ojo (retinografías) de ambos ojos por paciente, presentes en el dataset **full_df.csv** de **ODIR-5K**, con el fin examinar la estructura, calidad y disponibilidad de las imágenes, así como identificar posibles inconsistencias, variaciones en el tamaño o el formato, diferencias de contraste o brillo, y cualquier otra característica relevante que pueda impactar en la fase de preprocesamiento y entrenamiento del modelo.

La revisión abarca:

- Análisis de dimensiones y formatos.
- Verificación de integridad.
- Inspección de la orientación de las imágenes.
- Cálculo de métricas de calidad visual.
- Evaluación por clase diagnóstica.

Todo ello orientado a definir el preprocesamiento más adecuado.

Metodología

Carga y exploración inicial de las imágenes

Se importa el dataset en formato CSV y se exploran sus dimensiones, así como la estructura, tipos de datos y valores nulos:

- 6.392 registros distribuidos en 19 columnas.
- Estructura coherente y bien organizada, integrando tanto información clínica como referencias cruzadas a las imágenes reales por paciente ('Left-Fundus', 'Right-Fundus', 'filepath', 'filename').
- Todas las columnas están completas, no presentan valores nulos.

Análisis del tamaño y orientación

A partir de la función **get_image_sizes()**:

- Se recorren todas las imágenes y se extraen sus dimensiones (alto, ancho, canales).
- Se identifican los tamaños más frecuentes y se verifica que no hay imágenes corruptas.

Resultados:

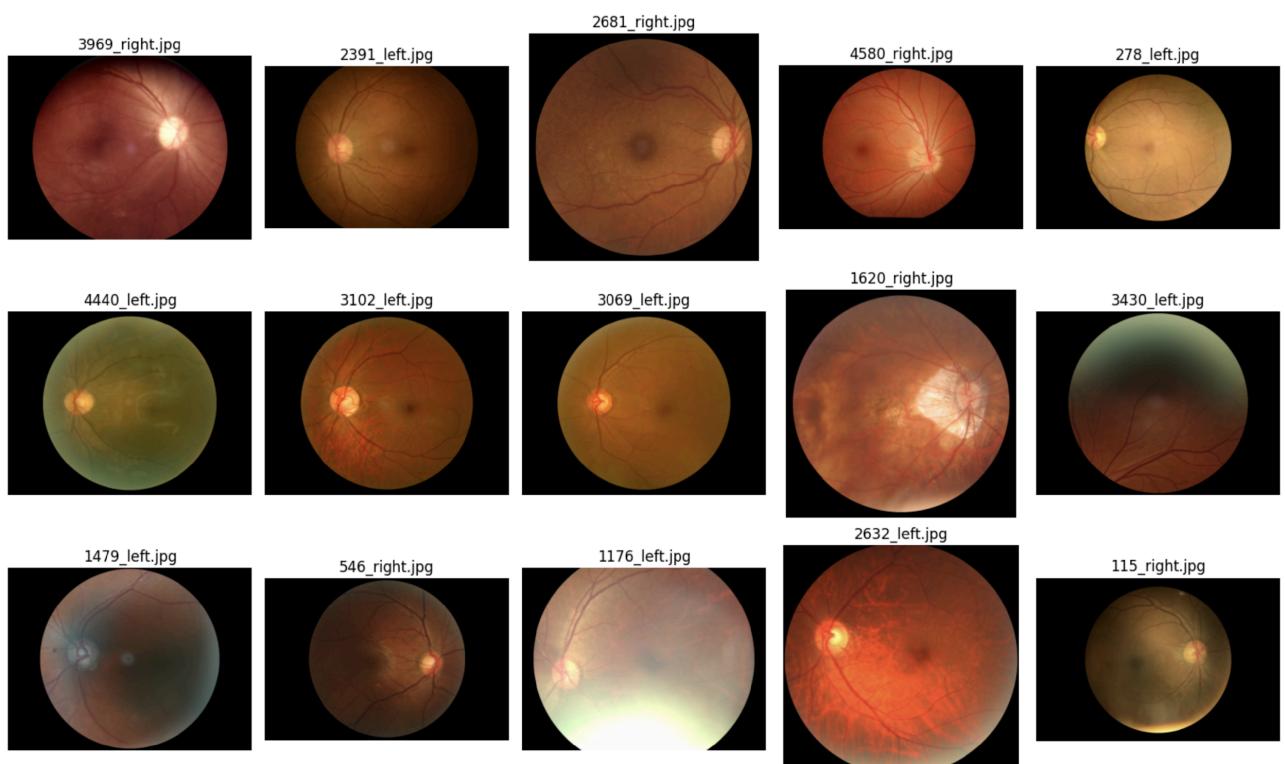
- Gran variabilidad en las dimensiones, más de 15 tamaños diferentes.
- Tamaño más común: 1728×2592×3 con 3.964 imágenes.
- Otros tamaños frecuentes: 1536×2048×3 (978 imágenes) y 1728×2304×3 (784 imágenes).

Implicaciones:

- Necesario aplicar un redimensionado uniforme durante el preprocesamiento: Entrada consistente para el modelo (ResNet, EfficientNet requieren resolución fija).
- Reducción de coste computacional, reduciendo el tamaño de las imágenes más grandes.
- Evitar errores en los DataLoaders al trabajar con batches.

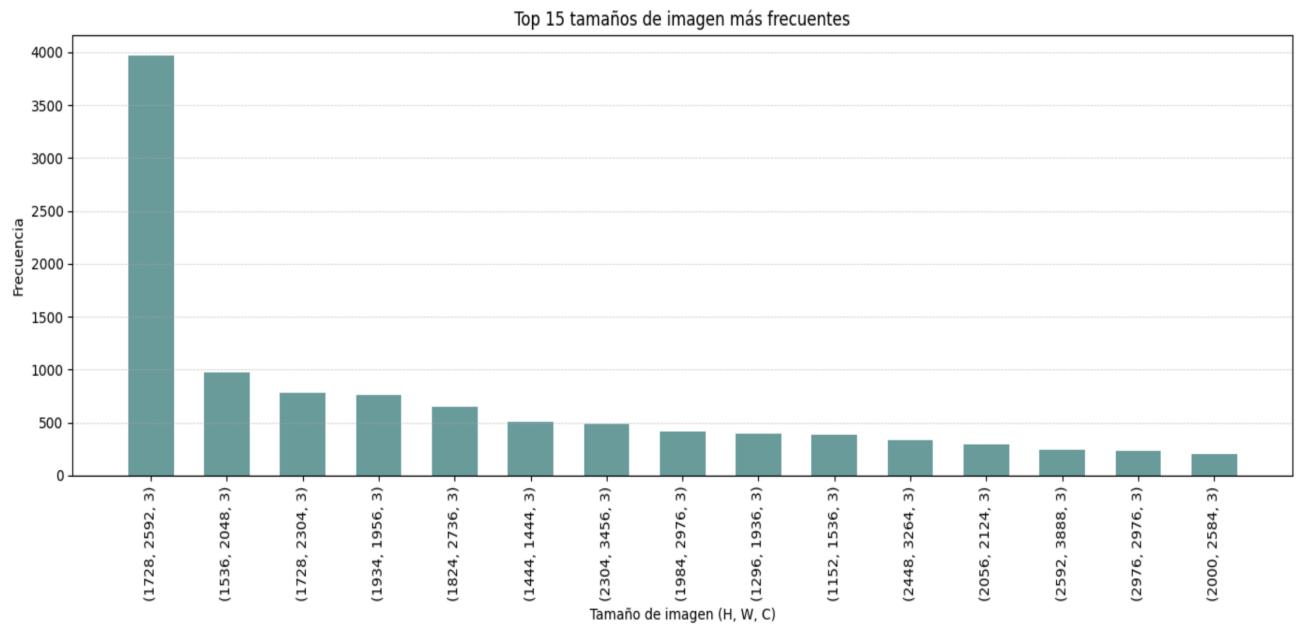
No se identificaron imágenes corruptas → todos los archivos están disponibles para su procesamiento posterior.

Inspección visual aleatoria (**show_random_images()**):



- Se observó variabilidad en orientación y leves rotaciones.
- Recomendación: aplicar **data augmentation** para mejorar la generalización del modelo, o garantizar que todas las imágenes se encuentren alineadas en la misma dirección anatómica.

Distribución de tamaños y clases:



- Gráfico de barras con los 15 tamaños más frecuentes → evidencia heterogeneidad.
- Distribución por clase diagnóstica:

- Clases D y N dominan (25–33%).
- Otras clases en proporciones mucho menores (3–6%).

Se observa que las clases están bastante desbalanceadas.

Se realiza el recuento por clase diagnóstica:

	casos	%
N	2101	32.87
D	2123	33.21
G	397	6.21
C	402	6.29
A	319	4.99
H	203	3.18
M	306	4.79
O	1588	24.84

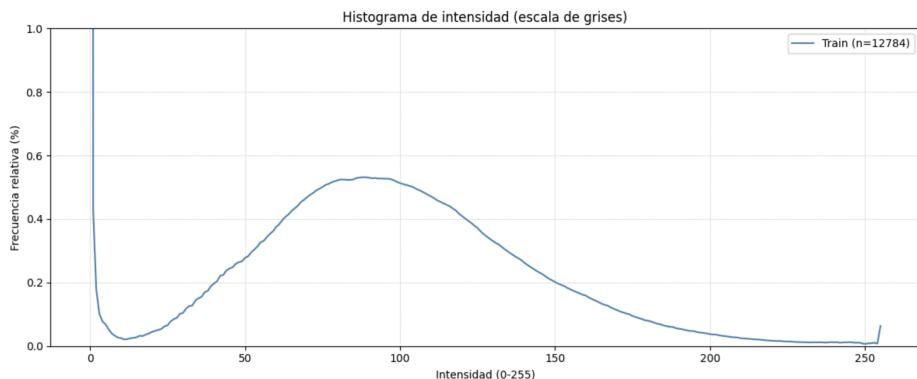
Análisis de color

El análisis busca evaluar variaciones en brillo, contraste y distribución del color, con el objetivo de considerar su impacto en las etapas posteriores de preprocesamiento, especialmente en lo relativo a la normalización y la aplicación de técnicas de data augmentation.

En patologías como retinopatía diabética y edema macular diabético, los canales rojo y verde son especialmente relevantes (microhemorragias, exudados).

Funciones implementadas:

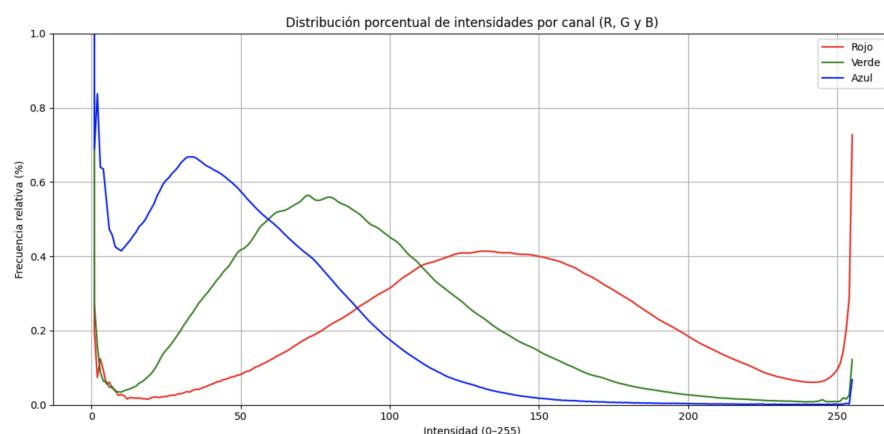
- **compute_grayscale_histogram()** → Calcula histograma promedio normalizado de intensidades en escala de grises.



- Intensidades medias están entre 50 –150, correspondientes a la región central de la retina, donde se concentra la mayor parte de la información clínica relevante.
- Intensidades bajas → áreas periféricas oscuras o fondo negro (sin valor diagnóstico).

Análisis de los canales de color Rojo, Verde y Azul (RGB) para examinar la distribución de intensidades y posibles desequilibrios cromáticos en las imágenes:

- **compute_rgb_histogram()** → Analiza distribución de intensidades RGB.
 - Se calculan los histogramas RGB para las imágenes del dataset. La gráfica no se visualiza correctamente.
 - **normalize_histogram_percent()** → Normalización a porcentaje para mejor visualización.



- Resultado: picos cercanos a 0 → fuerte presencia de fondo negro.

Hallazgos:

- El fondo negro podría inducir al modelo a aprender patrones irrelevantes.

- Canal azul: menor variabilidad e intensidad relativa → aporta menos información diagnóstica.

- Recomendación: aplicar **ColorJitter()** para resaltar estructuras anatómicas relevantes. Esta técnica permite ajustar de forma aleatoria parámetros como el brillo, el contraste y la saturación, lo cual puede facilitar la detección de las diferentes patologías.

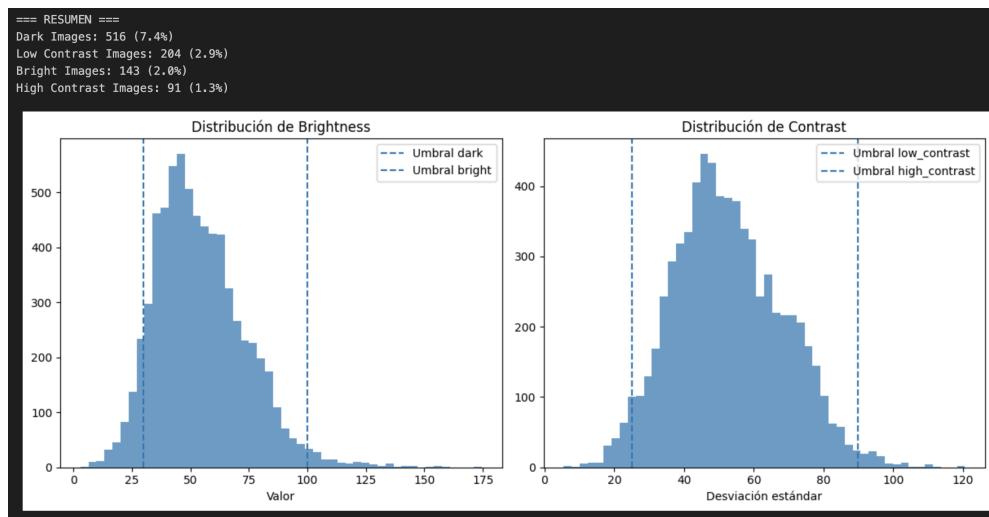
Análisis de calidad

Objetivo: detectar bajo contraste, sobreexposición o iluminación desigual.

Umbrales definidos: 'dark', 'bright', 'low_contrast', 'high_contrast'.

Funciones implementadas:

- **analyze_image()** → calcula brillo y contraste por imagen, marcando flags de calidad.
- **analyze_dataset()** → calcula estadísticas globales para cada categoría diagnóstica.
- **plot_stats()** → histogramas de brillo y contraste indicando los umbrales.



Resultados:

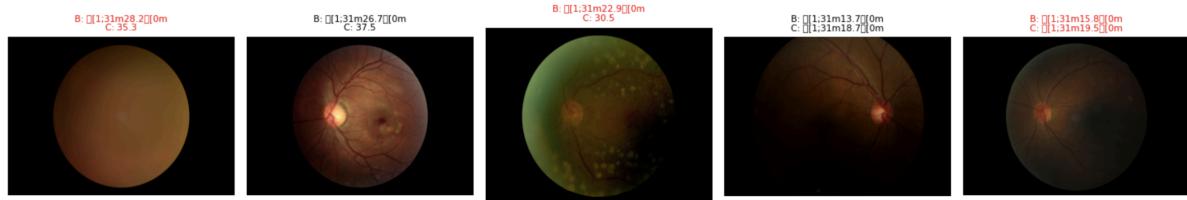
- Imágenes oscuras: 7,4%.
- Bajo contraste: 2,9%.
- Brillantes: 2,0%.
- Alto contraste: 1,3%.

Conclusión: la mayoría de las imágenes tienen una calidad aceptable. Se podría valorar usar **CLAHE** (*Contrast Limited Adaptive Histogram Equalization*) en el preprocesamiento para realzar detalles y mejorar el contraste local sin introducir mucho ruido. Pero se debería aplicar de forma selectiva, no global.

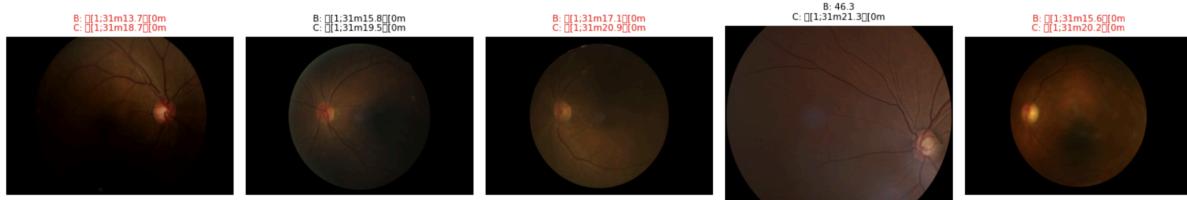
Validación visual:

display_examples() → muestra ejemplos por categoría. Para cada imagen se calcula y anota el brillo y el contraste, lo que permite una inspección cualitativa rápida de los casos marcados por los umbrales, confirmando los resultados de forma cualitativa.

Imágenes oscuras (516 imágenes)



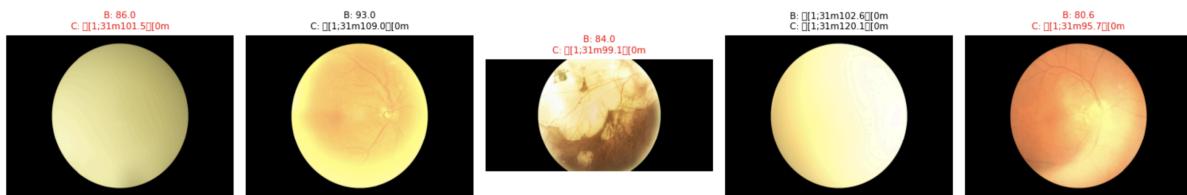
Imágenes con bajo contraste (204 imágenes)



Imágenes demasiado brillantes (143 imágenes)



Imágenes con alto contraste (91 imágenes)



Distribución de imágenes por clase diagnóstica según calidad

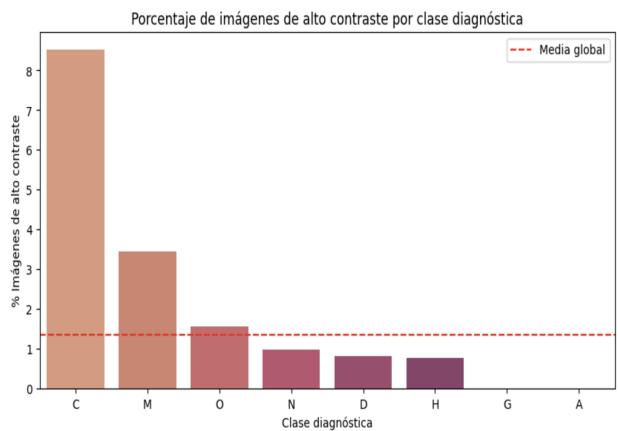
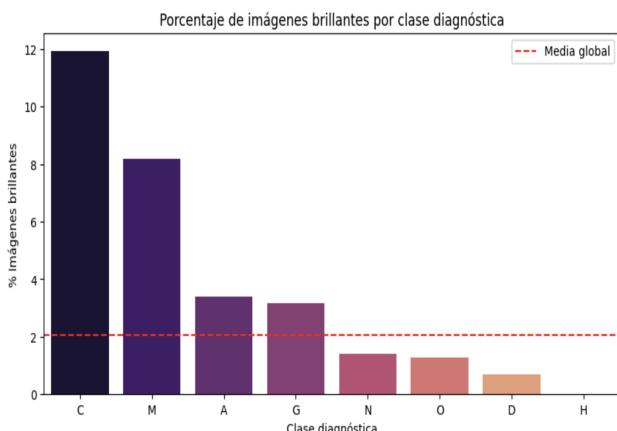
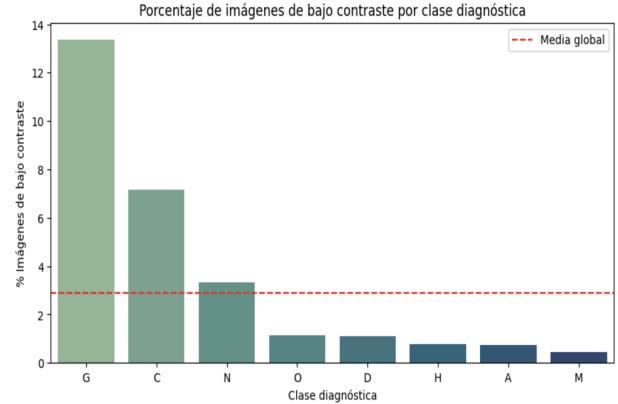
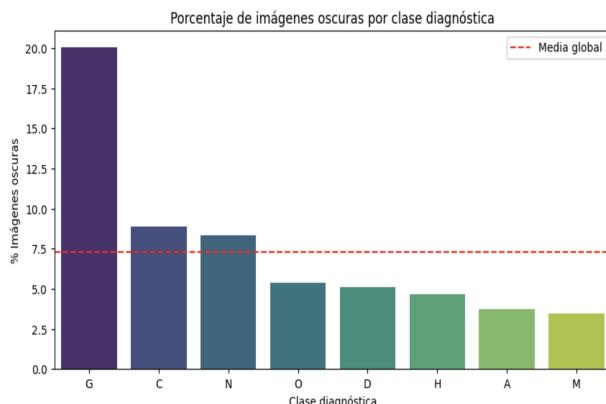
Objetivo: identificar si determinadas patologías están más afectadas por baja calidad, y así justificar la aplicación de técnicas de preprocesamiento específicas para las clases diagnósticas más afectadas.

Procedimiento:

- Creación de un dataframe con 'filename', métricas de brillo/contraste y flags de calidad (is_dark, is_bright, is_low_contrast, is_high_contrast) según los umbrales previamente establecidos.
- Expansión de la columna 'labels' en indicadores binarios, devolviendo **quality_df** con 'ID', 'filename', métricas y etiquetas para análisis por patología.

DataFrame de calidad de imagen: (6392, 17)																
ID	filename	brightness	contrast	is_dark	is_bright	is_low_contrast	is_high_contrast	N	D	G	C	A	H	M	O	labels
0	0_right.jpg	33.274280	30.609474	False	False	False	False	1	0	0	0	0	0	0	0	N
1	1_right.jpg	53.834958	65.333838	False	False	False	False	1	0	0	0	0	0	0	0	N
2	2_right.jpg	34.812584	35.673807	False	False	False	False	0	1	0	0	0	0	0	0	D
3	4_right.jpg	50.609196	64.242010	False	False	False	False	0	1	0	0	0	0	0	0	D
4	5_right.jpg	71.283488	57.147460	False	False	False	False	0	1	0	0	0	0	0	0	D

- Cálculo de % de imágenes problemáticas por clase diagnóstica, crea un dataframe ordenado por porcentaje y se dibuja un gráfico de barras para clase, añadiendo una línea de media global como referencia.



Resultados:

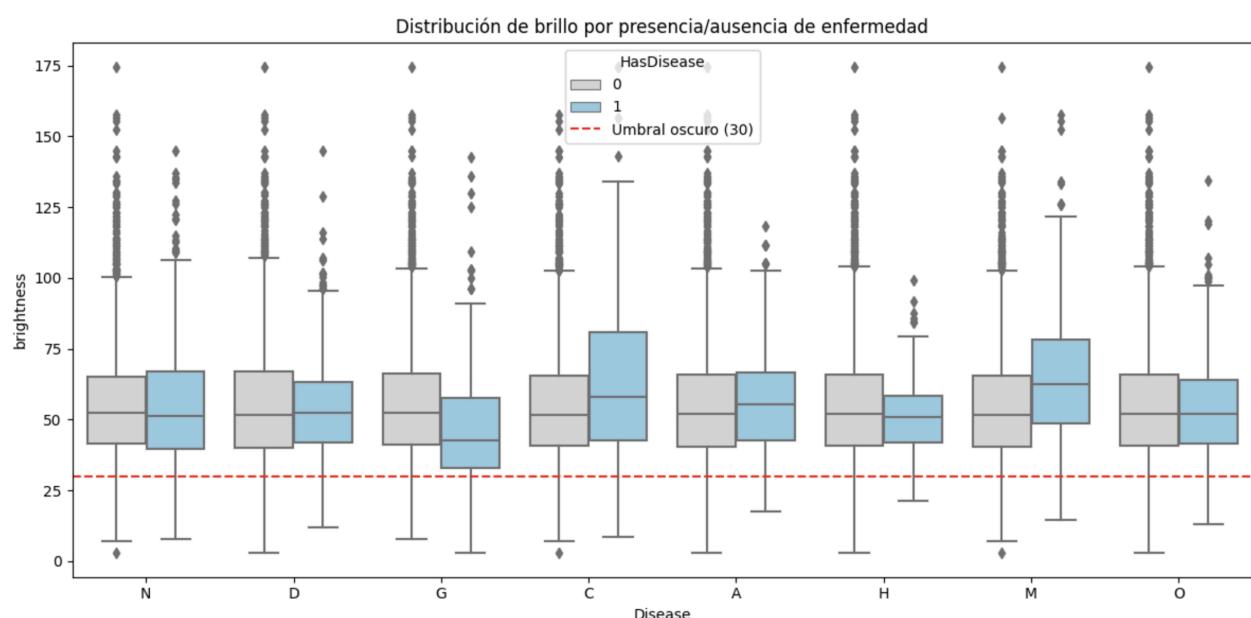
- Glaucoma (G): mayor proporción de imágenes oscuras y bajo contraste.
- Cataratas (C) y Fondo normal (N): ligeramente más oscuras que la media.
- Cataratas (C) y Macular disease (M): valores más altos de imágenes brillantes y alto contraste.

Validaciones adicionales:

- `plot_by_disease_and_category()` → visualizar ejemplos por clase y calidad.



- Boxplot → brillo medio por clase diagnóstica, diferenciando casos positivos (con enfermedad) y negativos (sin enfermedad).



- Test de Welch (no asume varianzas iguales): se compara el brillo medio entre imágenes con y sin cada patología (N–O):

Disease		n_with	n_without	mean_brightness_with	mean_brightness_without	mean_diff	p_value
0	M	232	6160	65.889023	54.066037	11.822986	4.130288e-12
1	G	284	6108	46.797581	54.853066	-8.055484	3.614651e-09
2	C	293	6099	62.780021	54.097146	8.682876	4.744749e-07
3	N	2873	3519	53.850602	55.021386	-1.170784	1.641094e-02
4	H	128	6264	51.585135	54.554620	-2.969485	2.084773e-02
5	A	266	6126	56.599459	54.403784	2.195675	6.107776e-02
6	D	1608	4784	53.821767	54.721496	-0.899729	7.180914e-02
7	O	708	5684	54.301118	54.519326	-0.218208	7.572183e-01

Glaucoma significativamente más oscuro; cataratas/miopía más brillantes.

Conclusiones

Imágenes oscuras (~7,4%):

- No eliminar en bloque (riesgo de sesgo por clase, ej. glaucoma).
- Aplicar ajustes selectivos (**ColorJitter(brightness)**) o ponderar su contribución con **sample_weight/WeightedRandomSampler** en lugar de descartarlas.

Imágenes con bajo/alto contraste:

- Bajo contraste: 2,9%.
- Alto contraste: 1,3%.
 - Valorar usar **CLAHE selectivo** en las marcadas como low_contrast y high_contrast ajustando clip limit para evitar artefactos. No se debería aplicar globalmente.

Tamaño de entrada:

- Variabilidad alta → usar tamaños moderados para evitar coste computacional.
- 224×224 (ResNet18/MobileNet).
- 380×380 (EfficientNet-B4).

Estrategia de redimensionado:

- **Resize + CenterCrop** (mantiene proporciones y región central).
- O bien **resize con padding** para cubrir FOV completo.

Simetría y Data Augmentation:

- Asegurar orientación izquierda/derecha correcta.
- Usar **RandomHorizontalFlip** y rotaciones leves (**RandomRotation**) para mayor robustez.

Aumento cromático:

- Dado el porcentaje bajo de incidencias (2,9% bajo contraste; 2,0% brillantes), aplicar **ColorJitter()** con intensidad moderada y, preferiblemente, de forma condicionada a las imágenes marcadas como problemáticas.

Balanceo de clases:

- Usar **WeightedRandomSampler** o pérdidas ponderadas (**pos_weight**).
- Split estratificado por paciente para preservar proporciones.

Normalización RGB:

- Normalización global por canal (media/std del train).
- Alternativa: valores de **ImageNet** si se usan modelos preentrenados.

4.4 Preprocesado de metadatos

Como ya se comentó en apartados anteriores, en el fichero *full_df.csv* se dispone de información sobre el paciente (edad, sexo, enfermedades oculares) y del fichero que contiene la imagen. El preprocesado incluye la limpieza, transformación y preparación de los datos para su posterior uso en modelos de machine learning. Después del preprocesado se obtendrá información sobre el paciente que se aportará al modelo junto con las imágenes y se extraerá también el *target*. Se considerará un problema de clasificación multi-etiqueta, donde cada muestra será asignada a una posible enfermedad (retinopatía diabética, cataratas, ...) o a un estado normal.

Librerías.

Se utilizan las siguientes librerías para el procesamiento y análisis de datos:

- *pandas*, utilizada para la manipulación de datos en estructuras tipo *DataFrame*.
- *numpy*, proporciona funciones numéricas y estructuras de datos eficientes.
- *polars*, alternativa a pandas, optimizada para velocidad y procesamiento columnar.
- *itertools.chain*, para concatenar iterables de forma eficiente.
- *re*, módulo de expresiones regulares para procesamiento y limpieza de texto.
- *seaborn*, biblioteca para visualización de datos basada en *Matplotlib*.
- *matplotlib.pyplot*, módulo principal de *Matplotlib* para crear gráficos 2D.

Transformaciones

Se aplican las siguientes transformaciones a los datos:

1. Eliminación de columnas redundantes o no informativas.

- *ID*, se elimina esta columna porque es solo un identificador único que no aporta información predictiva.
- *Left-Diagnostic Keywords* y *Right-Diagnostic Keywords*, se considera eliminar estas columnas ya que vamos a crear una columna que indicará la enfermedad por ojo de forma individual.
- Columnas enfermedad (*N*, *D*, *G*, ...), estas columnas corresponden a las clases objetivo (*target*) para la predicción. Se eliminarán ya que tenemos la información en *labels*.
- *filepath* y *target*, después de nuestro planteamiento estas columnas pueden ser eliminadas ya que no tienen más valor.

2. Binarización de variables (sexo del paciente)

La columna *Patient Sex* se transforma a variable binaria para que el modelo pueda interpretarla fácilmente.

3. Codificación de variables categóricas (*Label Encoding*)

Se utilizará la columna *labels* para crear un código del *target* para cada enfermedad, considerando que es buena opción y que no afectará a la hora de entrenar el modelo.

4. Otras columnas.

- *filename*, se mantiene como referencia de que ojo se trata.
- *Patient Age*, se usará como información de entrada al modelo.

5. Filtrado de datos.

Se eliminan todas las muestras de pacientes mayores de un año.

Fichero de salida.

Los datos procesados se guardan en formato *Parquet* con compresión ZSTD, lo que optimiza tanto el espacio de almacenamiento como la eficiencia en futuras operaciones de lectura.

Fichero: *dataset_meta_ojouni.parquet*

4.5 Preprocesado de imágenes.

Las imágenes se transforman a un tamaño cuadrado específico usando el script *transform_img.py* manteniendo la relación de aspecto y la calidad de la imagen.

Este proceso es esencial para preparar imágenes para modelos de *deep learning*, asegurando consistencia en las dimensiones de entrada mientras preserva las características importantes de las imágenes.

Los tamaños empleados para transformar las imágenes son 224x224 y 384x384.

Librerías.

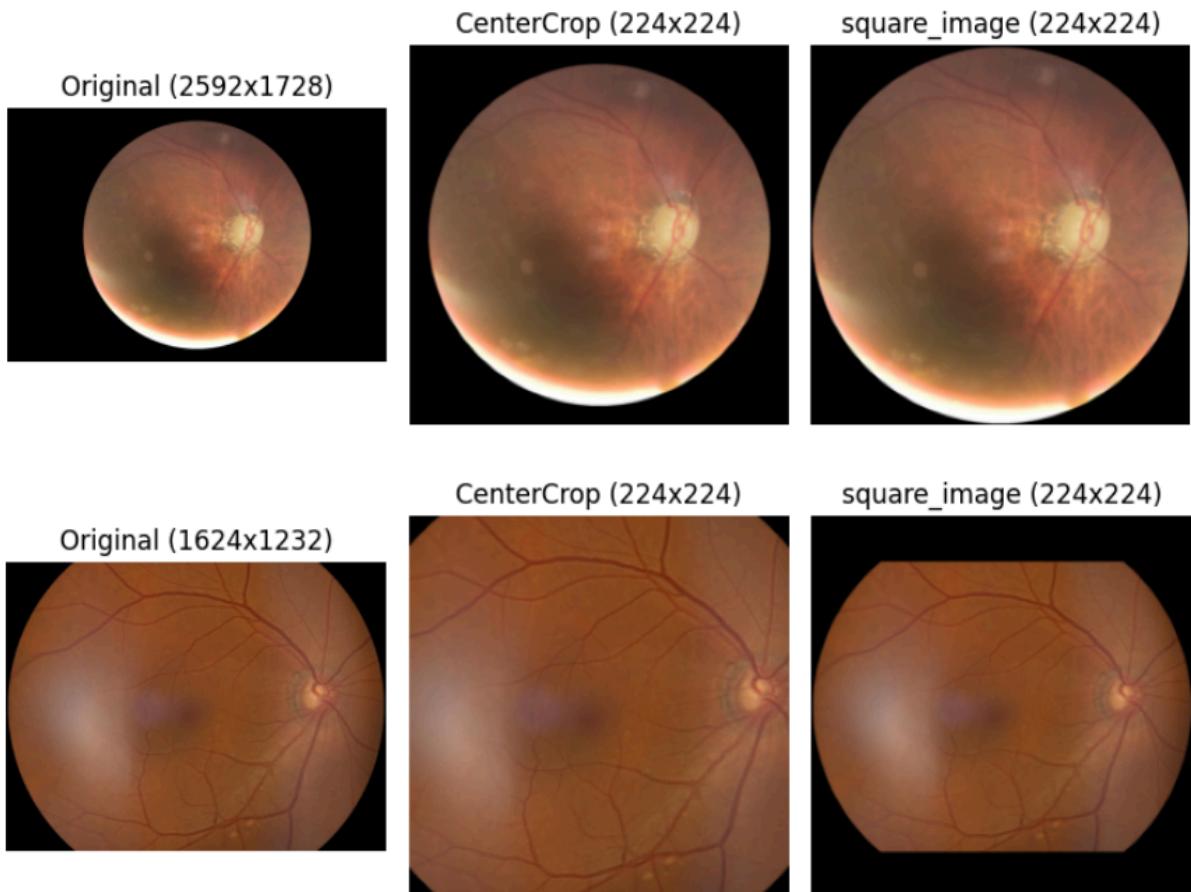
Se utilizan las siguientes librerías para el procesamiento de imágenes.

- *OpenCV (cv2)*, lectura de imágenes
- *PyTorch Vision (transforms)*, transformaciones de variables de imágenes
- *PIL/Pillow*, manipulación y guardado de imágenes
- *tqdm*, barra de progreso en operaciones de procesado

Flujo de procesamiento

1. Conversión a imagen cuadrada:

Mediante la función *square_image()* se trata de mejorar la transformación a formato cuadrado realizada por la función *CenterCrop* de la librería *PyTorch Vision* que en su transformación no hace uso de la característica circular de las imágenes. La función *square_image()* hace uso de esta característica para hacer un recorte más homogéneo de las imágenes como se observa a continuación.



La transformación de las imágenes al formato cuadrado sigue estos pasos:

- Detección y recorte de región de interés.

La función identifica y elimina los bordes negros que comúnmente rodean a las imágenes.

- Expansión a formato cuadrado.

Una vez recortada la región relevante, la función calcula la diferencia entre el alto y el ancho y añade 'padding' negro de manera simétrica al lado más corto. De esta forma se mantiene el contenido centrado en la imagen resultante y se preserva toda la información sin distorsión

2. Redimensionado:

Aplica `transforms.Resize(size)` para escalar la imagen al tamaño solicitado, utilizando interpolación bilineal por defecto de *PyTorch*.

3. Conversión de formato de color:

Convierte de BGR (formato *OpenCV*) a RGB (formato *PIL*) para su correcta interpretación.

5. MVP Técnico (Modelado y Evaluación)

5.1 Entrenamiento y optimización.

Baseline del modelo

El baseline se entrenó con una arquitectura ResNet18 preentrenada en ImageNet como extractor de características para las imágenes de fondo de ojo, combinada con una rama MLP que integra metadatos clínicos (edad y sexo) en un escenario multiclase de una sola etiqueta por ojo.

La configuración inicial incluyó batch size de 32, 10 épocas, optimizador AdamW, tasa de aprendizaje (LR) = 1e-4, weight decay = 1e-4, dropout = 0.5 y la función de pérdida CrossEntropyLoss.

Para este entrenamiento se implementó un pipeline reproducible apoyado en tres scripts principales: ***create_split_dataset.py***, encargado de generar splits estratificados en train/val/test (80/10/10) y guardar los subconjuntos; ***eye_pytorch_dataset.py***, que define la clase *EyeDataset* y las transformaciones de imágenes, integrando también las variables clínicas (edad y sexo) para crear un dataset multimodal listo para PyTorch; y por último, ***load_dataloaders.py***, que construye los DataLoaders de PyTorch a partir de los splits y aplica las transformaciones definidas.

Estos componentes permitieron organizar los datos de manera estructurada, aplicar transformaciones específicas en cada fase y facilitar la integración entre imágenes y metadatos.

Este primer experimento estableció la línea base de rendimiento y sirvió como referencia para posteriores ajustes de hiperparámetros, técnicas de regularización y estrategias de data augmentation.

Los experimentos se registraron y monitorizaron en MLflow, almacenando las métricas de evaluación, parámetros y artefactos (reportes y matrices de confusión) por cada entrenamiento.

Name	Time created	Last modified	Description	Tags
ExperimentoNac_singlelabel_top5_best	09/03/2025, 08:20:00 AM	09/03/2025, 08:20:00 AM	-	
RAO_Clinico	08/31/2025, 05:49:42 PM	08/31/2025, 05:49:42 PM	-	
Experimento_4_Leticia_ResNet18_Top5	08/25/2025, 10:39:11 PM	08/25/2025, 10:39:11 PM	-	
ExperimentoNac_singlelabel_top5	08/25/2025, 07:27:35 PM	08/25/2025, 07:27:35 PM	-	
ResNet50_MPS	08/25/2025, 09:17:45 AM	08/25/2025, 09:17:45 AM	-	
Experimento_Leticia_ResNet18	08/24/2025, 07:39:25 PM	08/24/2025, 07:38:25 PM	-	
Experimento_Leticia_ResNet18	08/23/2025, 08:44:45 PM	08/23/2025, 08:44:45 PM	-	
DataAvengerDavid	08/22/2025, 09:13:15 AM	08/22/2025, 09:13:15 AM	-	
Experimento_David_EfficientNet_MPS	08/22/2025, 09:10:11 AM	08/22/2025, 09:10:11 AM	-	
DataAvengerSofia	08/20/2025, 08:24:39 PM	08/20/2025, 08:24:39 PM	-	
ExperimentoNac_singlelabel	08/20/2025, 10:11:11 AM	08/20/2025, 10:11:11 AM	-	
Default	08/20/2025, 10:11:11 AM	08/20/2025, 10:11:11 AM	-	

Vista registro experimentos en mlflow.

Ejemplo vista Overview Experimento.

Metric	Value
test_top5	0.9230769230769231
val_loss	1.6576153347327274
train_loss	0.2504396044980845
test_loss	1.5475431811678542
val_top5	0.9025157232704403
lr	1.0001228143732988e-7
val_acc	0.6682389937106918
test_acc	0.6828885400313972

Parameter	Value
ema_decay	0.9995
label_smoothing	0.05
onecycle_pct_start	0.2
scheduler	OneCycleLR
num_classes	8
weight_decay	0.0005
num_workers	4
head_lr_mult	8.0
epochs	35
batch_size	32
meta_dim	3
lr	0.0001
backbone	resnet34_imagenet
balanced_sampler	True

Vista métricas y parámetros Experimento.

Entrenamientos exploratorios iniciales

El objetivo de estos entrenamientos iniciales fue verificar y afinar el pipeline multimodal (imagen + metadatos), comparar backbones preentrenados y ajustar LR/regularización/scheduler priorizando estabilidad y tiempo de cómputo antes de la optimización fina.

- **ExperimentoNau_singlelabel.** Partiendo de ResNet18 y migrando a **ResNet34**, se incorporan progresivamente: *warmup + cosine*, **LR discriminativo** (cabeza/meta 5–8× backbone), **WeightedRandomSampler**, **label smoothing = 0.05**, **OneCycleLR** por *batch*, **EMA** de pesos, pruebas con **Mixup**, **AMP**, **early stopping** y ajustes de **weight decay**.
- **DataAvengersSofía.** Comparativa de arquitecturas con configuración controlada: **ResNet18/50**, **EfficientNet-B0/B3**, **ConvNeXt-Tiny**, **ViT-B/16**. Se emplean **label smoothing = 0.05** y **early stopping**; se compara todas las clases vs. subset 5 clases [0,1,2,5,6]; se registran tiempos de entrenamiento para ponderar capacidad/coste.

- **Experimento_David_CustomNet_MPS.** *Smoke test* del pipeline en Apple Silicon (**MPS**) con red personalizada (sin *transfer learning*). Configuración simple (**AdamW**, **CosineAnnealingLR**, **15 épocas**) para validar flujo, dimensiones y compatibilidad. Métrica principal: accuracy.
- **Experimento_Leticia_ResNet18.** **class weights + label smoothing (0.05), gradient clipping, sin scheduler, early stopping por F1-macro.** Variaciones de batch/épocas/weight decay para estabilizar F1-macro.

Además, se incluyó un **clasificador solo-imagen** (ResNet18) para aislar la contribución de la rama de metadatos: mantiene **class weights** según distribución de entrenamiento, **label smoothing = 0.05**, **clipping (max_norm = 5.0)**, **AdamW (LR = 1e-4, WD = 5e-4)** y **early stopping** por F1-macro. Este run permitió cuantificar la ganancia de incluir metadatos.

Hallazgos iniciales:

- Reducir a 5 clases (0,1,2,5,6) mejora estabilidad y métricas.
- Backbones más sólidos: ResNet18/34 y EfficientNet-B0; el resto no compensa el coste.
- Metadatos aportan mejora real (multimodal > solo imagen).
- Clase 2 (Retinopatía diabética) se observa que es el cuello de botella.
- Configuración que funciona bien: label smoothing ≈ 0.05 , EMA, OneCycleLR (con LR fijo base $\approx 1e-4$).

Experimentos finales

Los experimentos finales se centraron en el subset de cinco clases [0,1,2,5,6]: DMAE, catarata, retinopatía diabética, miopía y normal.

Data augmentation (para mejorar generalización y mitigar *overfitting*):

- RandomAffine ($\pm 7^\circ$, $\pm 2\%$ *translate*, *scale* 0.95–1.05).
- ColorJitter ($\pm 10\%$ brillo/contraste/saturación, *hue* 0.02).
- RandomHorizontalFlip (0.5).
- RandomErasing ($p = 0.25$, *scale* 0.01–0.03, *ratio* 0.3–3.3, *value* aleatorio).

ExperimentoNauv_singlelabel_top5

- **train_Nauv8_5class.** ResNet34 + metadatos; **sampler balanceado**, **label smoothing 0.05**, **OneCycleLR** con **LR discriminativo** (head/meta = 8× backbone), **AMP** y **EMA**; **early stopping (patience = 10)** por **val_acc**. (*Nota: con 5 clases el Top-5 ≈ 100% y no es informativo; se registra por consistencia.*)
- **train_nauv9_384.** Igual planteamiento con **384x384** (batch 16), **OneCycleLR**, **EMA**, **AMP**, **smoothing 0.05** y **early stopping** (val_acc). Evalúa si la mayor resolución aporta mejora estable.

- **train_nau_v10_384.** Añade *warmup* con backbone **congelado (3 épocas)** y posterior **unfreeze + OneCycleLR**, manteniendo **sampler, EMA/AMP, smoothing 0.05** y **early stopping** (*val_acc*).

Experimento_4_Leticia_ResNet18_Top5

- **train_4_Leticia (224 px).** ResNet18 + MLP metadatos; **class weights** (sin sampler), **label smoothing 0.05**, **sin scheduler, gradient clipping, early stopping por F1-macro (patience = 6)**.
- **train_5_Leticia (224 px).** Igual base, añadiendo **ReduceLROnPlateau** (monitorizando **val F1-macro**), **class weights, label smoothing 0.05, AMP** y **early stopping** por F1-macro (patience = 6).

Run	Modelo	Img (px)	Acc_val	F1-macro val	Acc_test	F1-macro test
train_Nauv8_5class	ResNet34 + metadatos	224	0.802	0.767	0.823	0.817
train_nauv9_384	ResNet34 + metadatos	384	0.773	0.765	0.811	0.802
train_nau_v10_384	ResNet34 + metadatos (warmup+unfreeze)	384	0.782	0.781	0.790	0.797
train_4_Leticia	ResNet18 + metadatos	224	0.716	0.718	0.724	0.750
train_5_Leticia	ResNet18 + metadatos (ReduceLROnPlateau + AMP)	224	0.702	0.738	0.720	0.775

Comparativa métricas globales

Tras evaluar los resultados se observó que en **ResNet34**, el mejor resultado se obtuvo a **224 px**: **train_Nauv8_5class** alcanzando un **0.802** de accuracy en validación y **0.823** en test, con F1-macro **0.767** y **0.817**, respectivamente. Elevar la resolución a **384 px** no aportó mejora consistente.

En **ResNet18**, **train_5_Leticia** mejoró el F1-macro frente a **train_4_Leticia (0.775 vs 0.750)** gracias a *ReduceLROnPlateau + AMP*, si bien quedó por debajo de ResNet34 en accuracy.

Incidencia y trazabilidad de datos. Tras obtener el mejor resultado inicial con el experimento **ExperimentoNau_singlelabel_top5 (train_Nauv8_5class)** y optimizarlo con Optuna, se detectó que los datasets empleados no eran los correctos. Se reconstruyeron y validaron. Y antes de reoptimizar se ejecutó de nuevo con los datos corregidos.

Al reentrenar **train_Nauv8_5class** con los datasets corregidos, el rendimiento disminuyó respecto a la ejecución previa: F1-macro de validación **0.764 → 0.705** y accuracy **0.817 → 0.710**; en test, F1-macro **0.828 → 0.770** y accuracy **0.850 → 0.743**. A la vista de estos resultados, se evaluó una configuración alternativa (**train_Sofia4_4**) como siguiente paso previo a una nueva fase de optimización.

ExperimentoNau_singlelabel_top5_best (train_Sofia4_4):

Objetivo: reevaluar el pipeline con datos corregidos usando una configuración eficiente y estable para el subconjunto de 5 clases [0,1,2,5,6].

Backbone imagen	EfficientNet-B0 (ImageNet)
Clases	5
Batch size	32
Épocas	15
Learning rate	1e-4 (constante)
Weight decay	5e-4
Optimizer	AdamW
Label smoothing	0.05
Balanceo	Class weights en CrossEntropyLoss
Early stopping	F1-macro (val), patience = 4
Augmentación	Afine $\pm 7^\circ$, ColorJitter moderado, HorizontalFlip (0,5) y RandomErasing

Configuración *ExperimentoNau_singlelabel_top5_best (train_Sofia4_4)*

Resultados:

Métrica	Valor
Acc_train	0.825
Loss_train	0.714
F1-macro train	0.855
Acc_val	0.740
Loss_val	0.990
F1-macro val	0.726
Acc_test	0.735
Loss_test	0.654
F1-macro test	0.735
Precision-macro test	0.706
Recall-macro test	0.789

Métricas globales

Clase	F1-score
0 – DMAE	0.645
1 – Catarata	0.812
2 – Diabetes	0.577
5 – Miopía	0.844
6 – Normal	0.797

F1 por clase_test

Conclusiones y decisión:

El modelo **EfficientNet-B0 + metadatos (train_Sofia4_4)** ofrece un buen equilibrio entre precisión global y F1-macro, con un comportamiento estable, aunque se observa algo de sobreajuste. Dado su F1-macro de validación/test en el rango **~0.73** y su robustez operacional, se selecciona como configuración base para la optimización con Optuna.

Optimización de hiperparámetros (Optuna)

Obtención de hiperparámetros óptimos (optuna_effi.py):

- **Ámbito:** **EfficientNet-B0** (preentrenada) + rama **MLP** de metadatos, subconjunto 5 clases [0,1,2,5,6], data augmentation moderada (Affine, ColorJitter, Flip, RandomErasing).
- **Criterio objetivo:** **maximizar** val_acc (se guarda también test_acc como atributo del trial).
- **Búsqueda** (TPE por defecto) + **pruning**: MedianPruner(n_warmup_steps=3).
- **Early stopping interno:** patience = 4 sobre validación dentro de cada trial.
- **Balanceo:** **WeightedRandomSampler** sobre el conjunto de entrenamiento (sin class weights en la pérdida).
- **Scheduler:** **OneCycleLR** (con pct_start tunable); **EMA** de pesos durante entrenamiento; AMP.
- **Espacio de búsqueda** (rangos principales):
 - lr [5e-5, 5e-4] (log), weight_decay [1e-5, 1e-3] (log)
 - dropout [0.30, 0.70], label_smoothing [0.00, 0.10]
 - epochs [12, 18], batch_size {16, 32}
 - onecycle_pct_start [0.05, 0.30], ema_decay [0.995, 0.9999]

Parámetro	Valor
learning_rate	0.0003385
weight_decay	0.0000647
dropout	0.5936
label_smoothing	0.0960
batch_size	16
epochs	12
onecycle_pct_start	0.2909
ema_decay	0.9957

best_params (Optuna)

Estos hiperparámetros se aplicaron después en los reentrenamientos “retoque” (v1-v4), que consolidan el rendimiento añadiendo técnicas de estabilización (**EMA**, **SWA**, **TTA/MC-Dropout**, y en v3 **LRs discriminativos**), y comparan variantes del *pipeline* bajo la misma configuración óptima de base.

Runs “retoque” con mejores hiperparámetros (train_Sofia_4_4_optuna_retoque):

Se reentrenó la configuración incorporando progresivamente técnicas de estabilización y evaluación robusta:

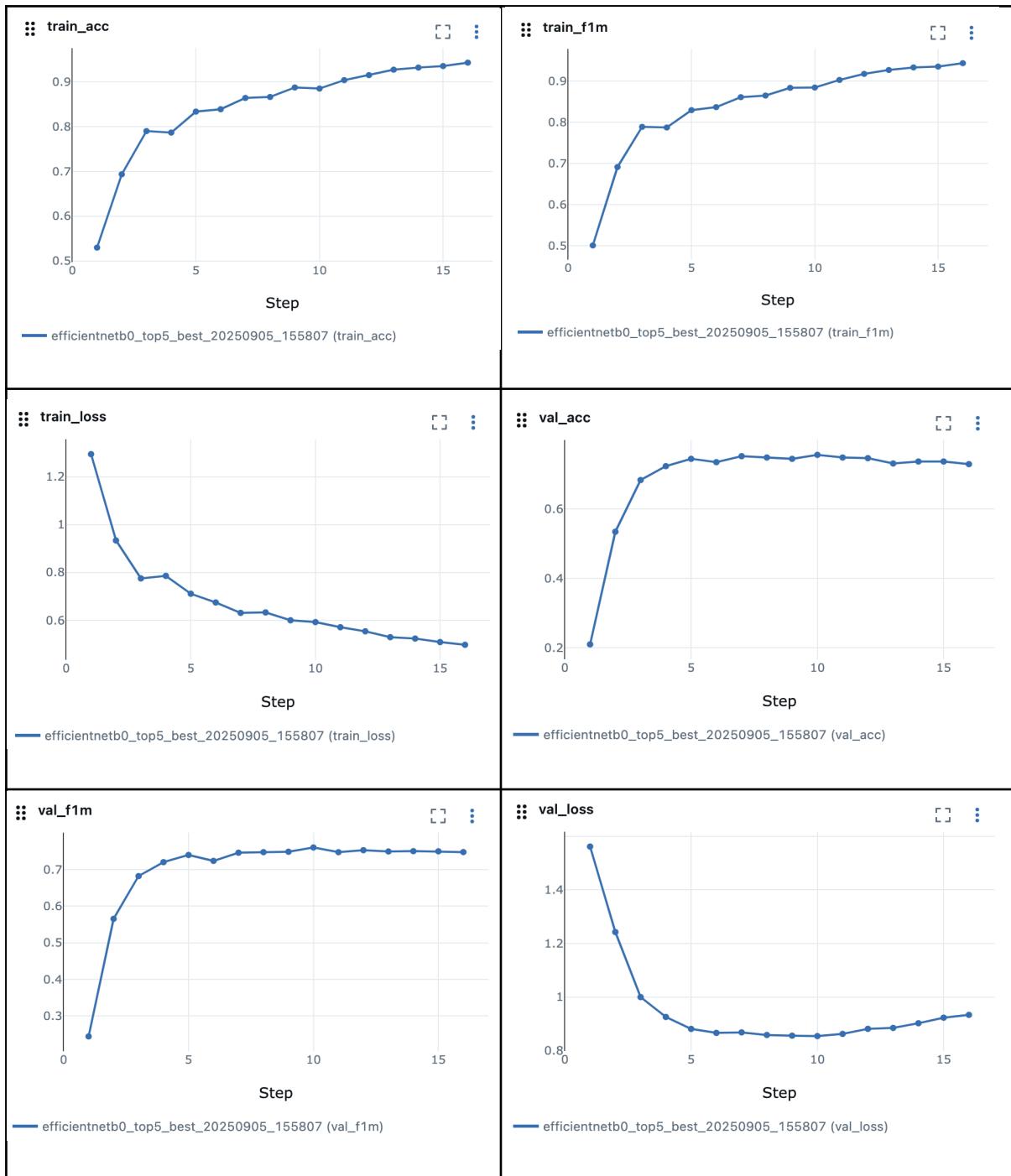
- **Retoque (v1):** OneCycleLR, EMA, AMP, TTA (flip) en validación/test y WeightedRandomSampler. Early stopping con paciencia 6.
- **Retoque (v2):** añade MC-Dropout (varias pasadas) combinado con TTA para promediar predicciones y estabilizar F1/accuracy. Mantiene EMA, OneCycleLR, sampler y paciencia 6.
- **Retoque (v3):** incorpora LRs discriminativos en OneCycleLR (backbone vs. cabeza/meta), SWA con recalibración de BatchNorm, y TTA + MC-Dropout en validación/test; mantiene EMA y WeightedRandomSampler (paciencia 6).
- **Retoque (v4):** mantiene SWA y EMA con selección automática del mejor para el test final; elimina el sampler (usa shuffle), conserva TTA y amplía early stopping (paciencia 8).

Run	Acc_val	F1-macro_val	Acc_test	F1-macro_test	Precision-macro test	Recall-macro test
Retoque v1	0.761	0.763	0.762	0.762	0.748	0.784
Retoque v2	0.729	0.748	0.777	0.788	0.804	0.777
Retoque v3	0.739	0.743	0.766	0.772	0.790	0.764
Retoque v4	0.763	0.749	0.773	0.777	0.856	0.744

Métricas Runs “retoque”

Los mejores resultados se obtuvieron con el modelo **train_Sofia_4_4_optuna_retoquev2**.

A continuación se muestran las gráficas de las métricas registradas, los resultados por clase, así como las matrices de confusión (val y test):



Gráficas de las métricas registradas en milflow

Clase	Precisión	Recall	F1-score	Soporte
0 – DMAE	0.818	0.692	0.750	26
1 – Catarata	0.852	0.793	0.821	29
2 – Retinopatía diabética	0.662	0.571	0.613	161
3 – Miopía	0.800	0.762	0.780	21
4 – Normal	0.737	0.812	0.773	287

val_classification_report (val)

Agregado	Precisión	Recall	F1-score	Soporte
Accuracy (global)	0.729	--	--	524
Macro avg	0.774	0.726	0.748	524
Weighted avg	0.727	0.729	0.726	524

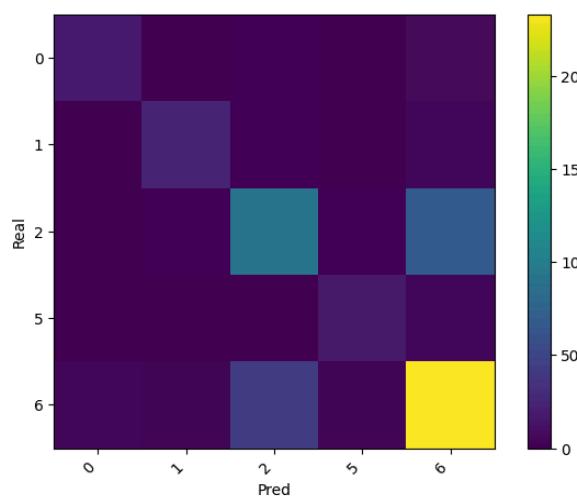
val_classification_report (val)

Clase	Precisión	Recall	F1-score	Soporte
0 – DMAE	0.652	0.556	0.600	27
1 – Catarata	0.931	0.900	0.915	30
2 – Retinopatía diabética	0.763	0.621	0.685	161
3 – Miopía	0.905	0.950	0.927	20
4 – Normal	0.769	0.861	0.812	287

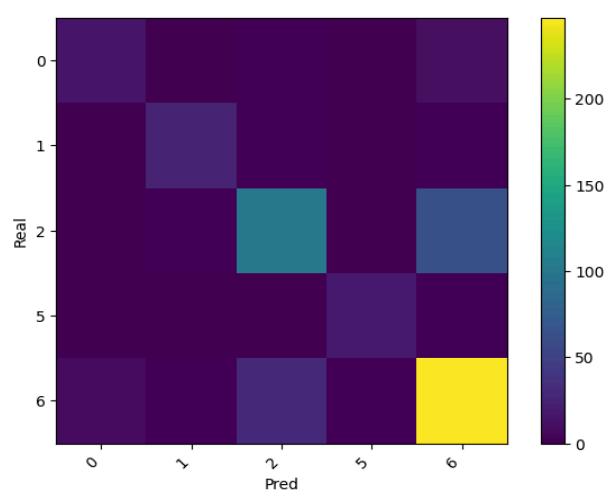
val_classification_report (test)

Agregado	Precisión	Recall	F1-score	Soporte
Accuracy (global)	0.777	--	--	525
Macro avg	0.804	0.777	0.788	525
Weighted avg	0.776	0.777	0.773	525

val_classification_report (test)



val_matriz de confusión



test_matriz de confusión

Evaluación resultados:

Rendimiento global:

- **Validación:** accuracy = **0.729**, F1-macro = **0.748**.
- **Test:** accuracy = **0.777**, F1-macro = **0.788**; precision-macro = **0.804**, recall-macro = **0.777**.

El incremento de **F1-macro** de validación a test sugiere **buena generalización**, coherente con el uso combinado de **EMA** y **TTA + MC-Dropout** durante la inferencia.

Desempeño por clase (test):

- **Miopía (M): F1 = 0.927** y **Catarata (C): F1 = 0.915** muestran el mejor rendimiento.
- **Normal (N): F1 = 0.812** es algo inferior pero aceptable.
- **DMAE (A): F1 = 0.600** y **Retinopatía diabética (D): F1 = 0.685** quedan por detrás, con margen de mejora, especialmente en **recall**.

Modelo final y conclusiones

Arquitectura modelo final (red_neuronal_final.py):

- **Backbone y fusión multimodal.** EfficientNet-B0 preentrenada (se sustituye la *classifier* por Identity para extraer *embeddings*) + rama MLP de metadatos (edad, sexo) y **cabeza de fusión** ($256 \rightarrow n_{\text{clases}}$).
- **Selección de clases y remapeo.** Se entran las clases **[0, 1, 2, 5, 6]** y se remapean a índices consecutivos **0–4** para entrenamiento y evaluación.
- **Normalización de metadatos.** La edad se **estandariza** con media y desviación del **conjunto de entrenamiento** (en NormalizeMetaCollate) para evitar *data leakage*.
- **Desbalance de clases.** En lugar de *class weights* en la pérdida, se utiliza **WeightedRandomSampler** con pesos inversamente proporcionales a la frecuencia por clase (mejora el **recall** de minoritarias).
- **Pérdida y regularización.** CrossEntropyLoss con **label smoothing = 0.096**; **dropout = 0.594** en la cabeza; **gradient clipping max_norm = 5.0**; **weight decay = 6.47e-05**.
- **Programación del LR.** OneCycleLR por *batch* (*pct_start* ≈ 0.29, *div_factor* = 10, *final_div_factor* = 100) con **LR máx. = 3.385e-4**.
- **Estabilización de pesos.** EMA explícita con **ema_decay = 0.9957**; validación y test se realizan sobre el **modelo EMA**.
- **Inferencia robusta.** TTA (flip horizontal) + **MC-Dropout** (5 pasos) en validación y test.
- **AMP y scheduler.** AMP (autocast + GradScaler) si hay CUDA; OneCycleLR se actualiza por iteración.
- **Criterio de early stopping.** Monitoriza **val_acc** (pacienza = 6, tolerancia = 1e-4) y guarda el mejor **checkpoint EMA**; la evaluación final en test se hace con **EMA + TTA/MC**.
- **Trazabilidad.** Registro en **MLflow**: parámetros, métricas por época y artefactos

(classification reports y matrices de confusión).

Líneas de experimentación futuras:

- **Pérdidas focalizadas** en clases difíciles: **Focal Loss o ASL** (Asymmetric Loss).
- Probar **LR discriminativo** (mayor LR en cabeza/MLP que en backbone).
- Ajuste de **umbrales por clase** para optimizar F1/recall donde interese clínicamente.
- **Data augmentation** dirigida a patrones característicos de A (DMAE) y D (Retinopatía Diabética) (p.ej., aplicando técnicas tipo **CLAHE**).

Conclusiones:

El modelo final alcanza un rendimiento sólido y estable (test: accuracy = 0.777; F1-macro = 0.788) y muestra buena capacidad de generalización respecto a validación, en línea con las estrategias de robustez aplicadas (EMA, TTA y MC-Dropout). A nivel de clases, el desempeño es bastante alto en Miopía y Catarata, correcto en Normal, e insuficiente en DMAE (A) y Retinopatía diabética (D), donde el recall sigue siendo el principal cuello de botella. Este patrón sugiere una combinación de desbalance de datos, variabilidad de adquisición (calidad/iluminación) y solapamiento fenotípico con otras categorías, factores habituales en imágenes de fondo de ojo.

En conjunto, el sistema es reproducible y trazable (MLflow) y constituye una base fiable para uso exploratorio. Para elevar su utilidad clínica, el foco debería centrarse en recuperar más verdaderos positivos en las clases A y D. Para ello, proponemos probar con pérdidas enfocadas (**Focal/ASL**), **augmentación dirigida**, **umbralado por clase** y **fine-tuning discriminativo** a mayor resolución. Estas acciones priorizan la sensibilidad donde más impacto tiene.

5.2 Mejora de la propuesta de valor del producto (RAG)

Pipeline RAG

Selección de PDFs + limpieza (transform.py)

Extracción y normalización del texto de las guías (eliminar índices, cabeceras, errores y unificar formato) para obtener .txt limpios listos para procesar.

Chunking (RecursiveCharacterTextSplitter, etc.)

División del texto en fragmentos manejables (p. ej. ~1000 tokens con solapamiento) para mantener contexto y facilitar la vectorización.

Embeddings (SBERT / OpenAI / Cohere)

Conversión de cada chunk a vectores semánticos mediante un modelo de embeddings (ej. MiniLM/SBERT o embeddings comerciales) para búsquedas por similitud.

Indexación (FAISS / otro vectorstore)

Almacenamiento de los vectores en un vectorstore (FAISS) y registro paralelo de los chunks (JSONL) para recuperación eficiente.

Retriever (k = 6–12)

Búsqueda de los k chunks más relevantes para la query (y sus variantes) mediante búsqueda por similitud en el índice; rango típico 6–12.

Fusión de resultados (RRF / Reciprocal Rank Fusion)

Consolidación y desduplicado de las listas recuperadas (de múltiples variantes de la query) priorizando documentos que aparecen en varias búsquedas.

Construcción de prompt (contexto + pregunta)

Concatenación de los chunks fusionados como contexto y ensamblado de la pregunta clínica en una plantilla estandarizada para el LLM.

LLM responde (generación controlada)

Envío del prompt al LLM (temperatura baja) para obtener una respuesta estructurada y basada en la evidencia recuperada (diagnóstico, pruebas, tratamiento, red flags, referencias).

Persistencia e integración

Guardado de índices y chunks; el módulo RAG se integra como componente de apoyo clínico que complementa la predicción por imagen.

El módulo Retrieval-Augmented Generation (RAG) se diseñó para aportar razonamiento clínico basado en evidencia a partir de guías oficiales, complementando la clasificación por imágenes. A continuación se describe el flujo de trabajo con enfoque técnico y, en cada fase, las secciones: Qué se hizo, Resultados y Decisión. Se incluyen alternativas valoradas (PyMuPDF, plantillas por rol, temperature=0, RAG Fusion vs Hybrid Retrieval, chunking con Cohere).

1) Selección de PDFs + limpieza (transform.py)

Qué se hizo.

- Recopilación de guías y consensos por patología (retinopatía, glaucoma, catarata, DMAE, miopía).
- Extracción de texto desde PDF y limpieza: eliminación de índices, cabeceras/pies repetidos, normalización Unicode y consolidación en párrafos.

Resultados.

- Colección curada por enfermedad con trazabilidad de fuentes.
- TXT limpios y estructurados listos para el chunking.

Decisión.

- Mantener organización por patología para aumentar precisión del retrieval.
- Se valoró PyMuPDF para eliminar cabeceras/pies por posición y gestionar páginas con imágenes; se pospone por simplicidad del MVP.

2) Chunking (segmentación en fragmentos)

Qué se hizo.

- Segmentación del texto en ~1.000 tokens con solapamiento de 150 para preservar continuidad semántica.

Resultados.

- Fragments con suficiente contexto clínico; reducción de cortes en secciones críticas.
- Registro del número de chunks por documento/grupo para control de calidad.

Decisión.

- Mantener 1.000/150 por equilibrio entre contexto y coste.
- Se consideró el chunking de Cohere (separadores semánticos) pero se pospuso para evitar nuevas dependencias en el MVP.

3) Embeddings (SBERT)

Qué se hizo.

- Vectorización de chunks con Sentence-Transformers 'all-MiniLM-L6-v2' (normalización L2, similitud por producto interno).

Resultados.

- Vectores compactos y rápidos de generar; buena relación calidad/latencia.
- Cohesión semántica por patología gracias a índices separados.

Decisión.

- Elegir MiniLM-L6-v2 por coste y rendimiento en MVP.
- Alternativas (OpenAI/Cohere embeddings) para fases con mayores requisitos de recall o presupuesto.

4) Indexación (FAISS / vectorstore)

Qué se hizo.

- Construcción de índices FAISS (IndexFlatIP) por patología y JSONL espejo con el texto de cada chunk.

Resultados.

- Vectorstores por enfermedad (p. ej., retinopatía, catarata, DMAE, miopía).
- Recuperación eficiente y local.

Decisión.

- FAISS por rendimiento, simplicidad y despliegue offline.

5) Retriever (k = 6–12)

Qué se hizo.

- Carga de índice y modelo de embeddings; recuperación de top-k chunks para la consulta.

Resultados.

- Recuperaciones consistentes con buen equilibrio coste-precisión a k=6.

Decisión.

- Fijar k=6 como valor por defecto (rango operativo recomendado 6–12).

6) Expansión y fusión de resultados (RAG Fusion)

Qué se hizo.

- Generación de variantes semánticas de la query con LLM y búsqueda por cada variante.
- Consolidación con Reciprocal Rank Fusion (RRF) y desduplicación para crear el contexto final.

Resultados.

- Mayor cobertura y robustez: los documentos que aparecen en varias listas son priorizados.
- Menor dependencia de la redacción exacta de la consulta.

Decisión.

- Adoptar RAG Fusion por robustez semántica y simplicidad de integración.
- Hybrid Retrieval (considerado): añadir un canal léxico (BM25/keywords) para no perder términos exactos (fármacos, siglas). Se pospone para mantener el MVP simple; se deja como mejora prioritaria si crece el corpus.

7) Construcción de prompt (contexto + pregunta)

Qué se hizo.

- Plantilla clínica estructurada con secciones: diagnóstico diferencial, pruebas, tratamiento, red flags, seguimiento, comunicación con paciente, información faltante, nivel de confianza y referencias.
- Contexto construido con los chunks fusionados + pregunta clínica parametrizada (edad, sexo, patología).

Resultados.

- Respuestas uniformes, auditables y basadas en evidencia del contexto.
- Identificación explícita de información faltante.

Decisión.

- Se prioriza la plantilla para rol 'médico' por alineamiento con el MVP.
- Se deja la extensión a otros roles (paciente, técnico) como mejora futura usando la misma estructura.

8) LLM responde (inferencia)

Qué se hizo.

- Uso de ChatOpenAI (gpt-4o-mini) con temperature=0.0 para maximizar determinismo y adherencia al contexto.

Resultados.

- Salidas estables y reproducibles; menor riesgo de alucinación.

Decisión.

- Conservar temperature=0 por seguridad y trazabilidad clínica; aumentar temperatura solo en tareas no clínicas.

9) Persistencia e integración en el proyecto

Qué se hizo.

- Persistencia de índices FAISS y chunks por patología; integración del RAG como módulo de apoyo clínico junto al clasificador de imágenes.

Resultados.

- Flujo end-to-end: clasificación de imagen → recomendaciones clínicas sustentadas.
- Modularidad para incorporar nuevas guías o patologías (re-chunk + re-index del grupo correspondiente).

Decisión.

- Mantener segmentación por enfermedad como contrato de integración entre visión y RAG.
- Dejar preparado el camino para Hybrid Retrieval y PyMuPDF en versiones futuras si aumenta el ruido o la heterogeneidad de maquetación.

Relevancia para el desarrollo posterior

- Validez clínica: el sistema no solo clasifica, también explica y recomienda apoyándose en guías verificables.
- Escalabilidad: añadir nuevas patologías/documentos sin cambios de arquitectura (añadir grupo → limpiar → chunkear → indexar).
- Robustez de recuperación: RAG Fusion reduce la dependencia de una única redacción; Hybrid Retrieval queda como incremento de recall para términos exactos.

6. MVP desplegado

El sistema se compone de dos componentes principales: un ‘backend’ desarrollado con FastAPI que aloja el modelo predictivo y gestiona las solicitudes de inferencia, y un ‘frontend’ construido con Streamlit que proporciona una interfaz amigable para la interacción con los usuarios. Esta arquitectura permite no solo realizar predicciones precisas, sino también gestionar eficientemente los datos de los pacientes y generar reportes detallados.

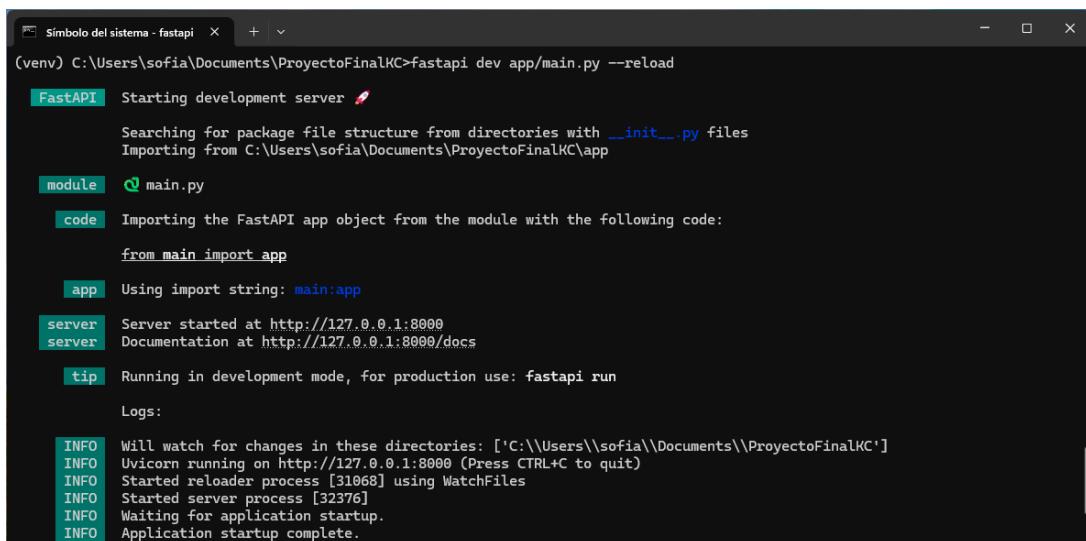
6.1 Arquitectura del Sistema

Diseño

La arquitectura del sistema sigue un patrón cliente-servidor que separa claramente las responsabilidades entre la lógica de negocio (backend) y la presentación (frontend). Esta separación ofrece ventajas significativas en términos de mantenibilidad, escalabilidad y flexibilidad.

El backend, implementado con FastAPI, se encarga de:

- Alojar el modelo de aprendizaje profundo para la clasificación de imágenes retinales
- Validar y procesar las entradas de los usuarios
- Realizar inferencias utilizando el modelo entrenado
- Proveer endpoints RESTful para la comunicación con el frontend



```
Simbolo del sistema - fastapi
(venv) C:\Users\sofia\Documents\ProyectoFinalKC>fastapi dev app/main.py --reload
FastAPI Starting development server 🚀
Searching for package file structure from directories with __init__.py files
Importing from C:\Users\sofia\Documents\ProyectoFinalKC\app
module main.py
code Importing the FastAPI app object from the module with the following code:
from main import app
app Using import string: main:app
server Server started at http://127.0.0.1:8000
server Documentation at http://127.0.0.1:8000/docs
tip Running in development mode, for production use: fastapi run
Logs:
INFO Will watch for changes in these directories: ['C:\\Users\\sofia\\Documents\\\\ProyectoFinalKC']
INFO Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO Started reloader process [31068] using WatchFiles
INFO Started server process [32376]
INFO Waiting for application startup.
INFO Application startup complete.
```

El frontend, desarrollado con Streamlit, proporciona:

- Una interfaz intuitiva para la carga de imágenes y datos del paciente
- Visualización de resultados de forma clara y comprensible
- Funcionalidades adicionales como generación de reportes y acceso a información contextual
- Gestión de consultas con histórico de entradas y resultados.

Uso de Docker Compose para contenedores

El sistema se despliega utilizando Docker Compose, que permite orquestar múltiples contenedores de forma coordinada, ofreciendo varias ventajas: aislamiento de componentes, donde cada servicio (backend, frontend) se ejecuta en su propio contenedor evitando conflictos de dependencias; facilidad de despliegue, ya que el sistema completo se puede desplegar con un único comando (docker compose up); y escalabilidad, al permitir escalar individualmente cada componente según las necesidades.

```
(py310) mapardo@mapardo-260-p126ns:~/KeepCoding/ProyectoFinalKC/app$ docker compose up
WARN[0000] /home/mapardo/KeepCoding/ProyectoFinalKC/app/docker-compose.yml: the attribute `version` is obsolete,
[+] Running 2/2
  ✓ Container app-fastapi-1   Created
  ✓ Container app-streamlit-1 Created
Attaching to fastapi-1, streamlit-1
streamlit-1 | Collecting usage statistics. To deactivate, set browser.gatherUsageStats to false.
streamlit-1 |
streamlit-1 | You can now view your Streamlit app in your browser.
streamlit-1 |
streamlit-1 | URL: http://0.0.0.0:8501
streamlit-1 |
fastapi-1  INFO:     Started server process [1]
fastapi-1  INFO:     Waiting for application startup.
fastapi-1  INFO:     Application startup complete.
fastapi-1  INFO:     Uvicorn running on http://0.0.0.0:8080 (Press CTRL+C to quit)
[...]
W Enable Watch
```

Flujo de datos

El flujo de datos a través del sistema sigue una secuencia bien definida:

1. El usuario carga una imagen y proporciona información sobre el paciente a través de la interfaz.
2. Streamlit envía estos datos al backend de FastAPI mediante una solicitud HTTP POST
3. FastAPI valida los datos recibidos y prepara la imagen para el modelo
4. El modelo, a partir de la imagen procesada y los datos del paciente, genera una predicción
5. FastAPI devuelve los resultados a Streamlit en formato JSON
6. Streamlit presenta los resultados al usuario y ofrece opciones adicionales

Sistema de registros.

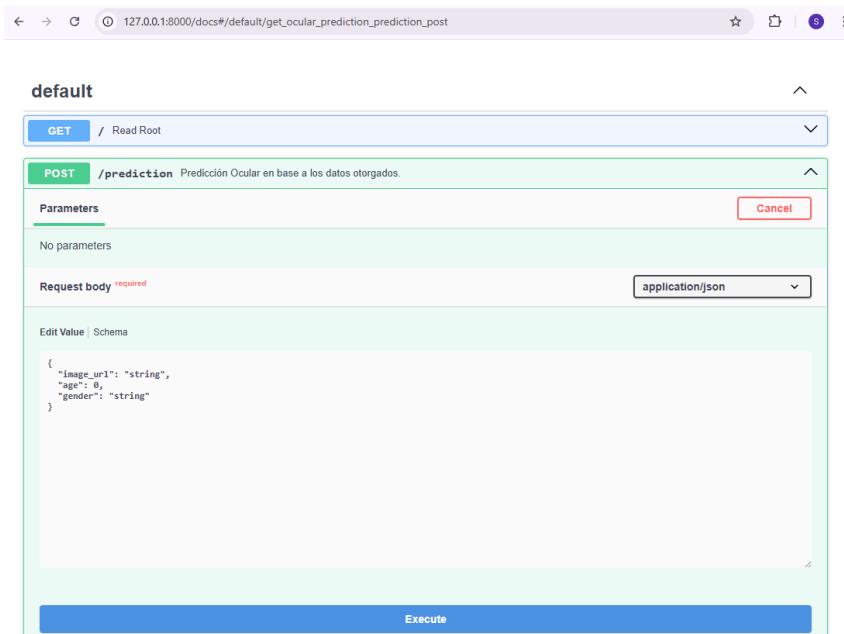
Se implementa un sistema completo de logging que registra todas las consultas en un archivo CSV (app\prediction_log.csv). Este sistema permite el seguimiento de todas las actividades realizadas a través de la aplicación, facilitando la auditoría y el análisis posterior.

6.2 Implementación ‘backend’ con FastAPI

El backend implementado con FastAPI sigue los principios de diseño de APIs RESTful y aprovecha las ventajas del framework para crear endpoints eficientes y bien documentados.

El endpoint /prediction procesa las solicitudes de inferencia:

- Procesado de la imagen para adaptarla a los requisitos del modelo (redimensionamiento, normalización, ...)
- Ejecución de la inferencia utilizando usando el modelo
- Procesa los resultados para convertirlos en formatos comprensibles



6.3 Implementación ‘frontend’ con Streamlit

La interfaz de usuario desarrollada con Streamlit prioriza la usabilidad y la claridad en la presentación de información.

A screenshot of a Streamlit application window titled 'Sistema de Predicción Ocular'. The main content area is titled 'Predicciones detalladas para la Salud Visual' and contains a bulleted list of eye diseases: Degeneración Macular Relacionada con la Edad (DMAE), Catarata, Retinopatía Diabética, Miopía Patológica, and Ojo Normal. Below this is a section titled 'Más allá de la Predicción' with explanatory text about the AI system's capabilities.

En primer lugar, mediante un formulario, se recoge toda la información necesaria para realizar la predicción (imagen y metadatos).

Información del Paciente

Edad
55 - +

Género
Femenino

Image

Elige una imagen en formato jpg

Drag and drop file here
Limit 200MB per file • JPG, JPEG

Browse files

35_left.jpg 12.9KB

x

Predecir

El ‘frontend’ implementa un sistema robusto para el manejo de archivos temporales. Este enfoque garantiza que las imágenes se almacenen de manera organizada y con nombres únicos que previenen conflictos. Cuando se lanza la inferencia a través del modelo, se muestra la imagen cargada y una vez terminado el cálculo se muestran los resultados. La forma de mostrar los resultados permite a los usuarios conocer no sólo el diagnóstico principal sino también el nivel de certeza del modelo para cada enfermedad.

Predicción exitosa!

Predicción

Catarata



Imagen

Distribución de Probabilidad entre Enfermedades

Degeneración macular relacionada con la edad: 1.26%



Catarata: 98.74%

Retinopatía diabética: 0.00%

Miopía patológica: 0.00%

Normal: 0.00%

6.3 Integración de Sistema RAG para Información Contextual

El sistema incorpora un módulo de Retrieval Augmented Generation (RAG) que proporciona información contextual adicional sobre las enfermedades oculares detectadas. Este módulo enriquece la experiencia del usuario al proporcionar información médica relevante y personalizada según la condición detectada y las características del paciente.

Información Adicional sobre la condición predicha

 Obtener información

 Exportar Reporte

Generación de Reportes

El sistema puede generar reportes completos que incluyen:

- Información del paciente
- Resultados de la predicción
- Información contextual obtenida mediante RAG
- Distribución de probabilidades

Información Adicional sobre la condición predicha

 Obtener información

Información obtenida correctamente.

◆ Diagnóstico diferencial

- Nombre: Catarata
 - Probabilidad estimada: Alta
 - Razonamiento: La paciente tiene 55 años y presenta un diagnóstico de catarata, lo cual es común en esta franja etaria.
 - Nivel de evidencia: Moderado
 - Confianza: Alta
- ◆ Pruebas diagnósticas recomendadas
- Prueba: Examen oftalmológico completo
 - Prioridad: Alta
 - Hallazgos esperados: Evaluación de agudeza visual, examen de biomicroscopía, medición de presión intraocular, y exploración del fondo de ojo.
 - Contraindicaciones: Ninguna relevante para esta prueba.
 - Referencias: Guías de Práctica Clínica de la American Academy of Ophthalmology y NICE.
- ◆ Opciones de tratamiento
- Opción: Cirugía de catarata (facoemulsificación o extracción extracapsular)
 - Indicación: Cuando la catarata afecta significativamente la calidad de vida o la agudeza visual.

Referencias

- Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
(Available at: <https://www.deeplearningbook.org/>)
- LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436–444.
<https://doi.org/10.1038/nature14539>
- LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278–2324.
<https://doi.org/10.1109/5.726791>
- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems* (NeurIPS), 25, 1097–1105.
- Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W. T., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems* (NeurIPS), 33, 9459–9474. <https://arxiv.org/abs/2005.11401>
- Gao, Y., Xiong, Y., Gao, X., Jia, K., & Pan, J. (2023). Retrieval-Augmented Generation for Large Language Models: A Survey. arXiv preprint. <https://arxiv.org/abs/2312.10997>
- Lathkar, M. (2023). High-Performance Web Apps with FastAPI: The Asynchronous Web Framework Based on Modern Python. Apress. <https://doi.org/10.1007/978-1-4842-9178-8>
- Kassambara, A. (2024, February 8). Building REST APIs with FastAPI: A Modern Python Framework. <https://www.datanovia.com/learn/programming/python/tools/fastapi-rest-api.html>
- Vogt, L. (2023). Streamlit for Data Science: Create interactive data apps in Python. Packt Publishing Limited. ISBN: 978-1-80323-295-9.
- Whitmire, A. L. (2021, September 27). Seaside Librarian: The Streamlit app is alive (sort of!). <https://amandawhitmire.github.io/blog/posts/2021-09-27-streamlit-app-alive/>
- Verma, P. S. (2023). Machine Learning Engineering with MLflow: Manage the end-to-end machine learning life cycle with MLflow. Packt Publishing Limited. ISBN: 978-1-80056-169-4.