



Escuela  
Politécnica  
Superior

# Estudio experimental de modelos de inteligencia artificial clásicos y profundos en la detección de degeneración retiniana en modelos murinos experimentales.



Grado en Ingeniería Biomédica

## Trabajo Fin de Grado

Autor:

Lucía Fernández López

Tutor/es:

Natalia Martínez Gil

Alberto Antonio De Ramón Fernández



Universitat d'Alacant  
Universidad de Alicante



# Estudio experimental de modelos de inteligencia artificial clásicos y profundos en la detección de degeneración retiniana en modelos murinos experimentales.

---

## **Autor**

Lucía Fernández López

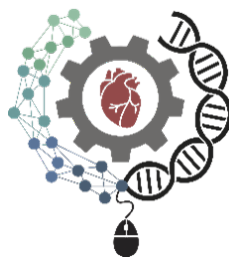
## **Tutor/es**

Natalia Martínez Gil

*Fisiología, Genética y Microbiología*

Alberto Antonio De Ramón Fernández

*Tecnología Informática y computación*



Grado en Ingeniería Biomédica



Escuela  
Politécnica  
Superior



Universitat d'Alacant  
Universidad de Alicante

ALICANTE, Octubre 2025



# Agradecimientos

Muchísimas gracias a mis padres, María José y Juan Manuel y a mi hermana Ana por haberme apoyado y acompañado durante toda la vida, por vuestro amor incondicional y porque sin vosotros no sería quién soy a día de hoy. Muchas gracias, a Lola, Meri y Yosra, os habéis convertido en unas verdaderas amigas para mí, de esas de para toda la vida. También me gustaría agradecer a mis dos tutores Natalia Martínez Gil y Alberto de Ramón Fernández, por haberme ayudado a la realización de este trabajo.



*Allá muevan feroz guerra  
ciegos reyes por un palmo más de tierra,  
que yo tengo aquí por mío  
cuanto abarca el mar bravío.  
Y si caigo, ¿qué es la vida?  
Por perdida ya la di,  
cuando el yugo de un esclavo  
como un bravo sacudí.*

José de Espronceda (Canción del pirata)





# Resumen

La distrofia corioidea areolar central (por sus siglas en inglés, CACD) es un trastorno macular hereditario, que se caracteriza por una degeneración progresiva de la mácula, una región localizada en el centro de la retina. Esta patología suele manifestarse en individuos adultos, comúnmente entre los 30 y 60 años de edad.

Como consecuencia, los pacientes experimentan una pérdida gradual de agudeza visual. Esta pérdida afecta significativamente la calidad de vida. Dado que se trata de un trastorno hereditario y neurodegenerativo, su diagnóstico temprano y la evaluación del historial familiar son aspectos clave para su abordaje clínico.

En este contexto, el presente Trabajo de Fin de Grado (TFG) tiene como objetivo el desarrollo de un algoritmo de clasificación binaria, capaz de distinguir entre retinas degeneradas y no degeneradas, utilizando datos experimentales proporcionados por el grupo de investigación NEUROVIS. Dichos datos provienen de diversas pruebas realizadas previamente en el laboratorio, lo que garantiza una base sólida y realista para el entrenamiento del modelo.

La metodología propuesta se basa en la aplicación de técnicas avanzadas de aprendizaje automático y aprendizaje profundo, explorando y evaluando distintas arquitecturas con el fin de optimizar la precisión del sistema de clasificación. Una mejora en la precisión del modelo supondría un avance significativo en la detección temprana de la degeneración retiniana en modelos animales, como el ratón, permitiendo reducir el número de pruebas innecesarias, ahorrar recursos y acelerar la investigación hacia posibles tratamientos para esta patología.



# Abstract

Central areolar choroidal dystrophy (CACD) is a hereditary macular disorder characterized by the progressive degeneration of the macula, a region located at the center of the retina responsible for sharp central vision. This condition typically manifests in adulthood, most commonly between the ages of 30 and 60.

As a result, patients experience a gradual loss of visual acuity, which significantly impacts their quality of life. Given its hereditary and neurodegenerative nature, early diagnosis and the assessment of family history are essential components for effective clinical management.

In this context, the present Final Degree Project aims to develop a binary classification algorithm capable of distinguishing between degenerated and non-degenerated retinas, using experimental data provided by the NEUROVIS research group. These data are derived from various laboratory tests previously conducted, ensuring a solid and realistic foundation for model training.

The proposed methodology is based on the application of advanced machine learning and deep learning techniques, exploring and evaluating different architectures in order to optimize the classification system's accuracy.

An improvement in model precision would represent a significant step forward in the early detection of retinal degeneration in animal models, such as mice. This could lead to a reduction in unnecessary testing, savings in time and resources, and faster progress in the search for potential treatments for this condition.



# Índice general

<b>Resumen</b>	<b>ix</b>
<b>Abstract</b>	<b>xi</b>
<b>1 Introducción</b>	<b>1</b>
1.1 Motivación . . . . .	1
1.2 Objetivos . . . . .	2
<b>2 Marco Teórico</b>	<b>5</b>
2.1 Retina y Visión . . . . .	5
2.2 Mutación en el gen de la Periferina . . . . .	6
2.3 Pruebas para la degeneración de la retina . . . . .	8
2.3.1 Prueba Optomotora . . . . .	8
2.3.2 Electroretinografía . . . . .	9
2.3.3 Tomografía de coherencia Óptica . . . . .	11
2.4 Arquitecturas de aprendizaje automático . . . . .	13
2.4.1 Árboles de Decisión (del inglés, <i>Random Forest</i> ) . . . . .	13
2.4.2 XGBoost . . . . .	13
2.4.3 Máquinas de Vectores de Soporte . . . . .	15
2.4.4 Naive Bayes . . . . .	15
2.4.5 Regresión Logística (en inglés <i>Logistic Regression</i> ) . . . . .	16
2.5 Redes neuronales: origen, estructura y arquitecturas utilizadas . . . . .	16
2.5.1 Origen histórico y modelo de McCulloch-Pitts . . . . .	16
2.5.2 Estructura funcional de las redes neuronales . . . . .	17
2.5.3 Arquitecturas utilizadas: DNN y Perceptrón Multicapa . . . . .	20

2.6	Técnicas de aumento de datos . . . . .	22
2.6.1	Perturbación Aleatoria ( <i>Random Noise</i> ) . . . . .	22
2.6.2	Escalado ( <i>Scaling</i> ) . . . . .	22
2.6.3	Generación de Muestras Sintéticas (del inglés, <i>Synthetic Minority Over-sampling Technique</i> , SMOTE) . . . . .	23
2.6.4	<i>Bootstrap Resampling</i> . . . . .	23
<b>3</b>	<b>Estado del Arte</b>	<b>25</b>
<b>4</b>	<b>Herramientas utilizadas</b>	<b>27</b>
4.1	Hardware . . . . .	27
4.2	Software . . . . .	27
4.3	Recursos Humanos . . . . .	28
<b>5</b>	<b>Metodología Aplicada y Resultados obtenidos</b>	<b>31</b>
5.1	Adquisición de datos . . . . .	31
5.2	Preprocesamiento de los datos . . . . .	33
5.2.1	Edición de variables desde Excel . . . . .	33
5.2.2	Preparación de la base de datos . . . . .	34
5.3	Planteamiento de los escenarios . . . . .	37
5.4	Primer escenario . . . . .	37
5.4.1	Preparación de datos . . . . .	38
5.4.2	Entrenamiento de arquitecturas ML . . . . .	38
5.4.2.1	Datos escalados . . . . .	41
5.4.2.2	Optimización de hiperparámetros . . . . .	43
5.4.2.3	Nuevas arquitecturas . . . . .	46
5.4.3	Entrenamiento de arquitecturas DL . . . . .	48
5.4.3.1	Eliminar <i>Dropout</i> . . . . .	53
5.4.4	Reducir el número de neuronas . . . . .	54
5.4.5	Data Augmentation . . . . .	55
5.4.6	Matriz de correlación . . . . .	62

---

---

5.5	Segundo escenario . . . . .	64
5.5.1	Preparación de datos . . . . .	64
5.5.2	Entrenamiento de arquitecturas ML . . . . .	64
5.5.3	<i>Cross Validation</i> . . . . .	67
5.5.4	Entrenamiento de arquitecturas DL . . . . .	68
5.6	Creación y Gestión de la base de Datos . . . . .	70
5.7	Aplicación web . . . . .	71
5.7.1	Arquitectura del Sistema y componentes . . . . .	71
5.7.2	Integración del modelo en web . . . . .	74
<b>6</b>	<b>Discusión y Líneas futuras</b>	<b>77</b>
<b>7</b>	<b>Costes</b>	<b>79</b>
<b>8</b>	<b>Conclusiones</b>	<b>81</b>
	<b>Bibliografía</b>	<b>83</b>

---





## Índice de figuras

2.1	Globo ocular en el que están representado las capas que lo componen: Túnica externa, intermedia y retina. Anatomía del ojo. Recuperado el 25 de abril de 2025 Autor desconocido (s.f.). . . . .	5
2.2	: Características morfológicas de los receptores Lledó Riquelme y Cuenca Navarro (2010) . . . . .	6
2.3	Segmento externo alterado en la retina del modelo C57BL/6J-Prph <sup>em1sal</sup> . Tanto el ratón con la mutación en homocigosis (derecha, Prph2 <sup>KIKI</sup> ) como en heterocigosis (centro, Prph2 <sup>WTKI</sup> ), muestran un desarrollo anormal de los segmentos externos de los fotorreceptores si se comparan con la retina de los ratones sanos (izquierda, Prph2 <sup>WTWT</sup> ) Ruiz-Pastor y cols. (2023). . . . .	7
2.4	A la izquierda Argos: Un sistema para evaluar la función visual en pequeños animales de investigación, fundamental en estudios preclínicos de enfermedades neurodegenerativas. A la derecha dibujo de su funcionamiento. Modificado a partir de Instead Technologies (s.f.) y Ruiz-Pastor y cols. (2023). . . . .	8
2.5	:Dispositivo OcuScience® HMsERG utilizado para la evaluación electrofisiológica de la visión en ratones OcuScience (s.f.) . . . . .	9
2.6	Disminución dependiente de la edad de la respuesta retiniana en los ratones doble mutantes (Prph2 <sup>KIKI</sup> ) y heterocigotos para la mutación (Prph2 <sup>WTKI</sup> ), comparado con los ratones sanos (Prph2 <sup>WTWT</sup> ). Modificado a partir de Ruiz-Pastor y cols. (2023) . . . . .	10
2.7	Plataforma de imagen SPECTRALIS® OCT: combinación de imagen de fondo con láser de escaneo y OCT simultáneo Heidelberg Engineering (s.f.) . . . . .	11

2.8	Alteraciones estructurales en las retinas de ratones mutantes <i>Prph2</i> evaluadas <i>in vivo</i> mediante tomografía de coherencia óptica (OCT) a lo largo de la edad. Nótese cómo disminuye el espesor de la retina total y de las diferentes capas en los ratones doble mutantes (derecha, <i>Prph2</i> <sup>KIKI</sup> ) y heterocigotos para la mutación (centro, <i>Prph2</i> <sup>WTKI</sup> ), en comparación con los ratones sanos (izquierda, <i>Prph2</i> <sup>WTWT</sup> ). Ruiz-Pastor y cols. (2023)	12
2.9	Marco conceptual del clasificador Random Forest Hemanth y cols. (s.f.)	13
2.10	Marco conceptual de XGBoost IBM (s.f.)	14
2.11	Margen de un hiperplano de separación: (a) hiperplano de separación no-óptimo y su margen asociado (no máximo) (b) hiperplano de separación óptimo y su margen asociado máximo. Carmona (2016)	15
2.12	Esquema de una neurona de McCulloch-Pitts, con dos dendritas y un axón La Máquina Oráculo (s.f.)	16
2.13	Red multicapa Izaurieta y Saavedra (s.f.)	18
2.14	Esquema del proceso de retropropagación Analytics Vidhya (2023)	20
2.15	Ejemplo de arquitectura MLP Taud y Mas (2018)	21
5.1	Imagen que muestra la tabla de Excel utilizada por el grupo de investigación como Base de datos original	31
5.2	Conjunto de datos preparado para entrenar	36
5.3	Matriz de confusión de XGBoost con hiperparámetros por defecto	40
5.4	Matriz de confusión de SVM con hiperparámetros por defecto	40
5.5	Matriz de confusión de Random Forest con hiperparámetros por defecto	41
5.6	Matriz de confusión de XGBoost con hiperparámetros por defecto y datos escalados	42
5.7	Matriz de confusión de Random Forest con hiperparámetros por defecto y datos escalados	42
5.8	Matriz de confusión de XGBoost con hiperparámetros optimizados	45
5.9	Matriz de confusión de Random Forest con hiperparámetros optimizados	45
5.10	Matriz de confusión de SVM con hiperparámetros optimizados	46
5.11	Matriz de confusión de Regresión Logística	47

---

5.12 Matriz de confusión de Naive Bayes . . . . .	47
5.13 Arquitectura del modelo MLP implementado . . . . .	50
5.14 Arquitectura del modelo DNN implementado . . . . .	51
5.15 Matriz de confusión de MLP con hiperparámetros por defecto . . . . .	52
5.16 Matriz de confusión de DNN con hiperparámetros por defecto . . . . .	52
5.17 Matriz de correlación con las variables de purebas ERG . . . . .	62
5.18 Matriz de correlación con todas las variables . . . . .	63
5.19 Matriz de confusión de XGBoost con GridSearchCV y Bootstrap aplicado . .	66
5.20 Matriz de confusión de SVM con GridSearchCV y Bootstrap aplicado . . . .	66
5.21 Matriz de confusión de Random Forest con GridSearchCV y Bootstrap aplicado	67
5.22 Base de datos creada desde <i>MySQL Workbench</i> . . . . .	71
5.23 Arquitectura cliente-servidor en aplicaciones web Mozilla (s.f.) . . . . .	71
5.24 Vista final de la interfaz web desarrollada. La aplicación permite al usuario introducir los inputs a través de un formulario dinámico. . . . .	74
5.25 Resultado de una clasificación binaria. La interfaz muestra si el caso ingresado presenta o no degeneración . . . . .	75
5.26 Vista final de la interfaz web en la vista consultor . . . . .	75

---



## Índice de tablas

3.1	Resumen comparativo de estudios sobre clasificación de enfermedades mediante inteligencia artificial aplicada a imágenes OCT y de fondo de ojo. . . . .	26
5.1	Precisión de cada modelo con hiperparámetros por defecto. . . . .	39
5.2	Precisión de <i>XGBoost</i> y <i>Random Forest</i> entrenados con datos escalados. . . .	41
5.3	Precisión de cada modelo con <i>GridSearchCV</i> aplicado. . . . .	44
5.4	Precisión de las arquitecturas de Regresión logística y Naive Bayes. . . . .	46
5.5	Precisión y Matriz de confusión de cada una de las redes neuronales. . . . .	52
5.6	Precisión y Matriz de confusión de cada una de las redes neuronales sin Dropout. .	53
5.7	Precisión y Matriz de confusión de cada una de las redes neuronales con menos neuronas por capa. . . . .	54
5.8	Precisión y Matriz de confusión de random Forest con las diferentes técnicas de Data Augmentation . . . . .	56
5.9	Precisión y Matriz de confusión de SVM con las diferentes técnicas de Data Augmentation . . . . .	57
5.10	Precisión y Matriz de confusión de <i>XGBoost</i> con las diferentes técnicas de Data Augmentation . . . . .	58
5.11	Precisión y Matriz de confusión de Regresión Logística con las diferentes técnicas de Data Augmentation . . . . .	59
5.12	Precisión y Matriz de confusión de Naive Bayes con las diferentes técnicas de Data Augmentation . . . . .	60
5.13	Precisión obtenida en MLPL por cada técnica de <i>Data Augmentation</i> . . . . .	61
5.14	Precisión obtenida en DNN por cada técnica de <i>Data Augmentation</i> . . . . .	61

5.15	Precisión de arquitecturas de aprendizaje automático sin <i>GridSearchCV</i> ni <i>Data Augmentation</i> . . . . .	64
5.16	Resultados tras aplicar <i>GridSearchCV</i> , sin <i>Data Augmentation</i> . . . . .	65
5.17	Precisión de <i>XGBoost</i> , <i>SVM</i> y <i>Random Forest</i> entrenados con hiperparámetros óptimos y datos aumentados . . . . .	66
5.18	Precisión en validación cruzada para <i>XGBoost</i> con 10 y 5 particiones . . . . .	68
5.19	Resultados de precisión con <i>Dropout</i> , sin <i>Data Augmentation</i> . . . . .	68
5.20	Resultados de precisión tras eliminar <i>Dropout</i> , sin <i>Data Augmentation</i> . . . . .	69
5.21	Resultados obtenidos de las redes neuronales entrenadas con hiperparámetros óptimos y datos aumentados . . . . .	69
7.1	Estimación de costes técnicos y de personal para el desarrollo del proyecto. . . . .	79

---

# 1 Introducción

El avance de la inteligencia artificial (IA) ha transformado el ámbito de la biomedicina y medicina, permitiendo abordar problemas complejos que antes requerían procesos manuales largos y tediosos de una forma mucho más cómoda y rápida. La creación de modelos, el aprendizaje automático (del inglés, Machine Learning, ML) y el aprendizaje profundo (del inglés, Deep Learning, DL) han demostrado ser herramientas poderosas para el análisis y clasificación de datos en investigaciones biomédicas Edreira y cols. (2021).

La detección de la degeneración retiniana en animales de experimentación especialmente en ratones, supone un reto significativo debido no sólo a la necesidad de un gran número de animales de estudio, sino también de la realización de múltiples pruebas experimentales para evaluar el estado de la retina. Actualmente, este proceso depende de la interpretación manual de los datos obtenidos en distintas pruebas que requieren el trabajo de un equipo numeroso de personas. Para optimizar este procedimiento y reducir la variabilidad de los resultados obtenidos el desarrollo de un modelo de clasificación basado en IA representa una alternativa innovadora con un enorme potencial.

El modelo permitiría analizar los datos obtenidos en el laboratorio y predecir si la retina está degenerada o no en ratones modificados genéticamente, facilitando el trabajo del equipo de investigación, reduciendo el tiempo y optimizando recursos en el laboratorio.

## 1.1 Motivación

A nivel global, se han identificado tres familias que comparten una patología degenerativa de origen genético que afecta la retina. Estas familias, ubicadas en Japón, Alemania y el norte de España (Cantabria) Ruiz-Pastor y cols. (2023), padecen Distrofia Coroidea Areolar Central (por sus siglas en inglés, CACD) y desarrollan ceguera progresiva y una pérdida

en la percepción del color debido a un fallo en la renovación del segmento externo de los fotorreceptores Albertos-Arranz y cols. (2021).

Actualmente, el grupo de investigación NEUROVIS del Departamento de Fisiología, Genética y Microbiología de la Universidad de Alicante, está investigando esta enfermedad rara Albertos-Arranz y cols. (2021). Para ello, emplean modelos murinos genéticamente modificados que reproducen los efectos de la enfermedad, permitiendo analizar su progresión y evaluar posibles tratamientos Albertos-Arranz y cols. (2021).

Sin embargo, uno de los principales desafíos al que enfrentan estos investigadores es determinar con precisión el estado de degeneración de la retina en los ratones, no sólo para estudiar la enfermedad, sino también para poder analizar el efecto de posibles terapias. En la actualidad, este proceso implica la realización de múltiples pruebas experimentales, lo que supone una gran inversión de personal, tiempo y recursos económicos y técnicos.

Además, mi motivación personal para realizar este TFG es culminar mis estudios en Ingeniería Biomédica en la Universidad de Alicante y, con ello, abrir la puerta a nuevas oportunidades, explorar otros campos y vivir nuevas experiencias.

## 1.2 Objetivos

Este proyecto tiene como objetivo aplicar un modelo de IA para clasificar el estado de degeneración de la retina en ratones modificados genéticamente. Para ello, el modelo analizará los valores obtenidos en pruebas específicas realizadas en el laboratorio, permitiendo una evaluación más rápida y precisa de la progresión de la enfermedad. Esto permitirá una identificación más temprana de la enfermedad, optimizando el uso de recursos y reduciendo el tiempo necesario para su análisis.

Para abordar este objetivo general, se proponen los siguientes objetivos específicos para crear un modelo:

- Revisión de la literatura: se realizará un estudio de artículos académicos previos en los que se haya aplicado inteligencia artificial para el análisis de datos de pruebas ópticas en ratones. El objetivo es identificar enfoques existentes que puedan servir de referencia para optimizar los resultados obtenidos en este proyecto.
-



- 
- Estudio de diferentes modelos: se explorarán diversas técnicas de clasificación para detectar la degeneración retiniana en ratones de manera más eficiente.
  - Evaluación del rendimiento: se llevará a cabo un preprocesamiento de datos y se probarán distintos modelos y técnicas de optimización para maximizar el rendimiento del sistema y obtener la mayor precisión posible en los resultados.
  - Documentación del proceso: se registrarán detalladamente todas las decisiones tomadas a lo largo del desarrollo del proyecto, incluyendo la metodología utilizada y los resultados obtenidos. Obteniendo conclusiones objetivas sobre la problemática abordada y evaluar la efectividad del modelo propuesto.
  - Creación de Base de Datos: además, se creará una base de datos para facilitar el acceso y la consistencia de los datos.

De este modo, este TFG busca aportar una solución innovadora y eficiente, combinando la IA y la investigación médica, con el objetivo final de contribuir al desarrollo de metodologías más precisas que, en el futuro, podrían ser aplicadas a la detección temprana de enfermedades oculares en humanos.

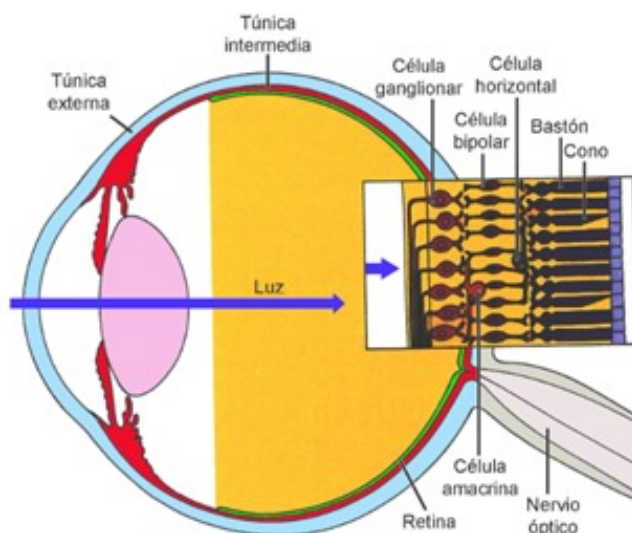
---



## 2 Marco Teórico

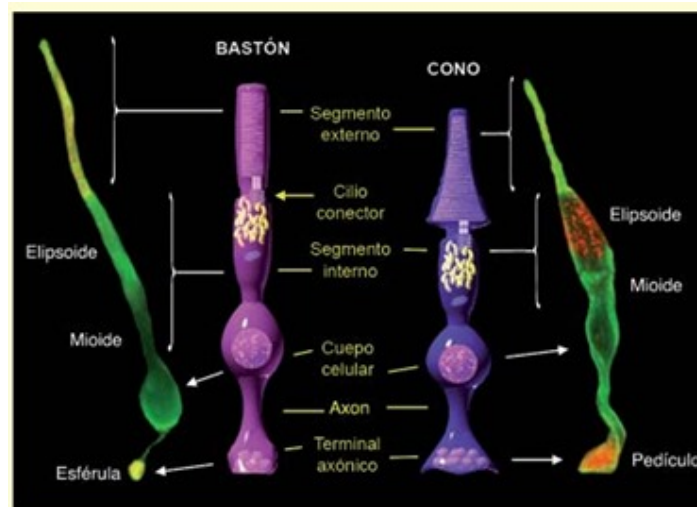
### 2.1 Retina y Visión

La visión es un proceso complejo en el que intervienen múltiples estructuras del ojo véanse en la Figura 2.1, Autor desconocido (s.f.), siendo la retina el tejido clave. Los fotorreceptores son células especializadas, ubicadas en la retina, están formadas por un segmento sináptico, un cuerpo celular, un segmento interno y un segmento externo como se puede observar en la Figura 2.2, Lledó Riquelme y Cuenca Navarro (2010). En este último, se encuentran proteínas esenciales para la fototransducción, como la opsina, que se organiza en una estructura de discos apilados.



**Figura 2.1:** Globo ocular en el que están representado las capas que lo componen: Túnica externa, intermedia y retina. Anatomía del ojo. Recuperado el 25 de abril de 2025 Autor desconocido (s.f.).

El segmento externo de los fotorreceptores está en constante renovación para garantizar su

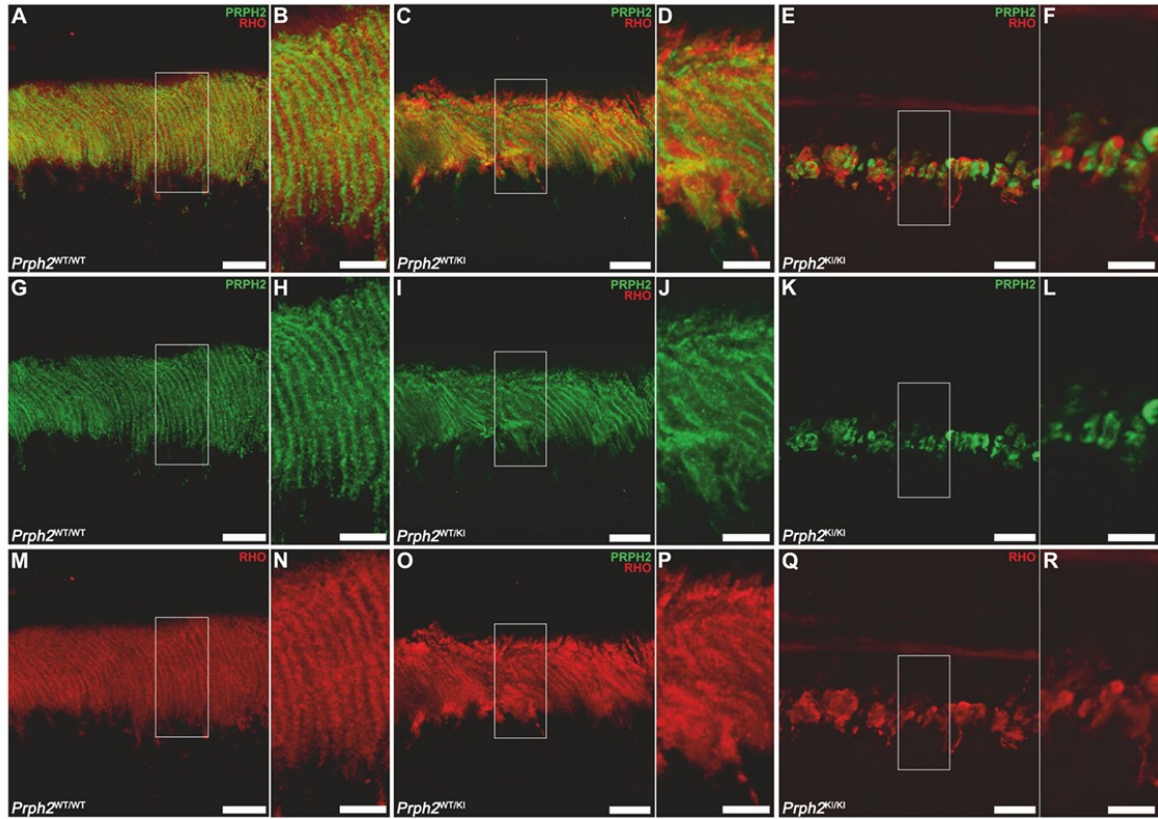


**Figura 2.2:** : Características morfológicas de los receptores Lledó Riquelme y Cuenca Navarro (2010)

correcto funcionamiento. Este proceso es llevado a cabo por el epitelio pigmentario de la retina (RPE, por sus siglas en inglés), con funciones tan importantes como eliminar los discos más antiguos, que contienen opsinas degradadas y segregar factores esenciales para mantener la integridad estructural de la retina Thebault (2011). Como se puede observar en la Figura 2.3, Ruiz-Pastor y cols. (2023), una mutación en el gen de la periferina (PRPH2), provoca la incorrecta regeneración de los segmentos externos de los fotorreceptores que tiene como consecuencia el desarrollo de enfermedades neurodegenerativas que puedan derivar en ceguera.

## 2.2 Mutación en el gen de la Periferina

El gen PRPH2 codifica la periferina-2, una proteína esencial en la formación y estabilidad de los discos membranosos de los segmentos externos de los fotorreceptores en la retina. Las mutaciones en PRPH2 se han asociado con diversas distrofias retinianas hereditarias, como la retinosis pigmentaria y la degeneración macular Peeters y cols. (2021). En este caso, se descubrió que la distrofia corioidea areolar central (CACD) también estaba formada por al menos cinco mutaciones diferentes en este gen, y que, por tanto, presentaba una alta heterogeneidad genética Ruiz-Pastor y cols. (2023). Finalmente, se generó un modelo murino con la mutación p.Arg195Leu (c.584G>T) en PRPH2, utilizando la tecnología CRISPR/Cas9.



**Figura 2.3:** Segmento externo alterado en la retina del modelo C57BL/6J-Prph<sup>em1sal</sup>. Tanto el ratón con la mutación en homocigosis (derecha, Prph2<sup>KI/KI</sup>) como en heterocigosis (centro, Prph2<sup>WT/KI</sup>), muestran un desarrollo anormal de los segmentos externos de los fotorreceptores si se comparan con la retina de los ratones sanos (izquierda, Prph2<sup>WT/WT</sup>) Ruiz-Pastor y cols. (2023).

Esta mutación implica la sustitución del aminoácido arginina (Arg) por leucina (Leu) en la posición 195 de la proteína, lo que podría afectar su función en la homeostasis y organización de los segmentos externos de los fotorreceptores. Para introducir esta mutación, se diseñó un oligonucleótido de ADN monocatenario (ssODN) de 200 nucleótidos como plantilla para la recombinación homóloga, junto con un sgRNA dirigido a la región intrón 1-exón 2 del gen Prph2. La edición genética se realizó en embriones de ratón C57BL/6J, y los animales resultantes fueron cruzados para obtener ratones heterocigotos (Prph2<sup>WT/KI</sup>) y homocigotos (Prph2<sup>KI/KI</sup>) Ruiz-Pastor y cols. (2023). El análisis genético confirmó la herencia mendeliana de la mutación, y la evaluación funcional de la retina mediante pruebas optomotoras (OPT), electroretinografía (ERG) y tomografía de coherencia óptica (OCT) Rösch y cols. (2014) permitió caracterizar su impacto en la visión y la estructura retiniana a lo largo del

envejecimiento. Este modelo murino representa una herramienta fundamental para estudiar los mecanismos patogénicos de las mutaciones en PRPH2, así como para el desarrollo de potenciales estrategias terapéuticas en enfermedades degenerativas de la retina.

## 2.3 Pruebas para la degeneración de la retina

Para determinar el estado de degeneración de la retina en ratones con la mutación se emplean diversas técnicas experimentales. Estas pruebas nos permiten evaluar tanto la estructura morfológica de la retina y nervio óptico como la función visual.

### 2.3.1 Prueba Optomotora

La prueba optomotora (OPT) es un método no invasivo utilizado para evaluar la agudeza visual (VA) en ratones García y cols. (2023). Se basa en la respuesta del animal ante un estímulo visual en movimiento. Para ello, se usa el sistema optomotor Argos (Instead Technologies) (Figura 2.4), el cual proyecta un patrón de rejilla vertical en rotación horizontal durante 5 segundos. Un observador entrenado analiza si los movimientos de la cabeza del ratón van en la misma dirección que el estímulo visual o si por el contrario el ratón no hace interacción con el medio.



**Figura 2.4:** A la izquierda Argos: Un sistema para evaluar la función visual en pequeños animales de investigación, fundamental en estudios preclínicos de enfermedades neurodegenerativas. A la derecha dibujo de su funcionamiento. Modificado a partir de Instead Technologies (s.f.) y Ruiz-Pastor y cols. (2023).

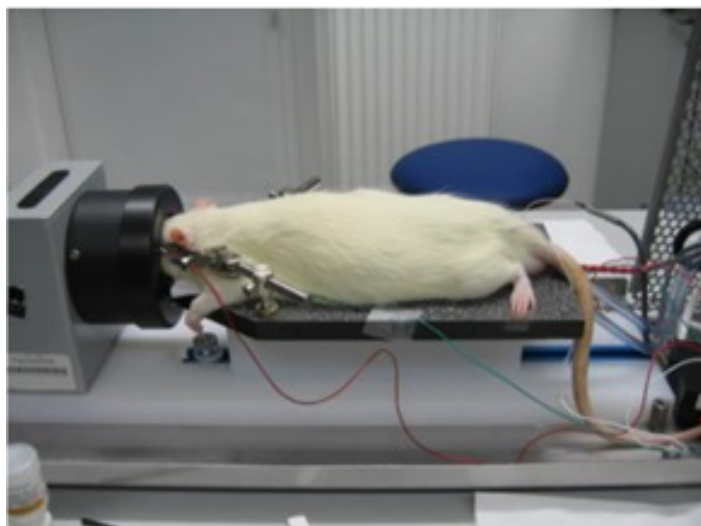
### 2.3.2 Electroretinografía

La electroretinografía (ERG) permite medir la respuesta eléctrica de la retina a estímulos luminosos, proporcionando información clave sobre la función de los fotorreceptores y otras células retinianas López y Docampo (1964).

En la prueba escotópica se somete al ratón a condiciones de oscuridad total tras una noche. Para facilitar la captación de señales, se dilatan las pupilas con tropicamida al 1% y se aplica un gel para evitar la deshidratación ocular Ruiz-Pastor y cols. (2023).

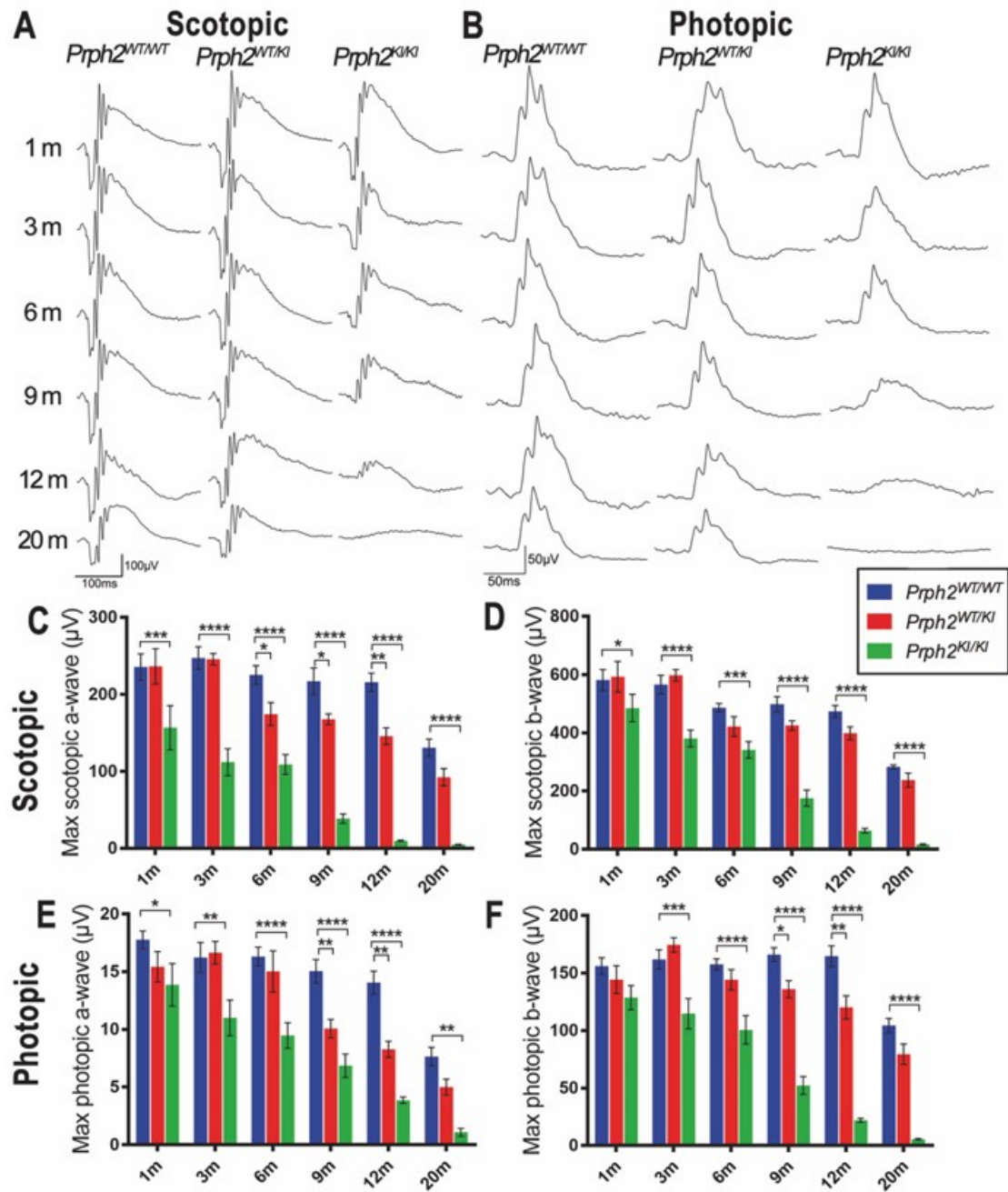
A continuación, se emplea el sistema HM<sub>s</sub>ERG LAB (Figura 2.5) OcuScience (s.f.), con los ratones anestesiados y mantenidos a una temperatura constante de 38 °C. Se les somete a catorce estímulos de luminancia creciente (de  $-5.5$  a  $1 \log \text{cd} \cdot \text{s}/\text{m}^2$ ) para evaluar las respuestas escotópicas.

Posteriormente, tras un período de adaptación a la luz, se registran respuestas fotópicas mediante ocho estímulos adicionales (de  $-2.5$  a  $0.9 \log \text{cd} \cdot \text{s}/\text{m}^2$ ).



**Figura 2.5:** Dispositivo OcuScience® HM<sub>s</sub>ERG utilizado para la evaluación electrofisiológica de la visión en ratones OcuScience (s.f.)

Una vez hecho el proceso experimental, se analizan las amplitudes de las ondas ERG



**Figura 2.6:** Disminución dependiente de la edad de la respuesta retiniana en los ratones doble mutantes ( $Prph2^{KI/KI}$ ) y heterocigotos para la mutación ( $Prph2^{WT/KI}$ ), comparado con los ratones sanos ( $Prph2^{WT/WT}$ ). Modificado a partir de Ruiz-Pastor y cols. (2023)

- La **onda A** representa la respuesta de los fotorreceptores. Se mide desde la línea base hasta el punto más negativo del registro.



- La **onda B** representa la respuesta de las células bipolares y de Müller. Se mide desde el punto más bajo de la onda A hasta el pico de la onda B.

Tal y como se puede observar en la Figura 2.6, Ruiz-Pastor y cols. (2023), la disminución progresiva de la amplitud de estas ondas indica la degeneración funcional de la retina Ruiz-Pastor y cols. (2023).

### 2.3.3 Tomografía de coherencia Óptica

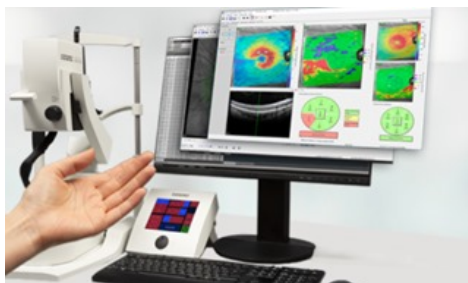
La tomografía de coherencia óptica (del inglés, *Optical Coherence Tomography*, OCT) es una técnica de imagen no invasiva que permite detectar cambios estructurales en la retina a lo largo del tiempo, lo que facilita el seguimiento de la degeneración retiniana en los modelos de estudio.

En primer lugar, se anestesia al animal, se le dilatan las pupilas y se aplica de nuevo un suero fisiológico para evitar la deshidratación de la córnea Ruiz-Pastor y cols. (2023).

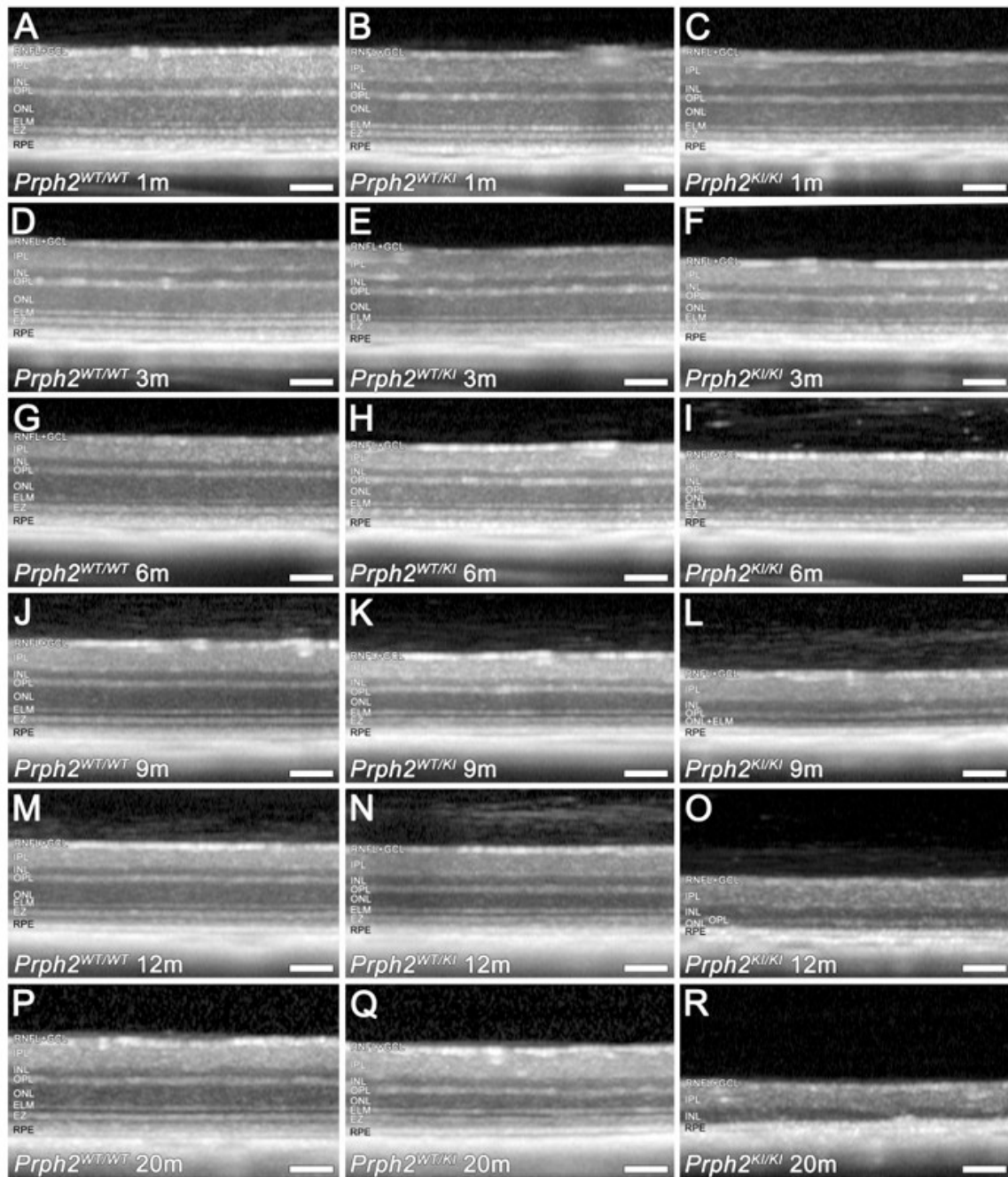
Gracias al sistema Spectralis OCT (Heidelberg Engineering), que se puede ver en la Figura 2.7 Heidelberg Engineering (s.f.), se mide el grosor de la retina total y de capas específicas (véase Figura 2.8 Ruiz-Pastor y cols. (2023)), tales como:

- Epitelio pigmentario de la retina (del inglés, *Retinal Pigment Epithelium*, RPE).
- Capa nuclear externa (del inglés, *Outer Nuclear Layer*, ONL).
- Capa nuclear interna (del inglés, *Inner Nuclear Layer*, INL).

Todo ello permite la obtención de imágenes de alta resolución del nervio óptico en distintos momentos del estudio.



**Figura 2.7:** Plataforma de imagen SPECTRALIS® OCT: combinación de imagen de fondo con láser de escaneo y OCT simultáneo Heidelberg Engineering (s.f.)



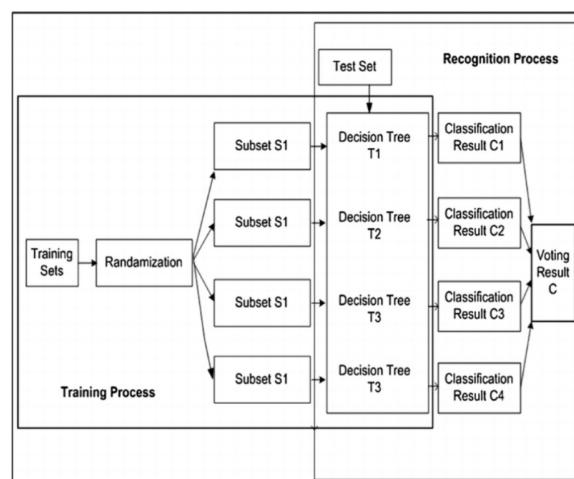
**Figura 2.8:** Alteraciones estructurales en las retinas de ratones mutantes *Prph2* evaluadas *in vivo* mediante tomografía de coherencia óptica (OCT) a lo largo de la edad. Nótese cómo disminuye el espesor de la retina total y de las diferentes capas en los ratones doble mutantes (derecha, *Prph2*<sup>KI/KI</sup>) y heterocigotos para la mutación (centro, *Prph2*<sup>WT/KI</sup>), en comparación con los ratones sanos (izquierda, *Prph2*<sup>WT/WT</sup>). Ruiz-Pastor y cols. (2023)

## 2.4 Arquitecturas de aprendizaje automático

### 2.4.1 Árboles de Decisión (del inglés, *Random Forest*)

El *Random Forest* fue propuesto por Leo Breiman en 2001. Es un método de clasificación basado en un conjunto de árboles de decisión independientes. Cada uno de estos árboles aprende de forma diferente, ya que se entrena con partes distintas de los datos y con diferentes características.

Cuando el modelo tiene que clasificar, cada árbol emite un voto, y la clase final se decide por mayoría. En principio la tasa de clasificación es mejor que la de clasificadores individuales como máquinas de vectores de soporte. A continuación, en la Figura 2.9 Hemanth y cols. (s.f.), se puede ver un esquema de su funcionamiento.



**Figura 2.9:** Marco conceptual del clasificador Random Forest Hemanth y cols. (s.f.)

### 2.4.2 XGBoost

*XGBoost*, desarrollado por Tianqi Chen en la Universidad de Washington IBM (s.f.), es un algoritmo de *boosting* que mejora el rendimiento combinando árboles de decisión débiles en árboles más fuertes mediante la suma de residuos. Se representa esquemáticamente en la Figura 2.10 IBM (s.f.).

A diferencia de *Random Forest*, los árboles de XGBoost se entrenan de forma secuencial, donde cada árbol trata de corregir los errores del anterior Chen y Guestrin (2016). Utiliza

descenso de gradiente para minimizar la función de pérdida durante el entrenamiento.

### Principales hiperparámetros de XGBoost

- **Tasa de aprendizaje:** controla la velocidad con la que el modelo aprende en cada iteración. Un valor bajo reduce la contribución de cada árbol, ayudando a prevenir el sobreajuste.
- **n\_estimators:** especifica el número de árboles que se entrenarán en el modelo. Aumentarlo puede hacer el modelo más complejo y propenso al sobreajuste.
- **Gamma (multiplicador de Lagrange):** controla la reducción mínima de error necesaria para dividir un nodo. Un valor alto puede hacer que el modelo sea más conservador y evite el sobreajuste, mientras que uno bajo permite más divisiones, lo que podría aumentar la complejidad.
- **max\_depth:** establece la profundidad máxima de cada árbol. Un valor más alto hace el modelo más complejo y aumenta el riesgo de sobreajuste.

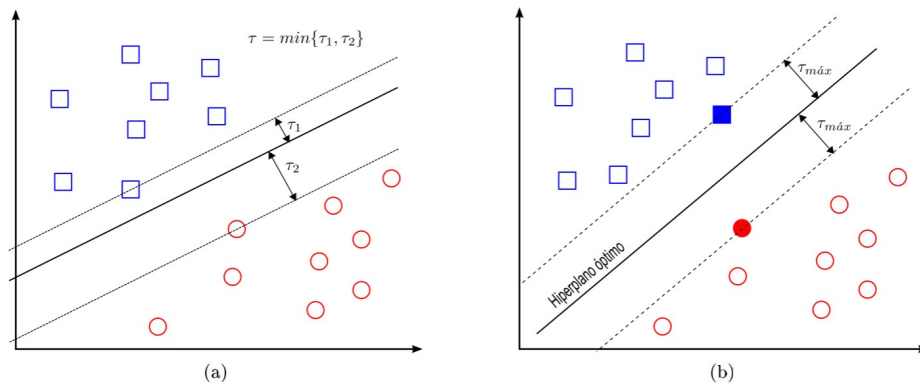


Figura 2.10: Marco conceptual de XGBoost IBM (s.f.)

### 2.4.3 Máquinas de Vectores de Soporte

Las máquinas de vectores de soporte (del inglés, *Support Vector Machine*, SVM) fueron introducidas en los años 90 por Vapnik y sus colaboradores: Boser et al. (1992) y Cortes & Vapnik (1995) Carmona (2016). Inicialmente fueron diseñadas para resolver problemas de clasificación binaria.

Es un modelo de clasificación que busca encontrar la mejor línea o frontera que separe los datos en diferentes clases. Esta frontera se denomina *hiperplano*, y debe estar lo más alejada posible de los puntos de cada clase, como se puede observar en (b) de la Figura 2.11 Carmona (2016).



**Figura 2.11:** Margen de un hiperplano de separación: (a) hiperplano de separación no-óptimo y su margen asociado (no máximo) (b) hiperplano de separación óptimo y su margen asociado máximo. Carmona (2016)

### 2.4.4 Naive Bayes

Naive Bayes es un modelo que se basa en el Teorema de Bayes (representado matemáticamente en la fórmula adjunta), el cual permite calcular la probabilidad posterior de un evento dado un conocimiento previo. Este método “ingenuamente” asume la independencia de los atributos dada la clase Hernández y cols. (2004).

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)} \quad (2.1)$$

Dependiendo del tipo de datos, se usan diferentes variantes:

- **GaussianNB:** es el que usaremos, se usa para datos continuos, asumiendo una distri-

bución normal.

- **MultinomialNB**: para datos de conteo, como frecuencia de palabras.
- **BernoulliNB**: para variables binarias.

### 2.4.5 Regresión Logística (en inglés *Logistic Regression*)

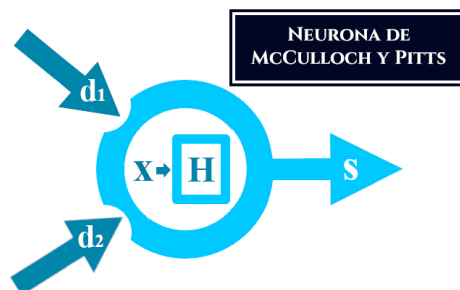
Es un modelo estadístico que se usa principalmente para la clasificación binaria. Su origen se remonta a la década de los sesenta (Confield, Gordon y Smith 1961); su uso se universaliza y expande desde principios de los ochenta Fiuza Pérez y Rodríguez Pérez (2000).

El modelo se basa en la función sigmoide, que transforma una combinación lineal de las variables independientes en una probabilidad entre 0 y 1, y durante su entrenamiento se ajustan los coeficientes para minimizar la función de pérdida logarítmica.

## 2.5 Redes neuronales: origen, estructura y arquitecturas utilizadas

### 2.5.1 Origen histórico y modelo de McCulloch-Pitts

Las redes neuronales tienen su origen en 1943, de la mano del psiquiatra neuroanatomista McCulloch y el matemático Pitts, quienes crearon por primera vez un modelo de red neuronal Izaurieta y Saavedra (s.f.). En su modelo, había canales de entrada en cada neurona, denominados dendritas, y un único canal de salida llamado axón. En la Figura 2.12 La Máquina Oráculo (s.f.) se puede observar una representación esquemática.



**Figura 2.12:** Esquema de una neurona de McCulloch-Pitts, con dos dendritas y un axónLa Máquina Oráculo (s.f.)

Su modelo matemático dependía de dos operaciones matemáticas que operan de forma consecutiva La Máquina Oráculo (s.f.).

**Función de suma:**

simplemente sumaba las entradas recibidas:

$$x = \sum c_i d_i \quad (2.2)$$

Donde:

- $c_i$ : entradas
- $d_i$ : peso asociado a cada entrada

**Función escalón:**

Es una función de activación, y, por lo tanto, se encarga de determinar si la neurona se activa o no. Esto depende del valor de  $U$  (el umbral de acción, como en una neurona real), de  $x$  (el número de excitaciones recibidas), y de  $H$ , que es la función escalón de Heaviside.

- Si  $x < U$ , entonces  $H = 0$  (la neurona no se activa).
- Si  $x \geq U$ , entonces  $H = 1$  (la neurona se activa).

$$s = H(x - U) \quad (2.3)$$

**2.5.2 Estructura funcional de las redes neuronales****Capas (entrada, ocultas, salida)**

Los modelos actuales requieren muchas neuronas, por lo que se agrupan aquellas que presentan comportamientos similares en capas. En la Figura 2.13 Izaurieta y Saavedra (s.f.) se pueden ver bien diferenciadas, siendo cada capa un vector de neuronas.

La red multicapa consta de tres tipos diferentes de capas:

---

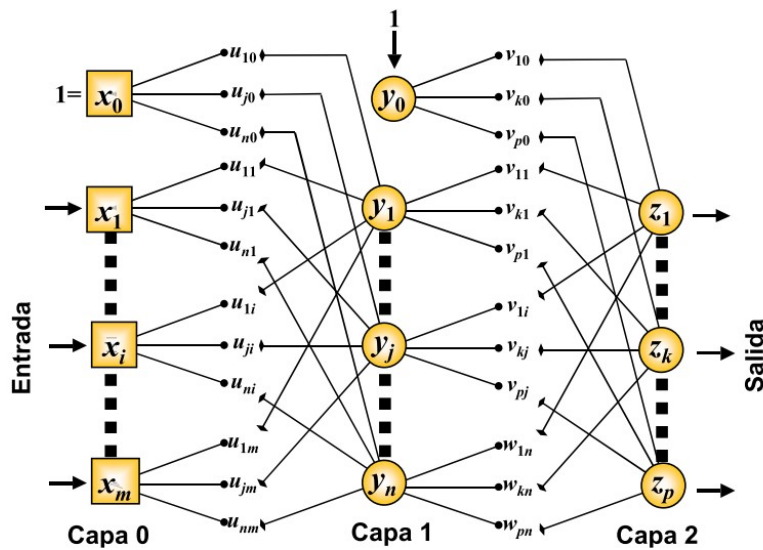


Figura 2.13: Red multicapa Izaurieta y Saavedra (s.f.)

- **Capa de entrada:** la que recibe los datos.
- **Capas ocultas:** procesan internamente la información. La parte clave es que aquí se aplican funciones no lineales. Si se usaran solo operaciones como multiplicar por una matriz o sumar un sesgo, el resultado final seguiría siendo una función lineal del input Izaurieta y Saavedra (s.f.).

Cuando se usan funciones no lineales como ReLU, sigmoide, tanh, etc., la red puede aprender relaciones complejas entre los datos.

- **Capa de salida:** genera el resultado final.

En todas las capas se aplican diferentes tipos de funciones, y en este trabajo se emplearán las dos siguientes:

### Función ReLU (del inglés, *Rectified Linear Unit*)

Es una función de activación introducida por Hahnoser et al. en los 2000 Hahnloser y cols. (2000), muy utilizada por su fácil implementación. Su definición es:

$$f(x) = (0, x) \quad (2.4)$$



Simplemente devuelve el valor de entrada si es positivo y 0 si es negativo Krishnamurthy (2024). La principal desventaja es que si durante el entrenamiento una neurona recibe valores negativos constantes, estos se convertirán en 0 y, por tanto, la neurona podría llegar a “morir”.

### Función Sigmoide

Es una función que se suele emplear en la capa de salida, y comúnmente en problemas de clasificación binaria. Su nombre proviene de su curva con forma sigmoidea, lo que facilita la captura de relaciones no lineales, se expresa matemáticamente como:

$$f(x) = \frac{1}{1 + e^{-x}} \quad (2.5)$$

Esta función transforma cualquier valor real en un rango entre 0 y 1 Tuzsuz (2022).

Es importante destacar que cada neurona realiza estas operaciones individualmente. Todas las neuronas de una capa suelen usar la misma función de activación, pero cada una tiene sus propios pesos.

### Operaciones globales

Además de las operaciones locales en cada neurona, existen operaciones que afectan globalmente a toda la red:

- **Funciones de pérdida (del inglés, *loss functions*):** cuantifican el error entre la salida predicha y la salida esperada, es decir, miden qué tan mal lo hizo la red.
  - **Detención temprana:** estrategia utilizada para prevenir el sobreajuste durante el entrenamiento de modelos. Detiene el entrenamiento cuando el rendimiento en los datos de validación deja de mejorar durante varias épocas seguidas (período de “paciencia”) GeeksforGeeks (s.f.).
  - **Algoritmos de optimización:** como *Stochastic Gradient Descent* (SGD) o Adam, que tratan de minimizar el error ajustando los pesos de la red para darle más importancia a unas neuronas que a otras.
-

- **Retropropagación (backpropagation):** proceso matemático que permite que la red aprenda de sus errores. El error calculado se envía hacia atrás desde la salida a la entrada y se aplica la regla de la cadena (calculando derivadas) para obtener los gradientes de la función de coste Analytics Vidhya (2023). Con estos gradientes, se ajustan los pesos de las conexiones entre las neuronas para mejorar la predicción. En la Figura 2.14 Analytics Vidhya (2023) se presenta un esquema de su funcionamiento.

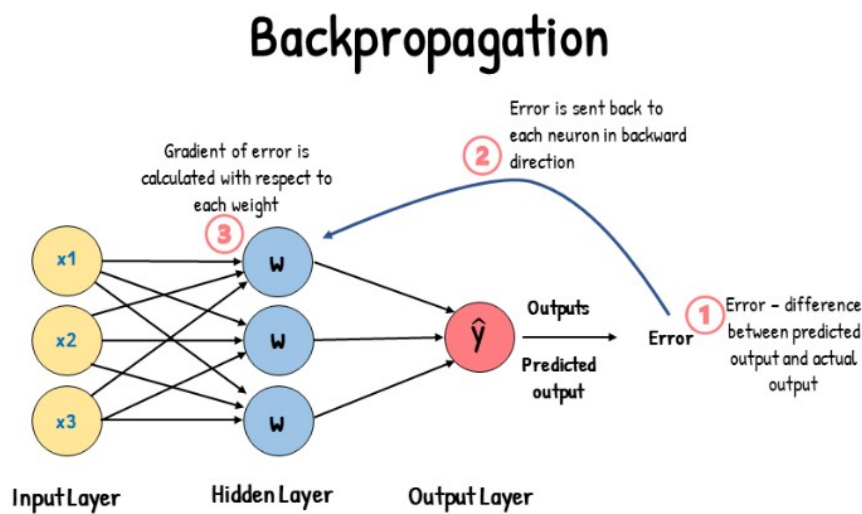


Figura 2.14: Esquema del proceso de retropropagación Analytics Vidhya (2023)

En este trabajo se emplearán dos de las arquitecturas de las redes neuronales más conocidas.

### 2.5.3 Arquitecturas utilizadas: DNN y Perceptrón Multicapa

#### Red Neuronal Profunda — *Deep Neural Network* (DNN)

Una Red Neuronal Profunda (DNN) es un tipo de red neuronal caracterizada por la presencia de múltiples capas ocultas. A diferencia de otras redes, las DNN no necesariamente requieren conexiones densas entre sus neuronas, lo que las hace más flexibles y adecuadas para diversas aplicaciones. En general, las DNN tienen una estructura más compleja en cuanto al número de capas y neuronas, lo que les permite modelar representaciones de datos más abstractas y complejas.

En este trabajo, se utiliza la arquitectura de DNN para compararla con otras estructuras

como el *Multilayer Perceptron* (MLP). El objetivo es analizar cómo influyen el número de capas en el rendimiento global del modelo.

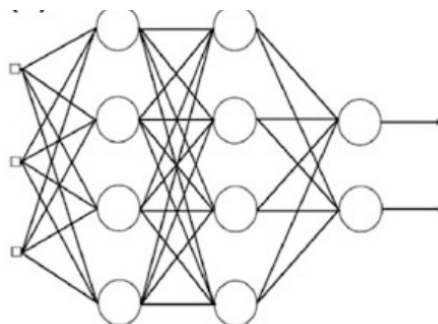
Las redes neuronales profundas pueden incorporar capas especializadas, como capas de convolución en el caso de tareas de procesamiento de imágenes, o redes recurrentes cuando se abordan secuencias temporales. Esta versatilidad en su diseño permite que las DNN sean aplicables a una amplia gama de problemas complejos.

### Perceptrón Multicapa — *Multilayer Perceptron* (MLP)

El Perceptrón Multicapa es un tipo de red neuronal que cuenta al menos con tres capas. Las conexiones entre las neuronas de diferentes capas son densas, es decir, cada neurona está conectada con todas las neuronas de la siguiente capa, como se puede observar en la Figura 2.15 Taud y Mas (2018).

Además, es una arquitectura ideal para situaciones en las que no se requiere tratamiento de datos con estructura espacial o temporal (como imágenes, secuencias, etc.).

Se ha considerado evaluar esta arquitectura con el fin de analizar el impacto de la conectividad de la red en el rendimiento del modelo, permitiendo comparar modelos en los cuales sus neuronas están más conectadas con aquellos que tienen mayor cantidad de neuronas o capas.



**Figura 2.15:** Ejemplo de arquitectura MLP Taud y Mas (2018)

## 2.6 Técnicas de aumento de datos

### 2.6.1 Perturbación Aleatoria (*Random Noise*)

Consiste en agregar una pequeña cantidad de ruido aleatorio a las características numéricas de los datos, y puede ser útil para introducir variabilidad y robustez en modelos entrenados con bases de datos pequeñas. Esta técnica se apoya en fundamentos teóricos como los establecidos por Rice (1944), quien desarrolló un análisis matemático del ruido aleatorio como un proceso estocástico Rice (1944).

La fórmula general es:

$$x' = x + e \quad (2.6)$$

Donde:

- $\mathbf{x}$ : valor original de la característica *feature*
- $\mathbf{e}$ : ruido aleatorio, normalmente generado de una distribución  $N(0, \sigma^2)$
- $\mathbf{x}'$ : nuevo valor perturbado

### 2.6.2 Escalado (*Scaling*)

En este trabajo se ha utilizado una técnica de *Data Augmentation* consistente en aplicar un vector de escalas al vector de características original. Cada característica individual se multiplica por un factor de escala diferente, elegido aleatoriamente dentro de un rango cercano a 1 (por ejemplo, entre 0.9 y 1.1). Esto permite simular pequeñas variaciones realistas en los datos, aumentando la diversidad del conjunto de entrenamiento y reduciendo el riesgo de sobreajuste Shorten y Khoshgoftaar (2019). Matemáticamente:

$$x' = Sx = (s_1 \cdot x_1, s_2 \cdot x_2, \dots, s_n \cdot x_n) \quad (2.7)$$

Donde:

- $\mathbf{s} = (s_1, s_2, \dots, s_n)$ : vector de factores de escala
-

- $\mathbf{x} = (x_1, x_2, \dots, x_n)$ : vector de características
- $\mathbf{si} > 0$ : para todos los  $i$
- $\mathbf{x}'$ : es el nuevo vector de características escalado

### 2.6.3 Generación de Muestras Sintéticas (del inglés, *Synthetic Minority Over-sampling Technique, SMOTE*)

SMOTE es una técnica propuesta por Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall y W. Philip Kegelmeyer en 2002 Chawla y cols. (2002), que genera muestras sintéticas a partir de las muestras existentes, interpolando entre ejemplos de la clase minoritaria en tareas de clasificación desbalanceada.

Es útil en clasificación cuando hay un desbalance entre las clases y se quiere mejorar el rendimiento del modelo al crear más ejemplos de la clase minoritaria, porque tiene como objetivo expandir la región de decisión de la clase minoritaria Chawla y cols. (2002).

De esta manera se generan nuevas instancias:

$$x_{\text{nuevo}} = x_i + \lambda \cdot (x_{\text{vecino}} - x_i) \quad (2.8)$$

Donde:

- $x_i$ : punto de la clase minoritaria
- $x_{\text{vecino}}$ : vecino cercano del mismo grupo
- $\lambda \sim \mathcal{U}(0, 1)$ : valor aleatorio entre 0 y 1
- $x_{\text{nuevo}}$ : nuevo punto generado

### 2.6.4 *Bootstrap Resampling*

El método de *bootstrap* fue propuesto por Bradley Efron en 1979 Efron (1979). Consiste en crear nuevos conjuntos de datos de entrenamiento mediante el re-muestreo con reemplazo de las muestras originales.

---

Es útil para crear múltiples subconjuntos de datos de manera aleatoria, y es un método muy común en modelos de ensamblado, como *Random Forest* y *Bagging*.

La fórmula general es la siguiente:

$$D^* = \{x'_1, x'_2, \dots, x'_n\} \tag{2.9}$$

Donde:

- $D = \{x_1, x_2, \dots, x_n\}$ : conjunto de datos original
- $D^*$ : nuevo conjunto de datos, normalmente su tamaño es igual al de  $D$

## 3 Estado del Arte

Esta sección se centra en la exploración detallada de los trabajos más relacionados y vanguardistas en el ámbito de la clasificación de la degeneración de la retina a través del uso de técnicas de DL o ML. El objetivo es proporcionar una descripción integral y precisa de cada estudio relevante, destacando las contribuciones distintivas y significativas que cada uno ha aportado al campo de estudio. Esta revisión proporcionará una base sólida para el entendimiento de la evolución del campo, las estrategias y tecnologías que definen el estado del arte en la clasificación de la degeneración retiniana. La Tabla 3.1 muestra un resumen con los aspectos más relevantes de cada trabajo.

Estudio	Mares y cols. (2024)	Akpınar y cols. (2024)	Balyen y Peto (2019)	García y cols. (2023)
<b>Modelo final usado (ganador)</b>				
	Convolutional Neural Networks (CNN) entrenadas en PIN-NACLE y HARBOR	Sequential Minimal Optimization (SMO) para AMD; también destacado uso de ResNet101	Deep Convolutional Neural Network (DCNN) según Ting et al. (2017)	PointNet++ (mejor rendimiento frente a DGCNN)
<b>¿Qué conclusiones?</b>	OCT combinada con IA permite un diagnóstico más preciso, eficiente y temprano	La IA tiene un alto potencial diagnóstico; sin embargo, hay limitaciones por falta de datos	Las redes neuronales profundas (DCNN) pueden superar el rendimiento clínico humano en tareas específicas	El enfoque con Geometric Deep Learning mostró alta precisión y eficiencia
<b>¿Con qué resultados?</b>	AUC = 0.94 en multiclass; AUC GA = 0.80, AUC nAMD = 0.68	AUC de hasta 0.984 (AMD con SMO); precisión máxima reportada: 98.8%	DR: AUC 0.936; AMD: AUC 0.931; Glaucoma: AUC 0.942 (Ting et al.)	F1-score: Parkinson 0.97; Esclerosis Múltiple 0.94; clasificación multi-clase sin temblor
<b>¿Qué tipos de enfermedad?</b>	Degeneración macular asociada a la edad (AMD), incluyendo formas geográficas y neovasculares	Glaucoma, AMD, DR, edema macular diabético, membrana epirretiniana, agujero macular	Retinopatía diabética (DR), degeneración macular (AMD), glaucoma	Alzheimer, Parkinson, Esclerosis Múltiple, Temblor Esencial
<b>¿Cuántas clases?</b>	4 clases (normal, AMD temprana, intermedia, avanzada)	Variable (de 2 hasta múltiples clases según estudio)	Binario o multiclase (ej. referable/no referable o niveles de severidad)	5 clases inicialmente (Sanos, Alzheimer, Parkinson, Esclerosis, Temblor esencial)
<b>¿Qué clasifican?</b>	Estadios de AMD (temprana, intermedia, avanzada) y actividad de la OCT	Clasificación de múltiples enfermedades oculares mediante imágenes OCT	Clasificación de DR, AMD y glaucoma en imágenes de fondo de ojo (citando resultados de Ting et al.)	Clasificación automatizada de enfermedades neurodegenerativas a partir de imágenes

**Tabla 3.1:** Resumen comparativo de estudios sobre clasificación de enfermedades mediante inteligencia artificial aplicada a imágenes OCT y de fondo de ojo.



## 4 Herramientas utilizadas

El éxito de este proyecto dependerá del uso adecuado de varios recursos humanos y materiales. Estos son fundamentales en cada una de las etapas del proyecto, y su correcta utilización permitirá alcanzar los objetivos establecidos. A continuación, se describirán estos recursos y se explicará cómo influyen en el desarrollo del proyecto.

### 4.1 Hardware

Mi ordenador personal, *VivoBook\_ASUSLaptop X421FL\_S433FL*, ha sido la herramienta principal para llevar a cabo diversas actividades relacionadas con el proyecto: investigar, redactar documentos, programar, entre otras funciones. Las características técnicas del equipo son las siguientes:

- **Procesador:** Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz, 2304 MHz, 4 procesadores principales, 8 procesadores lógicos
- **Memoria RAM:** 16GB
- **Sistema Operativo:** Microsoft Windows 11 Home
- **Tarjeta gráfica:** NVIDIA GeForce MX250

### 4.2 Software

- **Entornos y plataformas**
  1. **Google Drive:** es una herramienta de almacenamiento en línea que permite guardar, sincronizar y compartir archivos a través de la nube. Se ha usado para la

realización de este TFG con el fin de facilitar la gestión y el acceso a la base de datos, artículos científicos de interés, material, etc.

2. **Google Colab:** es una plataforma en la nube que permite ejecutar código Python de manera colaborativa. Ofrece recursos computacionales como GPUs y TPUs sin costo. Se ha seleccionado esta plataforma ya que, personalmente, me sentía más familiarizada con ella debido a proyectos previos realizados durante el grado.
3. **Visual Studio Code:** es un editor de código fuente gratuito y de código abierto. Ha sido muy útil para poder usar nuestro modelo en una aplicación a través de un script.

- **Lenguajes y plataformas de desarrollo**

1. **Python:** se ha seleccionado este lenguaje de programación para el desarrollo de este proyecto, debido a su amplia disponibilidad de bibliotecas especializadas en aprendizaje automático y análisis de datos.
2. **JavaScript:** se ha seleccionado este lenguaje de programación para desarrollar la aplicación web, dado que es un lenguaje con el que también estoy familiarizada debido a trabajos anteriores realizados durante el grado.

- **Frameworks de la IA**

1. **PyTorch:** es una biblioteca de código abierto para Python dedicada al aprendizaje automático. Se caracteriza por su alta flexibilidad y dinamismo, lo que la hace especialmente popular entre investigadores y desarrolladores que trabajan en áreas como el aprendizaje profundo y la inteligencia artificial.

- **Librerías principales:** Numpy, Pandas, Matplotlib

## 4.3 Recursos Humanos

Dado que este TFG es individual, el estudiante asume tres de los cuatro roles en el proyecto: programador, diseñador y analista. Los tutores, por su parte, actúan como jefes de proyecto.

En su rol de analista, diseñador y programador, el estudiante es responsable de definir los requisitos del sistema según las necesidades planteadas por los jefes de proyecto, además de

---

tomar las decisiones de diseño necesarias para su desarrollo, así como realizar la codificación y las pruebas correspondientes.

Los tutores brindarán orientación al estudiante y resolverán sus dudas a través de reuniones semanales, mientras supervisan el avance del proyecto y garantizan que se siga el cronograma establecido.

---



## 5 Metodología Aplicada y Resultados obtenidos

En este apartado se describe el origen de los datos utilizados y la metodología empleada para abordar el problema de detección de la degeneración retiniana.

### 5.1 Adquisición de datos

La base de datos utilizada en este TFG ha sido proporcionada por el Departamento de Fisiología, Genética y Microbiología de la Universidad de Alicante, específicamente por la profesora Natalia Martínez Gil, tutora de este trabajo.

Se trata de una base de datos almacenada en un archivo “.xlsx” que contiene un total de 258 instancias, cada una representando un ratón individual. La base de datos original se muestra en la Figura 5.1.

Registro Control WT Peripherin prph2										Escotópico				Fotópico				F. DICT				Data OCT				
Edad sacrificio	ID animal	Sexo	P30	3m	4m	6m	9m	12m	17-21m	Max a.w	CI	Max b.w	CI	Max a.w	CI	Max b.w	CI	F. Optomet	Edad Opt	Data Opt	F. DICT	CI	Max a.w	CI	Max b.w	CI
1	WT-25	♀	10/20/05							224.1	262.1	540.4	628.0	16.9	19.5	146.7	166.1	10/20/06	1.3m	0.572	10/20/06					
1	WT-81	♀																								
1	WT-82	♀																								
1	WT-83	♀																								
1	WT-84	♀																								
1	WT-85	♀																								
13	WT-23	♀	10/20/03							147.6	217.6	330.6	501.6	21.6	19.3	123.1	155.2	10/20/06	1.3m	0.574	10/20/06					
13	WT-24	♀	10/20/04							216.9	291.6	779.5	530.5	14.2	15.3	84.3	83.7	10/20/06	1.3m	0.594	10/20/06					
13	WT-26	♂	10/20/01							286.7	597.6	507.3	492.3	14.3	16.0	51.6	84.5	10/20/06	1.3m	0.590	10/20/06					
13	WT-27	♂	10/20/02							282.1	188.6	686.6	470.4	17.5	16.5	181.2	142.6	10/20/06	1.3m	0.616	10/20/06					
2	WT-86	♀																								
2	WT-89	♀																								
2	WT-90	♀																								
2	WT-91	♀																								
3	WT-1	♂	04/02/03							240.3	287.7	564.5	680.7	17	7	83.4	172.6	5/9/2006	3m	0.636	5/9/2006					
3	WT-2	♀	04/02/04							226	184.1	471.7	443.7	19	11	25.5	23.8	5/9/2006	3m	0.572	5/9/2006					
3	WT-3	♂	04/02/05							342*	292.9	772.6	559	26.1	21	212.5	173.7	5/9/2006	3m	0.572	5/9/2006					
3	WT-10	♀	03/03/02								183.1	-	467.2	-	15	-	131.4	4/9/2006	3m	0.550	4/9/2006					
3	WT-11	♀	03/03/03							246.8	203.4	598.7	493.3	14	14	82.3	147	4/9/2006	3m	0.594	4/9/2006					
3	WT-12	♀	03/03/04							189.1	215.4	472.3	546.5	7.2*	12	49.2*	149.5	4/9/2006	3m	0.636	4/9/2006					
3	WT-13	♀	03/03/05							329	245.7	734.3	550.4	16	14.3	200.9	175	4/9/2006	3m	0.616	4/9/2006					
3	WT-20	♀																								
3	WT-21	♀																								
3	WT-22	♀																								
3	WT-23	♀																								
3	WT-24	♀																								
3	sin numero (16)	♀																								
3	sin numero (17)	♀																								
3	sin numero (18)	♀																								

**Figura 5.1:** Imagen que muestra la tabla de Excel utilizada por el grupo de investigación como Base de datos original

Originalmente, la base de datos se componía de dos hojas de cálculo diferenciadas: la primera, denominada “WT”, contenía información relativa a ratones de tipo control, es decir,

sin mutación genética, los cuales no presentan degeneración a lo largo del tiempo. La segunda hoja, llamada “KIWT”, incluía datos correspondientes a ratones con una mutación que, en algún momento de su desarrollo, conduce a degeneración retinal.

En la hoja “WT”, cada ratón estaba descrito mediante un conjunto de 25 variables, entre las que destacan:

- **Edad de sacrificio:** edad (en meses) a la que el ratón fue sacrificado.
  - **ID animal:** identificador único asignado a cada ratón.
  - **Sexo:** sexo del ratón.
  - **P30, 3m, 4m, 6m, 9m, 12m, 17-21m:** nombres de los archivos correspondientes a los resultados de la prueba de electroretinografía (ERG) realizada a los 30 días, 3 meses, etc., respectivamente.
  - **Dato ERG:** valor de la respuesta ERG (medido en microvoltios,  $\mu\text{V}$ ).
  - **ERG Escotópica:**
    - **Max a-wave OD/OI:** valor máximo de la onda A en el ojo derecho (OD) e izquierdo (OI).
    - **Max b-wave OD/OI:** valor máximo de la onda B en OD y OI.
  - **ERG Fotópica:**
    - **Max a-wave OD/OI:** valor máximo de la onda A en el ojo derecho e izquierdo.
    - **Max b-wave OD/OI:** valor máximo de la onda B en ambos ojos.
  - **Fecha Optomotor y Edad Opto:** fecha y edad en meses del ratón al momento de realizar la prueba optomotora (OPT).
  - **Dato OPT:** resultado de la prueba OPT.
  - **Fecha OCT y Edad OCT:** fecha y edad del ratón durante la prueba OCT.
  - **Dato OCT:** valor obtenido en la prueba OCT (medido en micras,  $\mu\text{m}$ ).
  - **ONL y Retina total:** medidas morfológicas de la retina del ratón.
-

Por otro lado, la hoja "KIWT" también incluía 25 variables, aunque presentaba ligeras diferencias respecto a la hoja anterior. En particular:

- **Fecha de sacrificio:** variable adicional que registraba la fecha exacta en que el ratón fue sacrificado.
- **4m:** la variable correspondiente a la ERG a los 4 meses no estaba presente en este caso.

Del total de 258 instancias registradas en ambas hojas, se identificó que 152 de ellas no contenían datos relevantes en las variables asociadas a las pruebas, por lo que fueron descartadas. Además, posteriormente se llevó a cabo un preprocesamiento explicado en detalle más adelante y finalmente, se trabajó con un conjunto reducido de 104 instancias.

Este conjunto de datos presenta un claro enfoque de problema de clasificación, aunque cabe destacar que no estaba originalmente preparado para su uso en tareas de análisis de datos. La ausencia de una variable objetivo explícita, como por ejemplo un campo que indicara si el ratón degeneraba o no, y la inconsistencia en los valores recogidos, han exigido un exhaustivo trabajo de limpieza y preparación previa antes de abordar cualquier modelo de aprendizaje automático.

## 5.2 Preprocesamiento de los datos

En este apartado se detallan las decisiones tomadas durante la preparación de los datos para su posterior entrenamiento con un modelo, explicando el razonamiento detrás de cada una de ellas.

### 5.2.1 Edición de variables desde Excel

El primer paso consistió en unificar las dos hojas de datos en una sola, con el objetivo de agrupar todas las instancias en un único conjunto. Sin embargo, al examinar el resultado, se detectaron inconsistencias en la organización de los datos, con valores ubicados en columnas incorrectas. Para corregir estos problemas, se realizaron las siguientes modificaciones en la hoja "KIWT" directamente desde Excel:

- Eliminación de la variable **"F.de Sacrificio"**

- Creción de la variable "4m"

### 5.2.2 Preparación de la base de datos

Una vez asegurada la coherencia entre las variables en todas las instancias, se procedió a consolidar los datos en una única hoja. Posteriormente, se añadió una nueva variable denominada "degenera", con los siguientes valores:

- 0: No degenera (correspondiente a los casos de la primera hoja).
- 1: Degenera (correspondiente a los casos de la segunda hoja).

Se identificó la duplicación de las variables ya que algunas medidas se realizan en ambos ojos (ojo derecho – OD; y ojo izquierdo – OI) por lo que fueron renombradas siguiendo una nomenclatura específica para evitar ambigüedades. Ahora, cada variable incluye un número que indica su correspondencia con una medición específica en la electroretinografía (ERG):

- **OI\_1 y OD\_1:** Representan el valor máximo medido en el ojo izquierdo y derecho, respectivamente, de la onda A en la ERG escotópica.
- **OI\_2 y OD\_2:** Corresponden al valor máximo de la onda B en la ERG escotópica para cada ojo.
- **OI\_3 y OD\_3:** Indican el valor máximo de la onda A en la ERG fotópica en el ojo izquierdo y derecho.
- **OI\_4 y OD\_4:** Representan el valor máximo de la onda B en la ERG fotópica en cada ojo.
- **OI\_5:** hace referencia a los datos en la OCT de la capa nuclear externa de la retina.
- **OD\_5:** Datos de la OCT de la retina entera.

Este cambio permite una mejor organización de los datos y facilita su interpretación en el análisis del estudio.

A continuación, se realizaron los siguientes pasos de limpieza:

---



1. Eliminación de las instancias en las que todas las columnas OI, OD y dato OPT contenían valores NaN. Sin embargo, si al menos una de estas columnas tenía un valor, la instancia se mantenía en la base de datos.
2. Eliminación de las variables Edad sacrificio, ID animal, F. Optomotor y F. OCT, ya que no resultaban relevantes para determinar si un ratón degenera o no.
3. Transformación de la variable Sexo, que inicialmente se encontraba representada por los símbolos " " y " ", en valores categóricos, debido a su posible relevancia en la degeneración de los ratones. Los valores asignados fueron:
  - 0 → Femenino.
  - 1 → Masculino.
  - -1 → Sin especificar.
4. A continuación, se decide crear la variable Edad\_ERG en el DataFrame con el objetivo de reducir la cantidad de variables que contienen en su mayoría valores nulos. Esta nueva variable se inicializa con valores NaN.
5. Posteriormente, se define un diccionario llamado `edad_map`, que asigna un valor numérico representativo a cada grupo de edad evaluado (por ejemplo, P30 → 1, 3m → 3, 6m → 6, etc.), considerando 17-21m como 19.
6. Se recorre este diccionario y, para cada columna correspondiente a una edad que contenga valores no nulos, se asigna el valor numérico definido en `edad_map` a la variable Edad\_ERG.
7. Finalmente, para normalizar esta variable, se aplica un escalado Min-Max, transformando sus valores a un rango entre 0 y 1 mediante la siguiente fórmula:

$$\text{EdadERG} = \frac{\text{EdadERG} - \text{mínimo}}{\text{máximo} - \text{mínimo}} \quad (5.1)$$

Donde

- **mínimo** = 1

- **máximo** = 19

Se eliminan las variables de edades originales, ya que toda la información relevante ha sido consolidada en la nueva variable Edad\_ERG. Además, se identifican filas adicionales (47, 48 y 49) que corresponden a encabezados duplicados de la segunda hoja, por lo que se procede a su eliminación para garantizar la coherencia del conjunto de datos.

Para garantizar la integridad y calidad del conjunto de datos, se eliminaron posibles filas duplicadas, aunque en principio no debería haberlas. Además, se realizó un reinicio del índice para mantener una numeración coherente tras la eliminación de registros, quedándonos una tabla como la mostrada en la Figura 5.2.

	Sexo	OD_1	OI_1	OD_2	OI_2	OD_3	OI_3	OD_4	OI_4	Edad Opto	Dato OPT	OD_5	OI_5	degenera	Edad_ERG
0	0.0	224.1	262.1	540.4	628.0	16.8	19.5	146.7	166.1	1.3	0.572	52.818	202.818	0.0	0.0
1	0.0	147.6	217.6	320.6	501.6	21.6	19.9	123.1	135.2	1.3	0.574	47.365	202.364	0.0	0.0
2	0.0	318.8	291.6	719.6	630.5	14.2	19.3	184.3	183.7	1.3	0.594	54.940	209.455	0.0	0.0
3	1.0	206.7	197.6	527.3	492.3	14.3	18.0	131.8	166.9	1.3	0.550	57.273	212.909	0.0	0.0
4	1.0	292.1	198.6	696.6	470.4	17.5	16.5	181.2	142.6	1.3	0.616	51.091	202.545	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
99	0.0	78.1	72.1	174.8	181.2	2.0	3.0	52.0	84.8	0.0	0.308	0.000	0.000	1.0	1.0
100	0.0	65.1	60.8	199.9	188.7	5.0	3.2	63.1	57.1	0.0	0.297	0.000	0.000	1.0	1.0
101	1.0	92.2	83.8	264.3	246.8	3.0	4.0	93.2	85.0	0.0	0.319	0.000	0.000	1.0	1.0
102	1.0	79.0	113.1	191.6	249.4	5.0	7.0	52.5	91.7	0.0	0.308	0.000	0.000	1.0	1.0
103	0.0	6.0	81.9	37.0	241.3	2.0	4.0	9.0	87.6	0.0	0.286	0.000	0.000	1.0	1.0

**Figura 5.2:** Conjunto de datos preparado para entrenar

Finalmente, el conjunto de datos se quedó con un total de **104 instancias y 15 variables.**, incluyendo la objetivo, por lo que hay un total de 14 variables predictoras.

Se llevó a cabo un análisis de la distribución de la variable objetivo (degenera) con el fin de evaluar la necesidad de aplicar técnicas de balanceo en el modelo. Los resultados mostraron que existen 57 instancias en las que se produce degeneración y 47 instancias en las que no, lo que indica que el conjunto de datos se encuentra relativamente equilibrado.

Dado que el tamaño de la base de datos es reducido, la aplicación de técnicas de balanceo o Data Augmentation podría generar instancias con combinaciones de datos poco realistas o

incluso imposibles. Por este motivo, se decidió entrenar el modelo sin realizar ajustes en la proporción de clases, priorizando la preservación de la integridad y representatividad de los datos originales.

### 5.3 Planteamiento de los escenarios

En este proyecto se desarrollarán dos modelos, cada uno diseñado para abordar el problema desde una perspectiva diferente y para ello se han planteado dos escenarios.

- **Primer escenario:** el modelo se entrenará únicamente con las variables relacionadas con el ERG, además del sexo, eliminando aquellas que no aportan información relevante para este análisis y excluyendo la de salida. En total, se utilizarán 10 variables predictoras.
- **Segundo escenario:** el modelo se entrenará con todas las variables disponibles en el *DataFrame* preprocesado, excluyendo la variable de salida. En este caso, se trabajará con un total de 14 variables predictoras.

### 5.4 Primer escenario

Con el objetivo de aprovechar al máximo los datos disponibles, se decidió diseñar un modelo basado exclusivamente en los registros de electroretinografía (ERG). Esta decisión se fundamentó en el hecho de que una proporción considerable de instancias del conjunto de datos contenía únicamente información proveniente de esta prueba. Por tanto, resultaba pertinente evaluar la viabilidad de construir un clasificador que funcionara únicamente con dichos datos, lo cual, de ser exitoso, podría traducirse en un ahorro significativo de tiempo y recursos en el proceso diagnóstico.

No obstante, esta iniciativa partía de una hipótesis inicial poco optimista respecto al rendimiento del modelo. Dicha suposición se sustenta en la experiencia previa del laboratorio, donde, durante la fase experimental, se observó que las pruebas de ERG, por sí solas, no resultaban suficientes para alcanzar un diagnóstico concluyente. En la práctica, se hacía necesario complementar la información obtenida mediante ERG con otras pruebas como la OCT y

---

OPT.

A pesar de estas limitaciones, se llevó a cabo el desarrollo del clasificador con el fin de obtener una precisión lo más alta posible dentro del contexto dado. Sin embargo, es importante señalar que, si bien se busca optimizar su rendimiento, este modelo no pretende sustituir enfoques más completos que integren múltiples fuentes de información, ya que se cuenta con un mayor número de variables que permiten construir modelos más robustos y fiables.

#### 5.4.1 Preparación de datos

Para preparar los datos del segundo modelo, se eliminarán las variables "Edad Opto", "Dato OPT", "OD\_5" y "OI\_5", ya que en este caso solo se trabajará con las medidas obtenidas en las pruebas ERG.

Durante el proceso de preprocesamiento de los datos correspondientes a nuestro segundo escenario, denominado `data_modelo2`, se observó que algunos valores contenían un asterisco. Tras investigar el motivo de su presencia, se concluyó que no tenían un significado relevante, por lo que se decidió eliminarlos de todo el *DataFrame*

Posteriormente, se detectaron de que ciertos números decimales, en lugar de estar separados por un punto (.), estaban separados por una coma (,). Este detalle se identificó una vez ya entrenado y evaluado el modelo, dado que los resultados obtenidos no eran coherentes y la clasificación no era precisa.

Adicionalmente, se eliminó todas las instancias en las que los valores de ERG que estuvieran representados por "?" o "-", ya que estos no aportaban valor al modelo y podrían afectar la calidad de los resultados.

En la siguiente fase, se ajustó el formato de los datos. Inicialmente, se convirtieron todos los valores a tipo *float64*, ya que la mayoría de los datos eran decimales, con el fin de que se mantuviera correctamente la naturaleza de estos.

#### 5.4.2 Entrenamiento de arquitecturas ML

Una vez definidos los conjuntos de variables de entrada y salida, se dividió los datos en un conjunto de entrenamiento (80%) y uno de prueba (20%). Además, se escalaron los todos datos para asegurar que estuvieran en una escala adecuada en caso de utilizar modelos que

---

requirieran como entrada datos escalados.

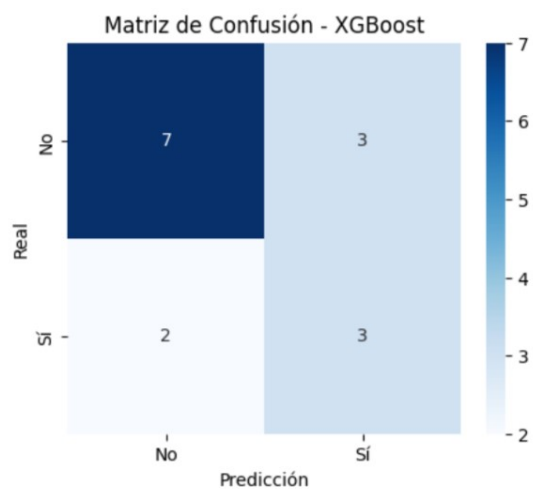
A continuación, se entrenaron los datos utilizando diversos modelos de ML. Los modelos empleados fueron *Random Forest*, SVM y *XGBoost*.

En una primera fase, cada modelo fue entrenado utilizando los valores por defecto (es decir, los más usados y comunes) de sus respectivos hiperparámetros, con el objetivo de obtener una línea base de rendimiento sobre la que realizar comparaciones posteriores y posibles ajustes. Se comienza con *XGBoost*, configurando el clasificador con 100 árboles de decisión, una tasa de aprendizaje de 0.1 y una semilla aleatoria para garantizar la reproducibilidad. Para el modelo SVM, fue necesario escalar previamente las características, dado que este algoritmo es sensible a las diferencias de escala entre las variables. Se usó *StandardScaler* para estandarizar los datos, transformando las características para que tengan media 0 y desviación estándar. Una vez escaladas las variables, se entrena el modelo SVM con un kernel radial (del inglés, *Radial Basis Function*, RBF), habilitando además la estimación de probabilidades. Finalmente, se entrena un *Random Forest* con 100 árboles y semilla aleatoria fija para garantizar resultados reproducibles. Este modelo no requiere escalado de características, por lo que se utiliza directamente el conjunto original de entrenamiento.

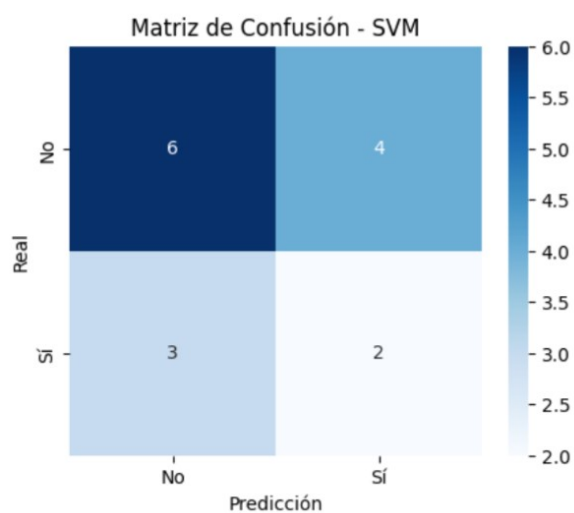
A continuación, en la tabla 5.1 se muestran los resultados de las precisiones obtenidas y las matrices de confusión en las figuras 5.3, 5.4 y 5.5.

Modelo	Precisión <i>train</i>	Precisión <i>test</i>
<i>XGBoost</i>	1.000	0.6667
SVM	0.8136	0.5333
<i>Random Forest</i>	1.000	0.5333

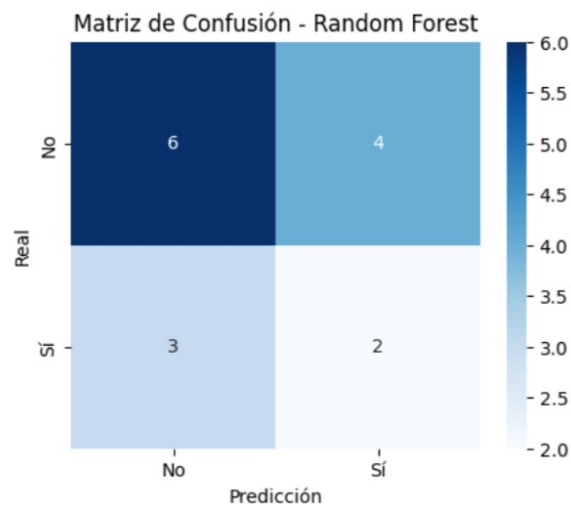
**Tabla 5.1:** Precisión de cada modelo con hiperparámetros por defecto.



**Figura 5.3:** Matriz de confusión de XGBoost con hiperparámetros por defecto



**Figura 5.4:** Matriz de confusión de SVM con hiperparámetros por defecto



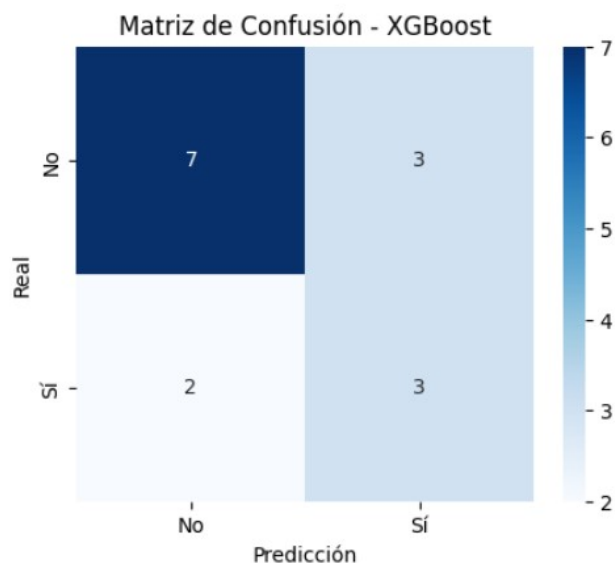
**Figura 5.5:** Matriz de confusión de Random Forest con hiperparámetros por defecto

#### 5.4.2.1 Datos escalados

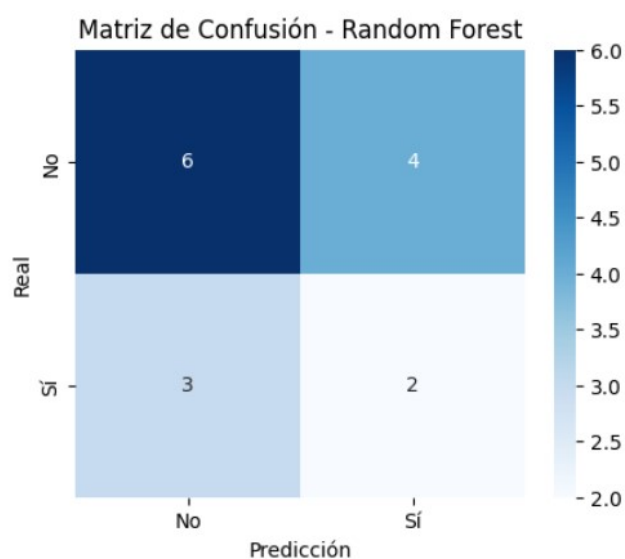
A continuación, se entrenan los modelos *XGBoost* y *Random Forest* con los datos escalados únicamente para probar y ver qué ocurre. En principio no debe haber mucha diferencia de resultados entre usar datos de entrenamientos escalados o no. Véanse los resultados obtenidos en la tabla 5.2 y las figuras 5.6 y 5.7.

Modelo	Precisión <i>train</i>	Precisión <i>test</i>
<i>XGBoost</i>	1.000	0.6667
<i>Random Forest</i>	1.000	0.5333

**Tabla 5.2:** Precisión de *XGBoost* y *Random Forest* entrenados con datos escalados.



**Figura 5.6:** Matriz de confusión de XGBoost con hiperparámetros por defecto y datos escalados



**Figura 5.7:** Matriz de confusión de Random Forest con hiperparámetros por defecto y datos escalados

Se comprobó que los resultados obtenidos por los modelos fueron exactamente los mismos tanto si se entrenaban con los datos escalados como sin escalar, lo que indica que, en este caso concreto, el escalado no afecta al rendimiento del modelo. No obstante, con vistas a una futura implementación del modelo en una aplicación web, se optó por entrenarlo directamente con los datos sin escalar. Esta decisión se justifica por motivos de simplicidad y eficiencia:



si el modelo se entrenara con datos escalados, sería necesario almacenar y aplicar el objeto estandarizador también en la web, lo cual implicaría un paso adicional para transformar los datos introducidos por el usuario en el formulario al mismo rango de valores con el que fue entrenado el modelo. Al evitar este paso, se facilita el proceso de integración y se reduce la complejidad.

#### 5.4.2.2 Optimización de hiperparámetros

Seguidamente, se trata de optimizar los modelos modificando los hiperparámetros característicos de cada uno de los modelos para mejorar su rendimiento. Para ello, se usa *Grid-SearchCV*, que permite evaluar todas las combinaciones posibles de un conjunto predefinido de hiperparámetros mediante validación cruzada, seleccionando la combinación que ofrece mayor rendimiento. A continuación, se discuten los valores obtenidos:

##### **XGBoost**

Para XGBoost, se obtuvieron como hiperparámetros óptimos una tasa de aprendizaje de 0.1, una profundidad máxima de los árboles de 4 y un total de 50 árboles a entrenar.

El hiperparámetro `learning_rate = 0.1` regula cuánto contribuye cada nuevo árbol al modelo global. Un valor bajo, como este, indica que el modelo aprende más lentamente, lo cual suele mejorar la capacidad de generalización y reducir el riesgo de sobreajuste.

En cuanto a `max_depth = 4`, este limita la profundidad de cada árbol individual, lo que evita la creación de estructuras demasiado complejas y favorece la simplicidad, algo especialmente útil en conjuntos de datos pequeños o con ruido.

Por último, se estableció `n_estimators = 50`, que representa la cantidad total de árboles entrenados. Este número moderado, combinado con un aprendizaje lento, permite alcanzar una buena precisión sin saturar el modelo ni aumentar demasiado el coste computacional.

##### **SVM**

En el caso del modelo SVM, los mejores hiperparámetros encontrados fueron  $C = 1$ , `gamma = 'scale'` y el uso del kernel RBF.

---

El parámetro  $C = 1$  establece un equilibrio entre mantener un margen amplio para la separación de clases y penalizar los errores cometidos sobre los datos de entrenamiento. Un valor intermedio como este permite al modelo ser flexible sin caer en el sobreajuste.

El hiperparámetro  $\gamma = \text{'scale'}$  ajusta automáticamente la influencia de cada muestra individual en la función de decisión, basándose en la varianza de las características. Esta configuración permite una buena adaptación del modelo sin necesidad de afinar manualmente este valor.

Finalmente, el uso de un  $\text{kernel} = \text{'rbf'}$  posibilita que el modelo capture relaciones no lineales entre las variables al proyectar los datos.

### **Random Forest**

Para el modelo Random Forest, los hiperparámetros seleccionados como óptimos fueron una profundidad máxima de 5 y un total de 10 árboles a entrenar.

El parámetro  $\text{max\_depth} = 5$  esta restricción impide que los árboles memoricen excesivamente los datos de entrenamiento, ayudando a mantener el modelo más simple, reducir el riesgo de sobreajuste y mejorar su capacidad de generalización.

Por otro lado, el valor  $\text{n\_estimators} = 10$  indica que el modelo se construye a partir de 10 árboles. Aunque este número puede parecer bajo, es suficiente en conjuntos de datos pequeños o moderados. Véanse los resultados obtenidos para cada uno de los modelos en la tabla 5.3 y las imágenes 5.8, 5.10 y 5.9.

Modelo	Precisión <i>train</i>	Precisión <i>test</i>
<i>XGBoost</i>	1.000	0.6667
SVM	0.8136	0.5333
<i>Random Forest</i>	0.9831	0.6000

**Tabla 5.3:** Precisión de cada modelo con *GridSearchCV* aplicado.

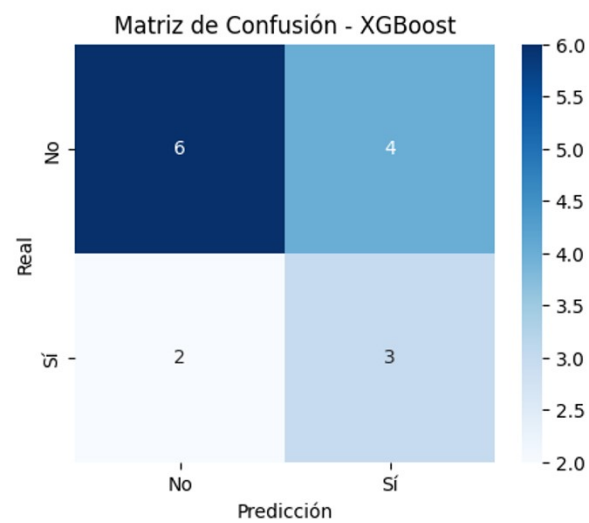


Figura 5.8: Matriz de confusión de XGBoost con hiperparámetros optimizados

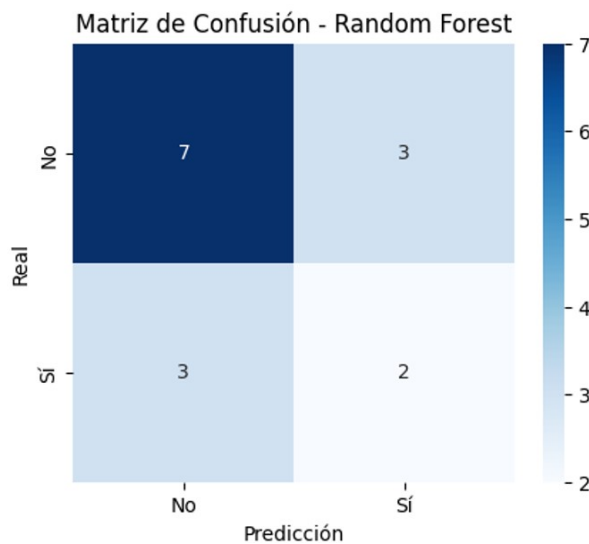
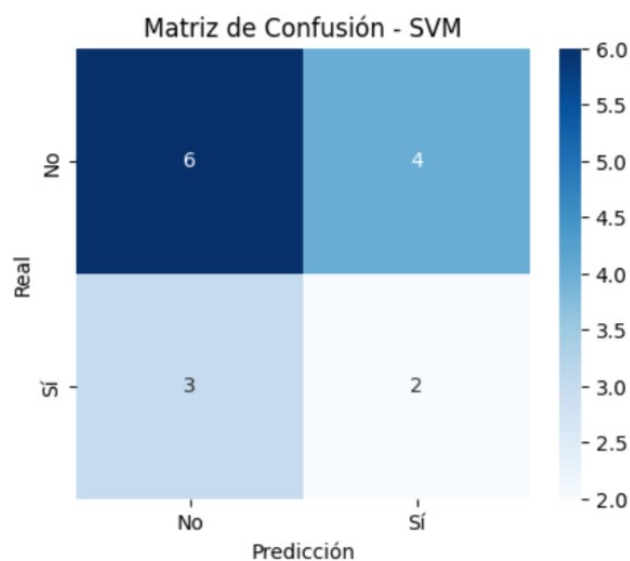


Figura 5.9: Matriz de confusión de Random Forest con hiperparámetros optimizados



**Figura 5.10:** Matriz de confusión de SVM con hiperparámetros optimizados

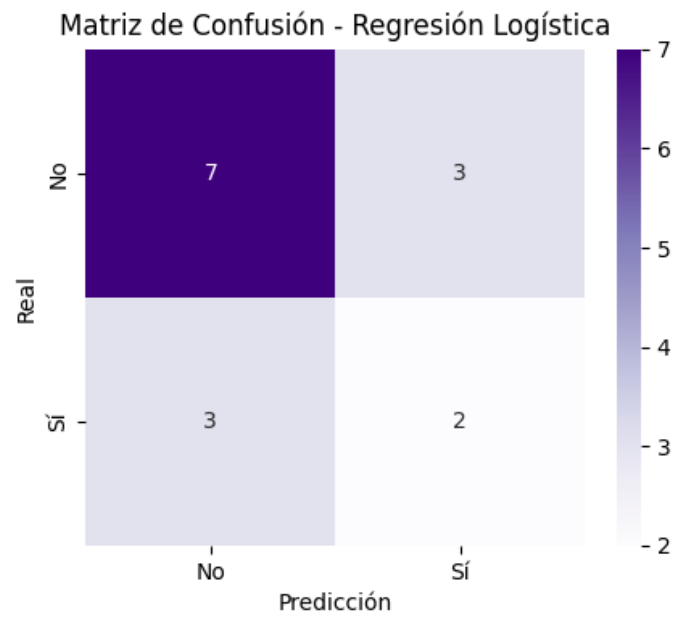
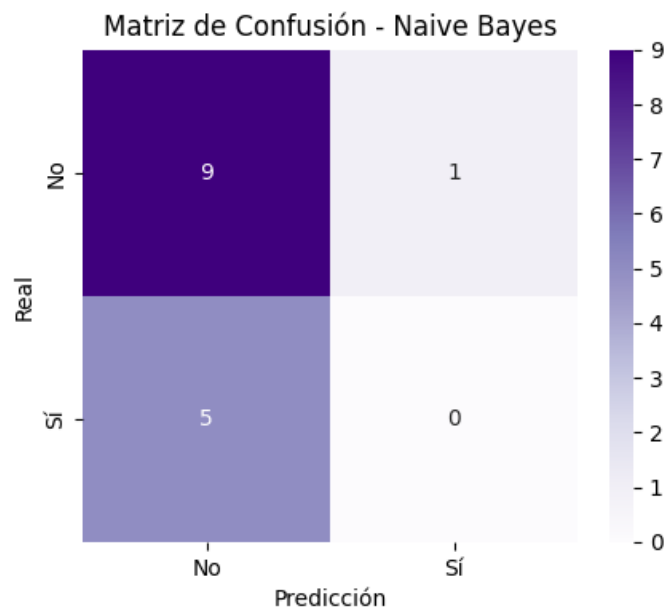
Se observa que Random Forest sube de 0.5333 a 0.6000 y que XGBoost y SVM se mantienen igual.

### 5.4.2.3 Nuevas arquitecturas

Los resultados siguen sin ser satisfactorios, por lo que se procede a entrenar el modelo con otros clasificadores; Naive Bayes y Logistic Regression. La elección de ambos modelos, se debieron principalmente a su sencillez y a su rendimiento destacable en contextos con pocos datos. Dada su sencillez, también son rápidos al entrenar, pero en este caso esa cualidad no nos es de gran interés ya que al haber tan pocas instancias y datos no temporales, los modelos iban a entrenar rápidamente. A continuación, se muestran los resultados obtenidos en 5.4, 5.11 y 5.12

Modelo	Precisión <i>train</i>	Precisión <i>test</i>
Regresión Logística	0.6949	0.6000
Naive Bayes	0.6610	0.6000

**Tabla 5.4:** Precisión de las arquitecturas de Regresión logística y Naive Bayes.

**Figura 5.11:** Matriz de confusión de Regresión Logística**Figura 5.12:** Matriz de confusión de Naive Bayes

A pesar de haber probado con otras arquitecturas, el rendimiento del modelo no mejora lo suficiente, por lo que más tarde se aplicarán otras técnicas para intentar incrementar el accuracy.

### 5.4.3 Entrenamiento de arquitecturas DL

Inicialmente, se entrenaron dos modelos de redes neuronales utilizando los valores típicos de sus respectivos hiperparámetros. Estos modelos fueron una Red Neuronal MLP y una DNN, con el objetivo de explorar sus capacidades en la clasificación de los datos. La posibilidad de entrenar redes neuronales convolucionales (del inglés *Convolutional Neural Network*, CNN) se desechó rápidamente, puesto que están diseñadas para trabajar con datos de estructura espacial, como imágenes, vídeos, etc. También se barajó la posibilidad de trabajar con Redes Neuronales Residuales (del inglés *Residual Networks*, ResNets), una variante de DNN, pero que introducen conexiones residuales, es decir, la información puede saltarse algunas capas intermedias. Estas redes están pensadas para tareas complejas y no resultaba una opción interesante, pues las redes neuronales no iban a ser demasiado profundas debido al conjunto de datos tan pequeño del que disponemos.

#### MLP

Se comenzó con un MLP, el cual es un tipo de red neuronal alimentada hacia adelante (*feedforward*) que consta de varias capas de neuronas. El modelo se configuró con las siguientes características:

##### 1. Estructura de la Red:

- **Capa de entrada:** La capa de entrada se configuró con una función de activación *ReLU* (*Rectified Linear Unit*), que ayuda a introducir no linealidades en el modelo.

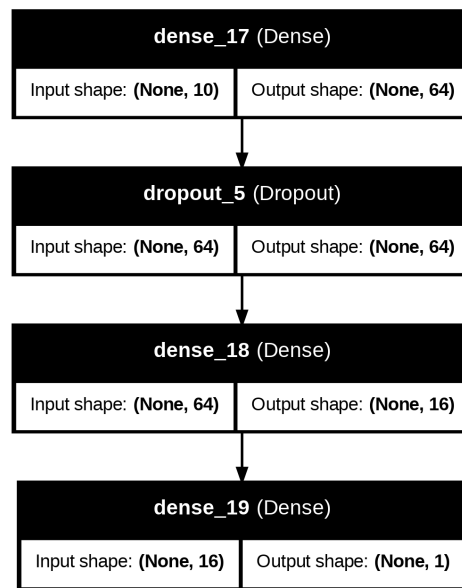
El modelo recibe como entrada un total de 10 variables predictoras, correspondientes a las características obtenidas en pruebas de laboratorio. Aunque en Keras la capa de entrada no se define de forma explícita, se establece de manera implícita en la primera capa densa mediante el parámetro `input_shape=(10,)`. Esto indica que cada instancia de entrada se representa como un vector con 10 valores numéricos. Esta información es fundamental para que la red pueda construir correctamente las conexiones entre las neuronas de entrada y la primera capa oculta.

---

- **Primera capa oculta:** contiene 64 neuronas con función de activación *ReLU*, encargadas de transformar el espacio de características de entrada y detectar relaciones no lineales.
  - **Segunda capa oculta:** cuenta con 16 neuronas, también con activación *ReLU*, que continúa procesando la información extraída previamente para aproximarse al patrón de salida.
  - **Capa de salida:** La capa de salida tiene 1 neurona con una función de activación sigmoide, lo cual es adecuado para problemas de clasificación binaria.
2. **Dropout:** Para prevenir el sobreajuste, se añadió una capa de *dropout* con una tasa de 0.2 entre la primera capa oculta y la segunda. Esto ayuda a evitar que el modelo se ajuste demasiado a los datos de entrenamiento al "apagar" aleatoriamente algunas neuronas durante el entrenamiento.
3. **Compilación del Modelo:** Se utilizó el optimizador *Adam*, que es muy eficaz para redes neuronales y el más utilizado, ya que ajusta los pesos de las neuronas durante el entrenamiento de manera eficiente. Además, usar un optimizador u otro no supone una gran diferencia en la obtención de los resultados. La función de pérdida seleccionada fue *binary\_crossentropy*, adecuada para problemas de clasificación binaria, y se monitorizó la precisión (*accuracy*) durante el entrenamiento.
4. **Entrenamiento:** Se entrenó el modelo durante 50 épocas con un tamaño de *batch* de 8 para que haya más actualizaciones por época y que, por tanto, haya más variabilidad entre *batches*. Además, se empleó un *early stopping* con paciencia de 10 épocas para evitar el sobreajuste, deteniendo el entrenamiento cuando no se observaba mejora en la validación.

A continuación se puede observar la arquitectura de la red neuronal gráficamente 5.13

---



**Figura 5.13:** Arquitectura del modelo MLP implementado

## DNN

El siguiente modelo entrenado fue una *DNN*, que tiene una estructura más compleja debido al mayor número de capas ocultas. La configuración utilizada fue la siguiente:

### 1. Estructura de la red:

- **Capa de entrada:** Al igual que en el *MLP*, la capa de entrada tiene neuronas con función de activación *ReLU*. El modelo recibe como entrada un total de 10 variables predictoras, correspondientes a las características extraídas del conjunto de datos. Esta información se especifica en la primera capa densa a través del parámetro `input_shape=(10,)`, lo que indica que cada instancia de entrada se representa como un vector de 10 valores.
- **Capas ocultas:** Se añadieron tres capas ocultas, la primera con 128, la segunda con 64 neuronas y otra con 32 neuronas, todas con activación *ReLU*.
- **Capa de salida:** Similar al modelo anterior, la capa de salida es de una neurona con activación sigmoide, lo que permite realizar la clasificación binaria.

### 2. Dropout:

Se emplearon dos capas de *dropout* con tasas de 0.3 y 0.2 respectivamente



para reducir el riesgo de sobreajuste. Estas capas ayudan a mejorar la generalización del modelo, especialmente en redes profundas como esta.

3. **Compilación del modelo:** Al igual que el *MLP*, se utilizó el optimizador *Adam* y la función de pérdida fue *binary\_crossentropy*. El modelo también se optimizó para la precisión como métrica principal.
4. **Entrenamiento:** El modelo *DNN* se entrenó durante 100 épocas con un tamaño de *batch* de 8, utilizando la misma técnica de *early stopping* para evitar el sobreajuste. Esta red, debido a su mayor complejidad, estuvo diseñada para capturar patrones más complejos en los datos.

A continuación se puede observar la arquitectura de la red neuronal gráficamente 5.14.

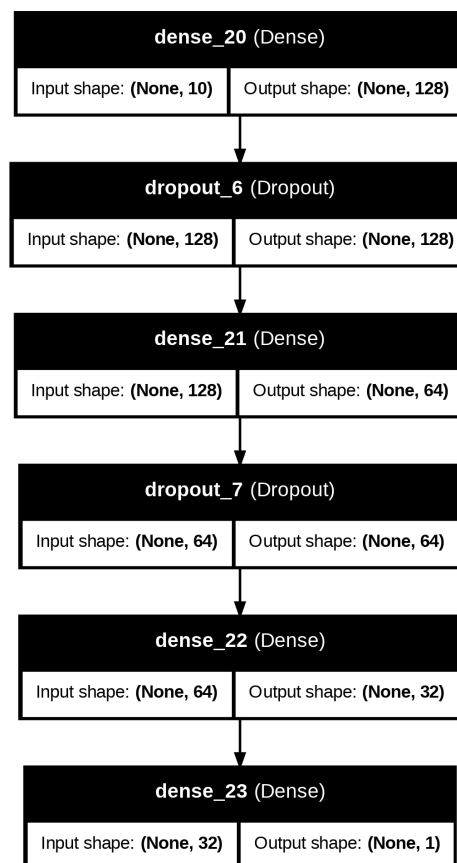
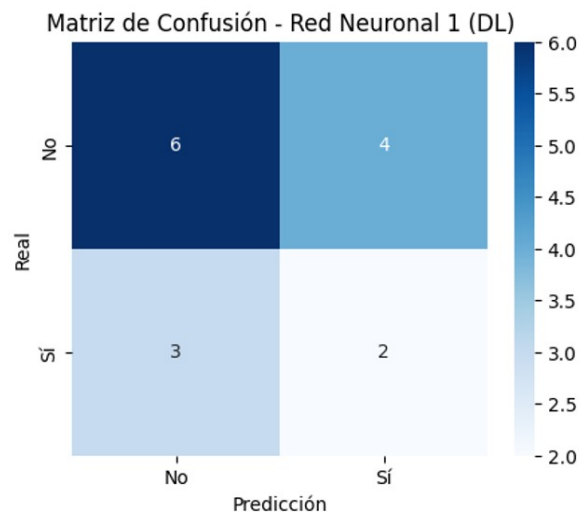


Figura 5.14: Arquitectura del modelo DNN implementado

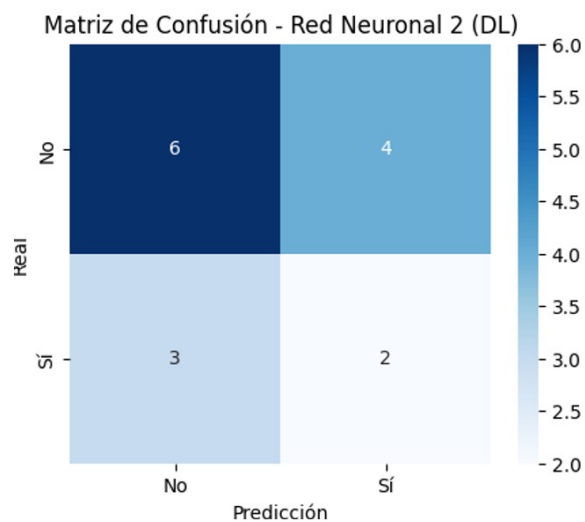
Los resultados se pueden observar en la siguiente tabla 5.5 y en las figuras 5.15 y 5.16

Modelo	Precisión <i>train</i>	Precisión <i>test</i>
MLP	0.7288	0.5333
DNN	0.8136	0.5333

**Tabla 5.5:** Precisión y Matriz de confusión de cada una de las redes neuronales.



**Figura 5.15:** Matriz de confusión de MLP con hiperparámetros por defecto

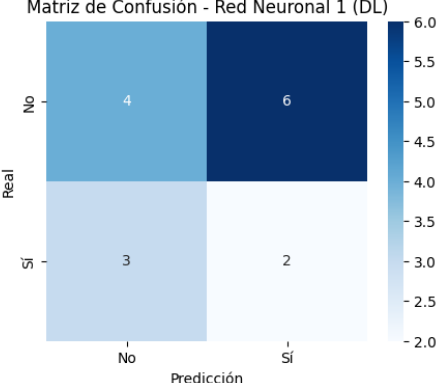
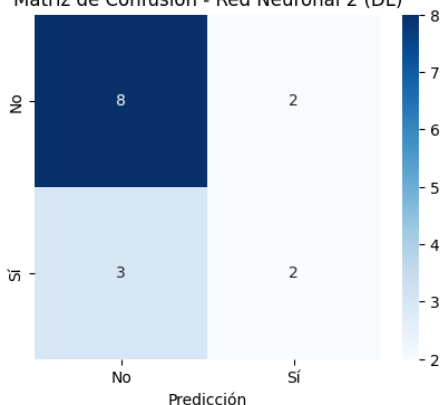


**Figura 5.16:** Matriz de confusión de DNN con hiperparámetros por defecto

Los resultados obtenidos no fueron los esperados, considerando que la clasificación es binaria y que un 0.5 de precisión supondría una clasificación al azar se desechan el posible uso de estos modelos.

#### 5.4.3.1 Eliminar *Dropout*

Como se ha visto el dropout es una técnica utilizada para evitar el sobreajuste, debido a nuestros pésimos resultados, vamos a eliminar el dropout en ambas redes y analizar los resultados que se obtienen en la tabla 5.6

Modelo	Precisión	Matriz de confusión									
MLP	0.4000	<p>Matriz de Confusión - Red Neuronal 1 (DL)</p>  <table border="1"> <thead> <tr> <th></th> <th>Predicción No</th> <th>Predicción Sí</th> </tr> </thead> <tbody> <tr> <th>Real No</th> <td>4</td> <td>6</td> </tr> <tr> <th>Real Sí</th> <td>3</td> <td>2</td> </tr> </tbody> </table>		Predicción No	Predicción Sí	Real No	4	6	Real Sí	3	2
	Predicción No	Predicción Sí									
Real No	4	6									
Real Sí	3	2									
DNN	0.6667	<p>Matriz de Confusión - Red Neuronal 2 (DL)</p>  <table border="1"> <thead> <tr> <th></th> <th>Predicción No</th> <th>Predicción Sí</th> </tr> </thead> <tbody> <tr> <th>Real No</th> <td>8</td> <td>2</td> </tr> <tr> <th>Real Sí</th> <td>3</td> <td>2</td> </tr> </tbody> </table>		Predicción No	Predicción Sí	Real No	8	2	Real Sí	3	2
	Predicción No	Predicción Sí									
Real No	8	2									
Real Sí	3	2									

**Tabla 5.6:** Precisión y Matriz de confusión de cada una de las redes neuronales sin Dropout.

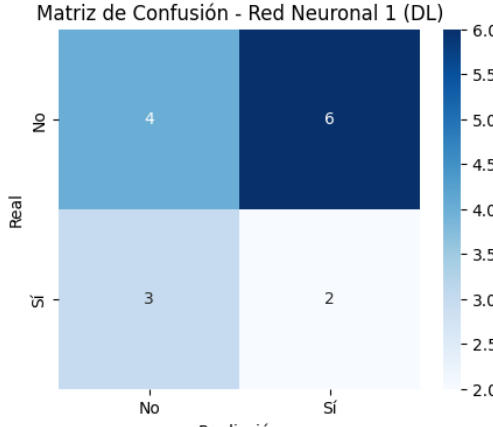
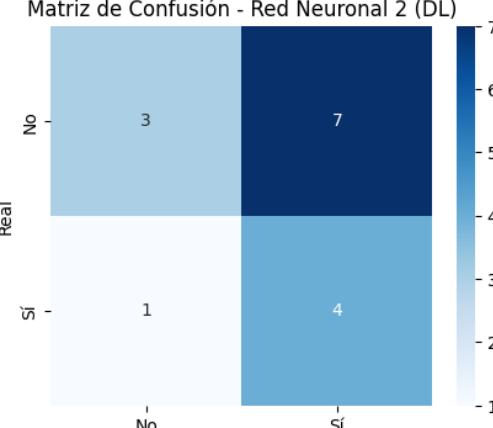
El perceptrón multicapa da un mal resultado, por lo que se dejará el dropout que tenía anteriormente y por el contrario se seguirá experimentando sin dropout en el caso de la DNN.

#### 5.4.4 Reducir el número de neuronas

A continuación, se procede á reducir el número de neuronas a la mitad en todas las capas ocultas de ambos modelos, dado que nuestra base de datos es muy pequeña y no requiere de redes tan complejas a primera vista.

- **MLP:** la primera capa oculta pasa de tener 64 neuronas a 32, y la segunda de 16 a 8
- **DNN:** la primera capa oculta tendrá 64 en vez de 128, la segunda 32 en vez de 64 y la tercera 16 en lugar de 32.

Se muestran los resultados en la siguiente tabla 5.7

Modelo	Precisión	Matriz de confusión
MLP	0.4000	<p>Matriz de Confusión - Red Neuronal 1 (DL)</p>  <p>Matriz de Confusión - Red Neuronal 2 (DL)</p> 
DNN	0.4667	

**Tabla 5.7:** Precisión y Matriz de confusión de cada una de las redes neuronales con menos neuronas por capa.

Se puede observar que esta vez los modelos tenían tan pocas neuronas que puede que no hayan tenido la suficiente capacidad para aprender patrones relevantes de los datos. Seguidamente, se hicieron numerosas pruebas manuales probando con menos neuronas por capa y se vió que la óptima era este número o incluso más. Por esta razón se decide dejar el número de neuronas que se había puesto por defecto.

### 5.4.5 Data Augmentation

Con el fin de mejorar la capacidad de generalización de los modelos y mitigar los efectos de la escasez de datos, se ha considerado oportuno volver a entrenar todos los modelos utilizando técnicas de *Data Augmentation*. El aumento de datos consiste en generar nuevas muestras a partir del conjunto original mediante transformaciones controladas, manteniendo la coherencia con la distribución de los datos reales.

Es importante destacar que el *data augmentation* solo se aplica durante la fase de entrenamiento. Esto significa que las instancias generadas artificialmente no se incluyen en la evaluación del modelo. Al evaluar el modelo sobre el conjunto de prueba, se busca medir su desempeño sobre datos reales, sin modificaciones, para obtener una estimación precisa de su capacidad de generalización. Por lo tanto, en la matriz de confusión de la evaluación, el número de instancias probadas corresponde exclusivamente a las instancias originales del conjunto de prueba, sin incorporar las instancias aumentadas.

Para ello, se han aplicado diversas técnicas de aumentación, y a partir de cada una de ellas se ha creado un nuevo conjunto de datos aumentado. Estas versiones alternativas permiten evaluar cómo afectan distintos enfoques de transformación a la robustez y el rendimiento de los modelos previamente entrenados. Se presentan los resultados de cada arquitectura con las cuatro técnicas aplicadas; *Random Noise*, *Scaling*, *SMOTE* y *Bootstrap Resampling* en las tablas 5.8, 5.9, 5.10, 5.11, 5.12, 5.13 y 5.14:

**Random Forest**

Modelo	Precisión	Matriz de confusión									
<i>Random Noise</i>	0.4667	<p>Matriz de Confusión - Random Forest</p> <table border="1"> <thead> <tr> <th>Real \ Predicción</th><th>No</th><th>Si</th></tr> </thead> <tbody> <tr> <th>No</th><td>5</td><td>5</td></tr> <tr> <th>Si</th><td>3</td><td>2</td></tr> </tbody> </table>	Real \ Predicción	No	Si	No	5	5	Si	3	2
Real \ Predicción	No	Si									
No	5	5									
Si	3	2									
<i>Scaling</i>	0.6667	<p>Matriz de Confusión - Random Forest</p> <table border="1"> <thead> <tr> <th>Real \ Predicción</th><th>No</th><th>Si</th></tr> </thead> <tbody> <tr> <th>No</th><td>7</td><td>3</td></tr> <tr> <th>Si</th><td>3</td><td>2</td></tr> </tbody> </table>	Real \ Predicción	No	Si	No	7	3	Si	3	2
Real \ Predicción	No	Si									
No	7	3									
Si	3	2									
SMOTE	0.6667	<p>Matriz de Confusión - Random Forest</p> <table border="1"> <thead> <tr> <th>Real \ Predicción</th><th>No</th><th>Si</th></tr> </thead> <tbody> <tr> <th>No</th><td>8</td><td>2</td></tr> <tr> <th>Si</th><td>3</td><td>2</td></tr> </tbody> </table>	Real \ Predicción	No	Si	No	8	2	Si	3	2
Real \ Predicción	No	Si									
No	8	2									
Si	3	2									
<i>Bootstrap Resampling</i>	0.6667	<p>Matriz de Confusión - Random Forest</p> <table border="1"> <thead> <tr> <th>Real \ Predicción</th><th>No</th><th>Si</th></tr> </thead> <tbody> <tr> <th>No</th><td>7</td><td>3</td></tr> <tr> <th>Si</th><td>2</td><td>3</td></tr> </tbody> </table>	Real \ Predicción	No	Si	No	7	3	Si	2	3
Real \ Predicción	No	Si									
No	7	3									
Si	2	3									

**Tabla 5.8:** Precisión y Matriz de confusión de random Forest con las diferentes técnicas de Data Augmentation

SVM

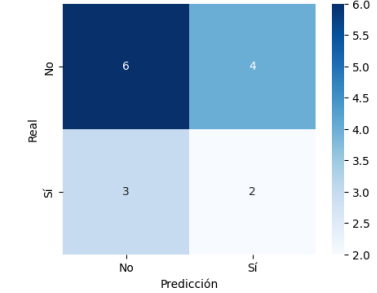
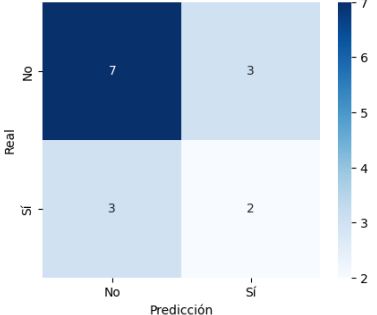
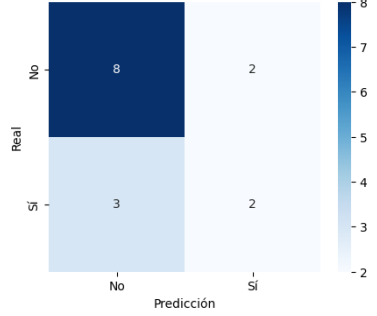
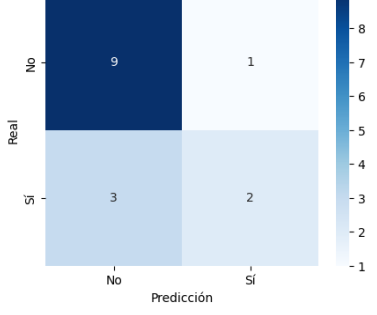
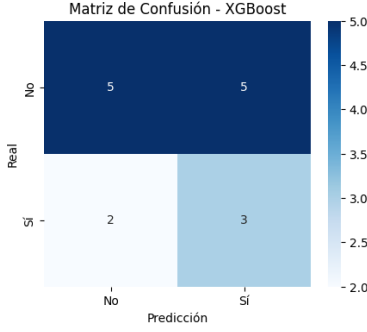
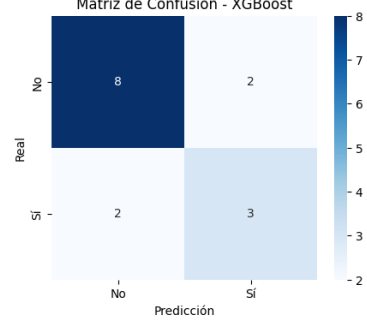
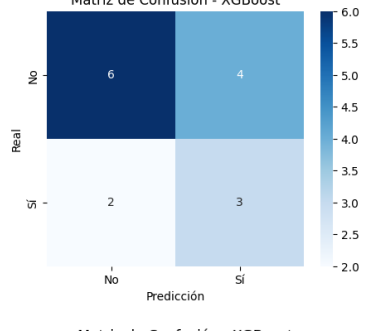
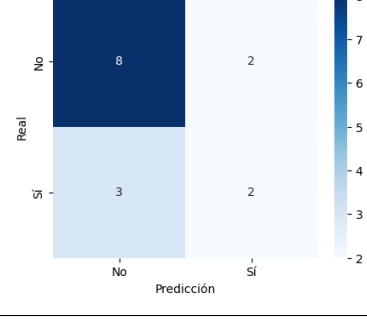
Modelo	Precisión	Matriz de confusión
<i>Random Noise</i>	0.5333	<div>Matriz de Confusión - SVM</div> 
<i>Scaling</i>	0.6000	<div>Matriz de Confusión - SVM</div> 
SMOTE	0.6667	<div>Matriz de Confusión - SVM</div> 
<i>Bootstrap Resampling</i>	0.7333	<div>Matriz de Confusión - SVM</div> 

Tabla 5.9: Precisión y Matriz de confusión de SVM con las diferentes técnicas de Data Augmentation

**XGBoost**

Modelo	Precisión	Matriz de confusión									
<i>Random Noise</i>	0.5333	 <p>Matriz de Confusión - XGBoost</p> <table border="1"> <thead> <tr> <th>Real \ Predicción</th><th>No</th><th>Sí</th></tr> </thead> <tbody> <tr> <th>No</th><td>5</td><td>5</td></tr> <tr> <th>Sí</th><td>2</td><td>3</td></tr> </tbody> </table>	Real \ Predicción	No	Sí	No	5	5	Sí	2	3
Real \ Predicción	No	Sí									
No	5	5									
Sí	2	3									
<i>Scaling</i>	0.7333	 <p>Matriz de Confusión - XGBoost</p> <table border="1"> <thead> <tr> <th>Real \ Predicción</th><th>No</th><th>Sí</th></tr> </thead> <tbody> <tr> <th>No</th><td>8</td><td>2</td></tr> <tr> <th>Sí</th><td>2</td><td>3</td></tr> </tbody> </table>	Real \ Predicción	No	Sí	No	8	2	Sí	2	3
Real \ Predicción	No	Sí									
No	8	2									
Sí	2	3									
SMOTE	0.6000	 <p>Matriz de Confusión - XGBoost</p> <table border="1"> <thead> <tr> <th>Real \ Predicción</th><th>No</th><th>Sí</th></tr> </thead> <tbody> <tr> <th>No</th><td>6</td><td>4</td></tr> <tr> <th>Sí</th><td>2</td><td>3</td></tr> </tbody> </table>	Real \ Predicción	No	Sí	No	6	4	Sí	2	3
Real \ Predicción	No	Sí									
No	6	4									
Sí	2	3									
<i>Bootstrap Resampling</i>	0.6667	 <p>Matriz de Confusión - XGBoost</p> <table border="1"> <thead> <tr> <th>Real \ Predicción</th><th>No</th><th>Sí</th></tr> </thead> <tbody> <tr> <th>No</th><td>8</td><td>2</td></tr> <tr> <th>Sí</th><td>3</td><td>2</td></tr> </tbody> </table>	Real \ Predicción	No	Sí	No	8	2	Sí	3	2
Real \ Predicción	No	Sí									
No	8	2									
Sí	3	2									

**Tabla 5.10:** Precisión y Matriz de confusión de *XGBoost* con las diferentes técnicas de Data Augmentation



Regresión Logística

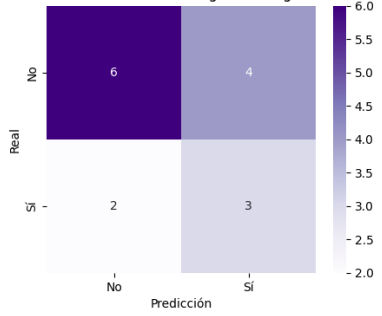
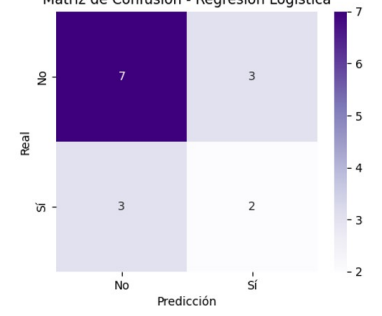
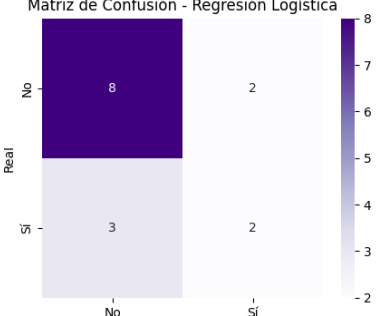
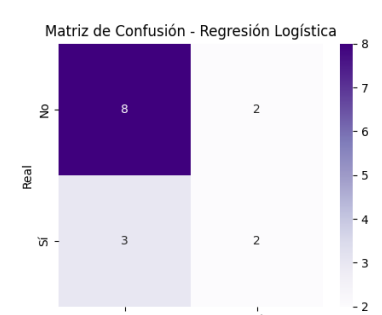
Modelo	Precisión	Matriz de confusión									
<i>Random Noise</i>	0.6000	<div>Matriz de Confusión - Regresión Logística</div>  <table><tr><td>Real \ Predicción</td><td>No</td><td>Sí</td></tr><tr><td>No</td><td>6</td><td>4</td></tr><tr><td>Sí</td><td>2</td><td>3</td></tr></table>	Real \ Predicción	No	Sí	No	6	4	Sí	2	3
Real \ Predicción	No	Sí									
No	6	4									
Sí	2	3									
<i>Scaling</i>	0.6000	<div>Matriz de Confusión - Regresión Logística</div>  <table><tr><td>Real \ Predicción</td><td>No</td><td>Sí</td></tr><tr><td>No</td><td>7</td><td>3</td></tr><tr><td>Sí</td><td>3</td><td>2</td></tr></table>	Real \ Predicción	No	Sí	No	7	3	Sí	3	2
Real \ Predicción	No	Sí									
No	7	3									
Sí	3	2									
SMOTE	0.6667	<div>Matriz de Confusión - Regresión Logística</div>  <table><tr><td>Real \ Predicción</td><td>No</td><td>Sí</td></tr><tr><td>No</td><td>8</td><td>2</td></tr><tr><td>Sí</td><td>3</td><td>2</td></tr></table>	Real \ Predicción	No	Sí	No	8	2	Sí	3	2
Real \ Predicción	No	Sí									
No	8	2									
Sí	3	2									
<i>Bootstrap Resampling</i>	0.6667	<div>Matriz de Confusión - Regresión Logística</div>  <table><tr><td>Real \ Predicción</td><td>No</td><td>Sí</td></tr><tr><td>No</td><td>8</td><td>2</td></tr><tr><td>Sí</td><td>3</td><td>2</td></tr></table>	Real \ Predicción	No	Sí	No	8	2	Sí	3	2
Real \ Predicción	No	Sí									
No	8	2									
Sí	3	2									

Tabla 5.11: Precisión y Matriz de confusión de Regresión Logística con las diferentes técnicas de Data Augmentation

## Naive Bayes

Modelo	Precisión	Matriz de confusión									
<i>Random Noise</i>	0.6667	<p>Matriz de Confusión - Naive Bayes</p> <table border="1"> <thead> <tr> <th>Real \ Predicción</th> <th>No</th> <th>Sí</th> </tr> </thead> <tbody> <tr> <th>No</th> <td>10</td> <td>0</td> </tr> <tr> <th>Sí</th> <td>5</td> <td>0</td> </tr> </tbody> </table>	Real \ Predicción	No	Sí	No	10	0	Sí	5	0
Real \ Predicción	No	Sí									
No	10	0									
Sí	5	0									
<i>Scaling</i>	0.6000	<p>Matriz de Confusión - Naive Bayes</p> <table border="1"> <thead> <tr> <th>Real \ Predicción</th> <th>No</th> <th>Sí</th> </tr> </thead> <tbody> <tr> <th>No</th> <td>9</td> <td>1</td> </tr> <tr> <th>Sí</th> <td>5</td> <td>0</td> </tr> </tbody> </table>	Real \ Predicción	No	Sí	No	9	1	Sí	5	0
Real \ Predicción	No	Sí									
No	9	1									
Sí	5	0									
SMOTE	0.6667	<p>Matriz de Confusión - Naive Bayes</p> <table border="1"> <thead> <tr> <th>Real \ Predicción</th> <th>No</th> <th>Sí</th> </tr> </thead> <tbody> <tr> <th>No</th> <td>9</td> <td>1</td> </tr> <tr> <th>Sí</th> <td>4</td> <td>1</td> </tr> </tbody> </table>	Real \ Predicción	No	Sí	No	9	1	Sí	4	1
Real \ Predicción	No	Sí									
No	9	1									
Sí	4	1									
<i>Bootstrap Resampling</i>	0.7333	<p>Matriz de Confusión - Naive Bayes</p> <table border="1"> <thead> <tr> <th>Real \ Predicción</th> <th>No</th> <th>Sí</th> </tr> </thead> <tbody> <tr> <th>No</th> <td>10</td> <td>0</td> </tr> <tr> <th>Sí</th> <td>4</td> <td>1</td> </tr> </tbody> </table>	Real \ Predicción	No	Sí	No	10	0	Sí	4	1
Real \ Predicción	No	Sí									
No	10	0									
Sí	4	1									

**Tabla 5.12:** Precisión y Matriz de confusión de Naive Bayes con las diferentes técnicas de Data Augmentation

**MLP**

Técnica de <i>Data Augmentation</i>	Accuracy
<i>Random Noise</i>	0.7333
<i>Scaling</i>	0.6000
<i>SMOTE</i>	0.3333
<i>Bootstrap Resampling</i>	0.7333

**Tabla 5.13:** Precisión obtenida en MLPL por cada técnica de *Data Augmentation*.**DNN**

Técnica de <i>Data Augmentation</i>	Accuracy
<i>Random Noise</i>	0.5333
<i>Scaling</i>	0.5333
<i>SMOTE</i>	0.6000
<i>Bootstrap Resampling</i>	0.7333

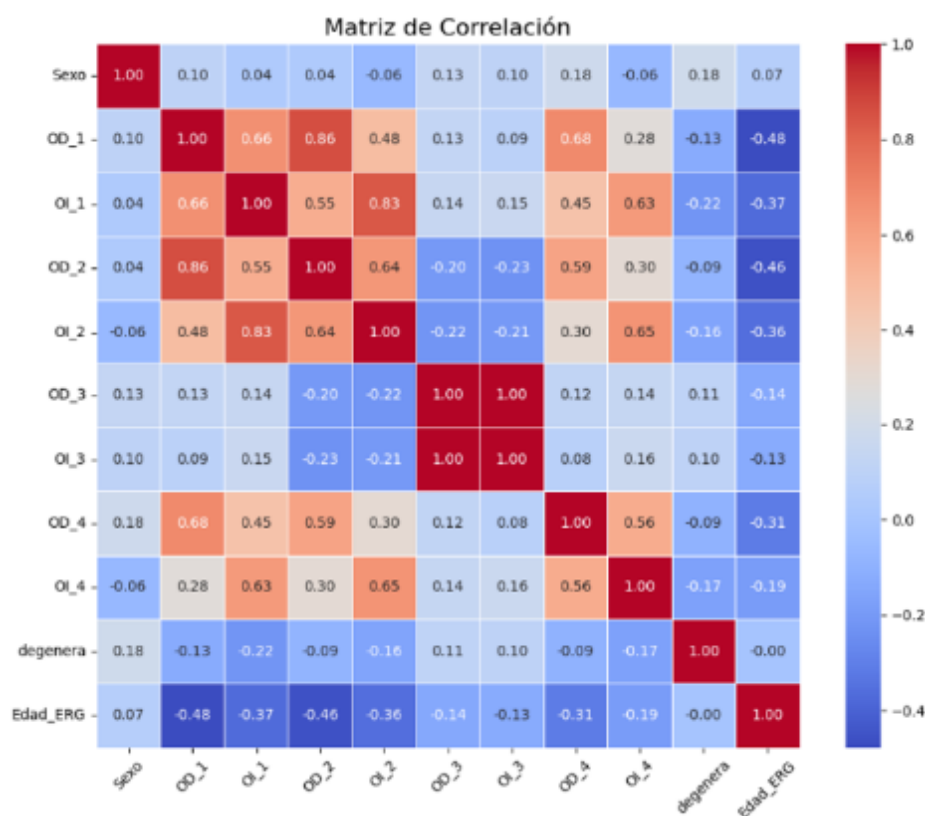
**Tabla 5.14:** Precisión obtenida en DNN por cada técnica de *Data Augmentation*.

En general, la técnica que da mejor resultado es la de *Bootstrap Resampling*, y la que peor, la de *Random Noise*. *Scaling* y *SMOTE* dan resultados medios. Esto se tendrá en cuenta para la realización del modelo para el segundo escenario.

En este primer escenario, basado exclusivamente en los datos de *ERG*, el modelo de clasificación binaria alcanzó una precisión (*accuracy*) máxima de 0.73. Aunque este valor no puede considerarse óptimo en términos absolutos, debe contextualizarse dentro de las limitaciones inherentes al conjunto de datos, ya que, como se ha señalado previamente, los registros de *ERG* por sí solos presentan alta variabilidad y baja representatividad clínica. A pesar de estas restricciones, el resultado obtenido ofrece una base de comparación válida y pone de manifiesto el potencial predictivo que puede extraerse, en cierta medida, a partir de esta única fuente de información.

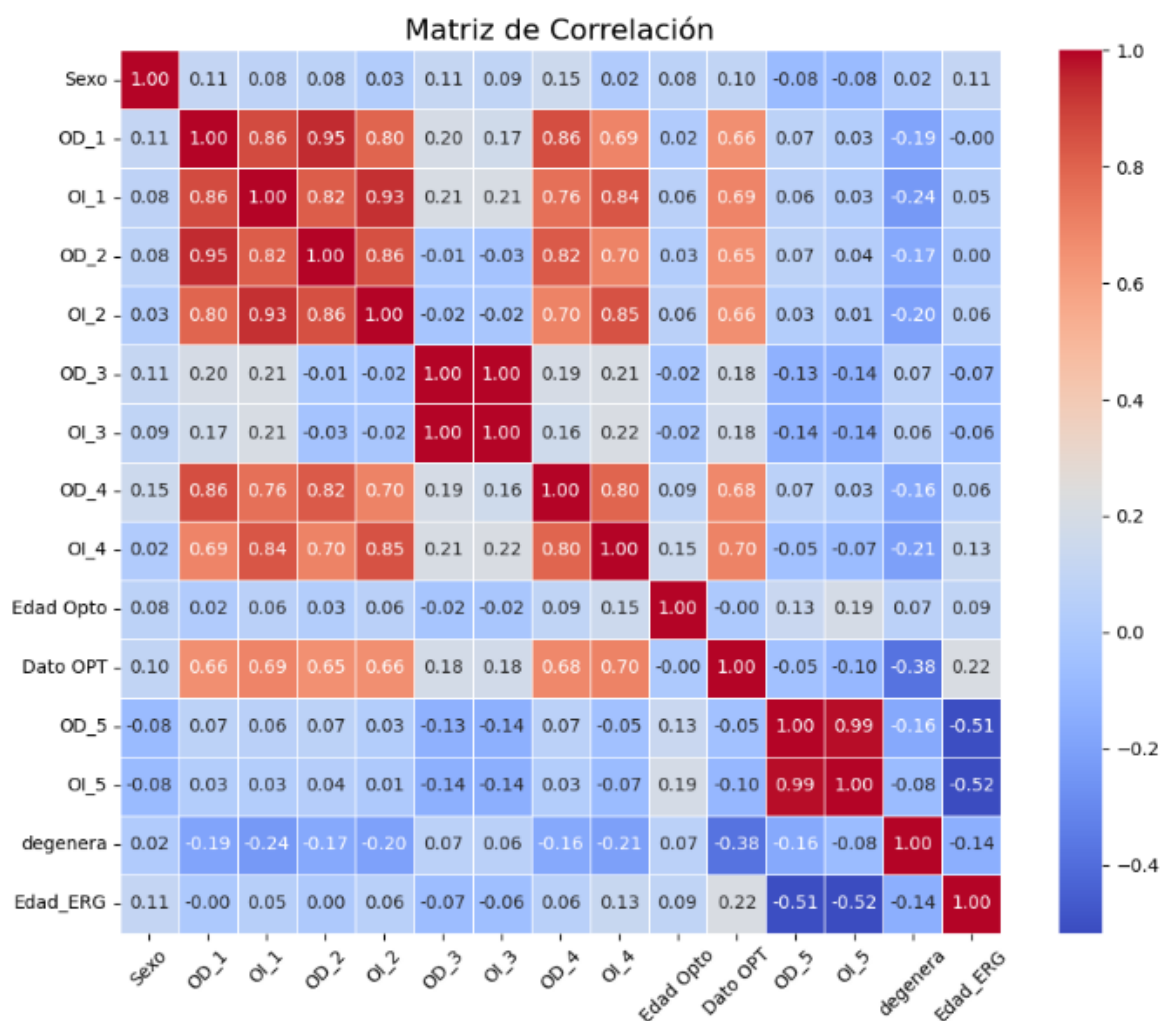
### 5.4.6 Matriz de correlación

Con el fin de profundizar en el comportamiento de las variables utilizadas en ambos escenarios, se realizará un análisis exploratorio de las respectivas matrices de correlación (véase en la Figura 5.17 y Figura 5.18). Este análisis tiene como objetivo identificar posibles relaciones lineales entre las variables de entrada y, en particular, determinar si algunas de ellas presentan una alta colinealidad o una baja variabilidad que pueda afectar negativamente al rendimiento del modelo.



**Figura 5.17:** Matriz de correlación con las variables de purebas ERG

Esta matriz no muestra asociaciones significativas con la variable **degenera**. Todas las correlaciones obtenidas en esta matriz fueron bajas, oscilando entre -0.22 y +0.18. La correlación más destacada en términos absolutos fue con **OI\_1** ( $r = -0.22$ ), aunque sigue siendo insuficiente para establecer una relación relevante.



**Figura 5.18:** Matriz de correlación con todas las variables

En la matriz de la Figura 5.18, se observa una correlación negativa moderada entre **degenera** y la variable **Dato OPT** ( $r = -0.38$ ), lo que sugiere que a medida que disminuyen los valores de dicho parámetro óptico, aumenta la presencia de degeneración visual.

Este hallazgo pone de manifiesto la importancia de contar con un conjunto de variables suficientemente amplio y clínicamente relevante para analizar adecuadamente la degeneración visual. En particular, la variable **Dato OPT** destaca como el principal predictor potencial, lo que refuerza la hipótesis de que los datos de *ERG*, de manera aislada, no contienen suficiente información discriminativa para un diagnóstico preciso, y subraya la importancia de integrar otros factores clínicos adicionales para mejorar el desempeño del modelo.

## 5.5 Segundo escenario

En este segundo escenario, se desarrolla un modelo que utiliza la totalidad de las variables disponibles, lo que permite alcanzar una mayor precisión predictiva en comparación con modelos contruidos a partir de subconjuntos de datos.

### 5.5.1 Preparación de datos

Para el procesamiento de datos del primer escenario, a diferencia del segundo, no se eliminó ninguna variable adicional. Se llevó a cabo también la eliminación del asterisco en algunos valores, las "m" que hacían referencia a los meses de las edades y se reemplazaron las comas (',') por puntos ('.').

Al igual que en el segundo escenario, también se reemplazan los valores "-" y "?" por NaN, se eliminan las filas donde todas las columnas sean NaN y también se convierten todos los valores al formato tipo *float64*.

### 5.5.2 Entrenamiento de arquitecturas ML

En este escenario, se utilizaron las mismas arquitecturas de aprendizaje automático que en el escenario anterior: *Random Forest*, *Support Vector Machine (SVM)* y *XGBoost*. La fase inicial consistió en el entrenamiento de los modelos sin aplicar la técnica de búsqueda de hiperparámetros mediante *GridSearchCV*, ni estrategias de aumento de datos (*Data Augmentation*). Los resultados obtenidos en esta primera etapa se pueden observar en la Tabla 5.15:

Modelo	Precisión <i>train</i>	Precisión <i>test</i>
Random Forest	1.000	1.000
SVM	0.7711	0.8095
XGBoost	1.000	0.9048

**Tabla 5.15:** Precisión de arquitecturas de aprendizaje automático sin *GridSearchCV* ni *Data Augmentation*.

En cuanto a los resultados obtenidos sorprendentemente, el modelo Random Forest ha

alcanzado una precisión perfecta tanto en el conjunto de entrenamiento como en el de test, lo que indica un rendimiento excelente. Este resultado resulta llamativo, ya que no es habitual que ocurra. Sin embargo, este modelo no será el seleccionado, puesto que es una arquitectura muy antigua y sencilla y la versión de *numpy* actual no es compatible con el módulo que esta arquitectura necesita.

Por su parte, el modelo SVM muestra una precisión elevada y equilibrada (77,1% en entrenamiento y 80,95% en test), lo que sugiere una buena capacidad de generalización. En el caso de XGBoost, se observa un posible sobreajuste, ya que la precisión en entrenamiento es del 100%, mientras que en test desciende ligeramente (90,48%).

Posteriormente, se procedió a una segunda fase en la que se implementó *GridSearchCV* para optimizar los hiperparámetros de cada modelo, manteniéndose la ausencia de técnicas de *Data Augmentation*. En este caso, los resultados mostraron una disminución en la precisión del modelo Random Forest, manteniéndose el mismo valor para SVM y una ligera reducción el rendimiento de *XGBoost*, obsérvese la precisión en el entrenamiento y en el test en la tabla 5.16

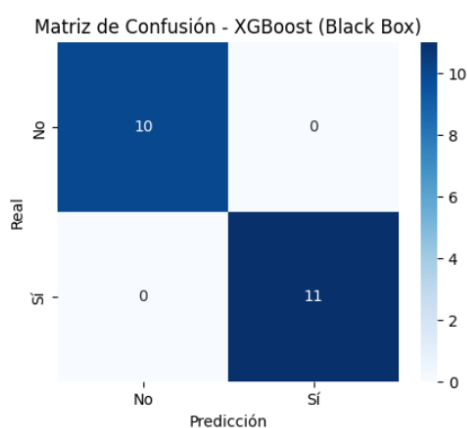
Modelo	Precisión <i>train</i>	Precisión <i>test</i>
Random Forest	1.000	0.8095
SVM	0.7711	0.8095
XGBoost	0.9759	0.8571

**Tabla 5.16:** Resultados tras aplicar *GridSearchCV*, sin *Data Augmentation*.

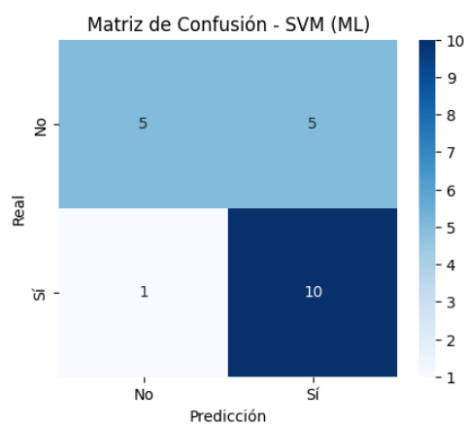
Finalmente, los modelos fueron configurados directamente con los hiperparámetros óptimos y entrenados con los datos aumentados, con la técnica *Bootstrap Resampling*, como se hizo en el primer escenario. Los resultados obtenidos se presentan en la siguiente tabla 5.17 y en las figuras 5.19, 5.20 y 5.21:

Modelo	Precisión <i>train</i>	Precisión <i>test</i>
<i>XGBoost</i>	0.8554	1.000
SVM	0.7470	0.7143
<i>Random Forest</i>	0.8795	0.9048

**Tabla 5.17:** Precisión de *XGBoost*, *SVM* y *Random Forest* entrenados con hiperparámetros óptimos y datos aumentados

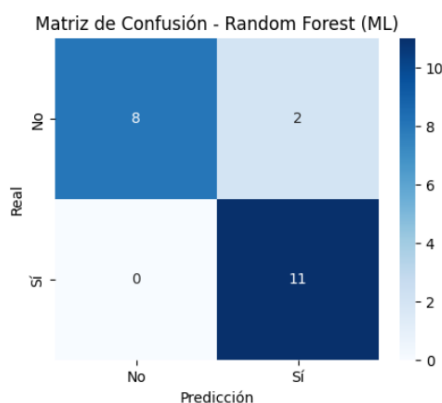


**Figura 5.19:** Matriz de confusión de XGBoost con GridSearchCV y Bootstrap aplicado



**Figura 5.20:** Matriz de confusión de SVM con GridSearchCV y Bootstrap aplicado





**Figura 5.21:** Matriz de confusión de Random Forest con GridSearchCV y Bootstrap aplicado

Tras evaluar el rendimiento del modelo, se observó que no se ha producido sobreajuste, ya que no alcanza un rendimiento perfecto sobre los datos de entrenamiento (85%). Este comportamiento es habitualmente una señal positiva, ya que indica que el modelo no ha memorizado los datos.

Sin embargo, resulta llamativo que el modelo haya logrado acertar todas las predicciones en el conjunto de test, lo que podría interpretarse como una excelente capacidad de generalización. No obstante, este resultado debe interpretarse con cautela, ya que puede que el rendimiento sea perfecto porque el tamaño del conjunto test es muy reducido y se deba a una coincidencia aleatoria.

### 5.5.3 Cross Validation

Dado que existe la posibilidad de que la partición de datos utilizada no sea del todo representativa, o que el modelo haya acertado todas las predicciones en test por simple casualidad debido al tamaño reducido de la muestra, se decidió aplicar validación cruzada para obtener una evaluación más fiable y generalizable.

En este contexto, se opta por centrar los esfuerzos en mejorar el modelo *XGBoost*, que actualmente presenta una precisión del 1.000 en test y 0.8554 en entrenamiento, ya que, además de su buen rendimiento inicial, es un modelo especialmente sencillo de integrar en una aplicación web.

A continuación, se procede a configurar el modelo *XGBoost* con los valores de sus hiperparámetros optimizados, entrenarlo con los datos aumentados y evaluarlo utilizando la técnica

de validación cruzada. Dado que el conjunto de datos cuenta con 104 instancias, de primeras se empleó una validación cruzada con 10 particiones ( $cv = 10$ ). Esta configuración permitía entrenar el modelo con el 90% de los datos en cada iteración, maximizando el uso de la muestra disponible y proporcionando una estimación más fiable del rendimiento general del modelo.

Posteriormente se volvió a aplicar validación cruzada con 5 particiones, ya que es lo más común y se obtuvo los siguientes resultados. Véase tabla 5.18

Modelo	Precisión en validación cruzada	Media
$cv = 10$	[1.0, 0.8889, 0.8889, 0.75, 0.875, 0.875, 1.0, 1.0, 0.875, 0.75]	0.8903
$cv = 5$	[1.0, 0.7647, 0.8824, 1.0, 0.875]	0.9044

**Tabla 5.18:** Precisión en validación cruzada para XGBoost con 10 y 5 particiones

#### 5.5.4 Entrenamiento de arquitecturas DL

En el caso de las redes neuronales, también se evaluaron las mismas que se usaron en el primer escenario y siguiendo el mismo procedimiento que se ha llevado a cabo con las arquitecturas de *Machine Learning*.

Se ha de comentar que esta vez, la capa de entrada contará con 14 nodos, ya que ahora las variables predictoras en este escenario son 14.

En primer lugar, se entrenaron los modelos con la técnica de regularización *Dropout* y sin aplicar *Data Augmentation*. Véanse los resultados obtenidos en la Tabla 5.19:

Modelo	Accuracy
MLP	0.7619
DNN	0.8095

**Tabla 5.19:** Resultados de precisión con *Dropout*, sin *Data Augmentation*.

Posteriormente, se eliminaron las capas de *Dropout* en las *DNN*, como se procedió a hacer en el primer escenario. Sin embargo, los valores se mantuvieron iguales (véase Tabla 5.20):

Modelo	Accuracy
MLP	0.7619
DNN	0.8095

Tabla 5.20: Resultados de precisión tras eliminar *Dropout*, sin *Data Augmentation*.

Finalmente, se aplicó la técnica de *Data Augmentation* mediante *Bootstrap*, obteniéndose nuevamente los mismos valores, se pueden observar en la tabla 5.21 de precisión para ambas arquitecturas. Esta falta de variación a través de diferentes configuraciones indica una estabilidad en el rendimiento, pero también una limitación en su capacidad de mejora dentro del presente conjunto de datos y metodología.

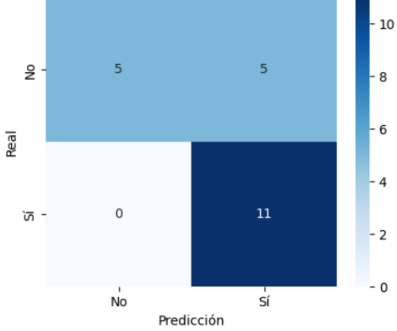
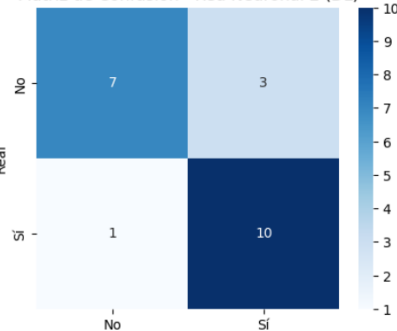
Modelo	Precisión	Matriz de confusión
<i>MLP</i>	0.7619	<div>Matriz de Confusión - Red Neuronal 1 (DL) </div>
<i>DNN</i>	0.8095	<div>Matriz de Confusión - Red Neuronal 2 (DL) </div>

Tabla 5.21: Resultados obtenidos de las redes neuronales entrenadas con hiperparámetros óptimos y datos aumentados

## 5.6 Creación y Gestión de la base de Datos

Para el almacenamiento de los datos utilizados en la aplicación web, se ha optado por emplear el sistema gestor de bases de datos *MySQL*. La elección de esta tecnología se fundamenta en varios motivos: se trata de una herramienta de código abierto, ampliamente utilizada en entornos profesionales y académicos, y ha sido utilizada de forma recurrente durante el desarrollo del grado universitario. Además, *MySQL* presenta una mayor madurez y estabilidad en comparación con otras opciones.

Como herramienta para la gestión visual de la base de datos, se ha utilizado *MySQL Workbench*, también de código abierto y desarrollada por la propia organización responsable de *MySQL*. Esta interfaz gráfica facilita tanto la creación como la administración de las bases de datos y sus respectivas tablas.

El proceso seguido fue el siguiente:

### 1. Creación del usuario y base de datos:

Se creó un usuario en el entorno *MySQL* con nombre `root` y contraseña `12345`. A continuación, se creó una base de datos llamada `animales`.

### 2. Diseño y creación de la tabla:

Se elaboró un script *SQL* que define la estructura de la tabla principal, también llamada `animales`, especificando los campos correspondientes y sus respectivos tipos de datos. Cabe destacar que en el csv no se eliminó la variable `IDanimal` y que no se modificó su tipo de dato original (por ejemplo, a `float64`), manteniendo su integridad como identificador único.

### 3. Importación de datos:

Una vez creada la estructura de la tabla, se procedió a insertar los datos contenidos en un archivo `.csv`. Para ello, se ejecutó el script en *MySQL Workbench*, lo que permitió cargar correctamente los registros en la base de datos. Véase la base de datos ya creada en la Figura 5.22

### 4. Acceso a través del servidor:

Con la base de datos ya creada, se configuró su acceso desde el *backend* de la aplicación

---

web a través del servidor, permitiendo así realizar consultas desde el módulo consultor y alimentar dinámicamente la interfaz del usuario.

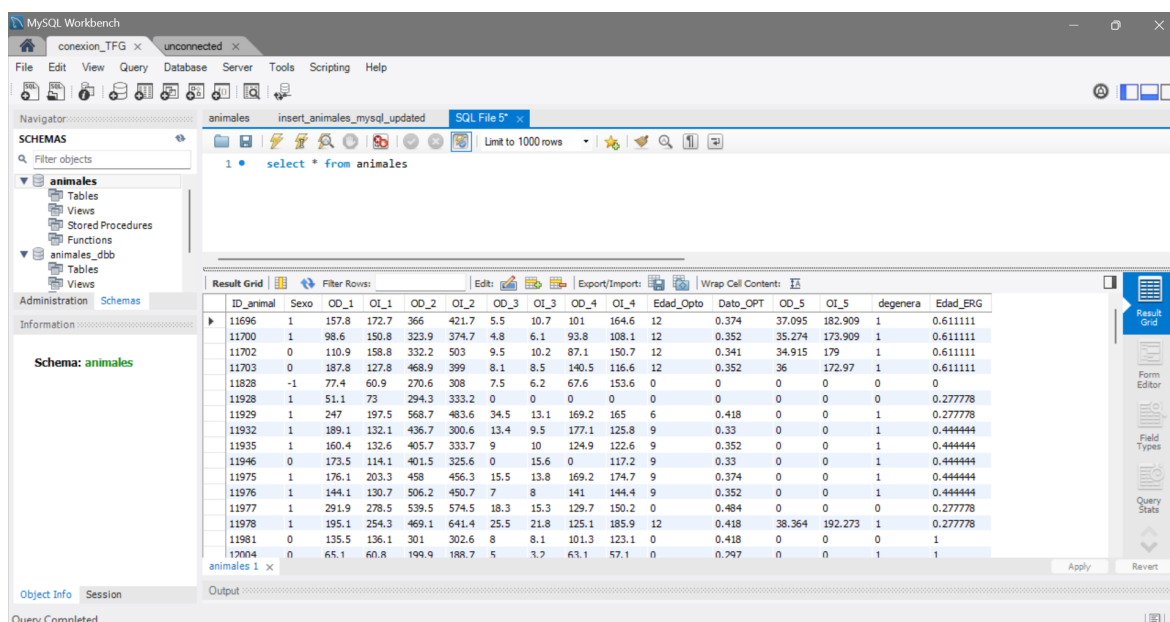


Figura 5.22: Base de datos creada desde *MySQL Workbench*

## 5.7 Aplicación web

### 5.7.1 Arquitectura del Sistema y componentes

El sistema utiliza una arquitectura *cliente-servidor* propia de las aplicaciones web Mozilla (s.f.). Véase un pequeño esquema en la Figura 5.23.

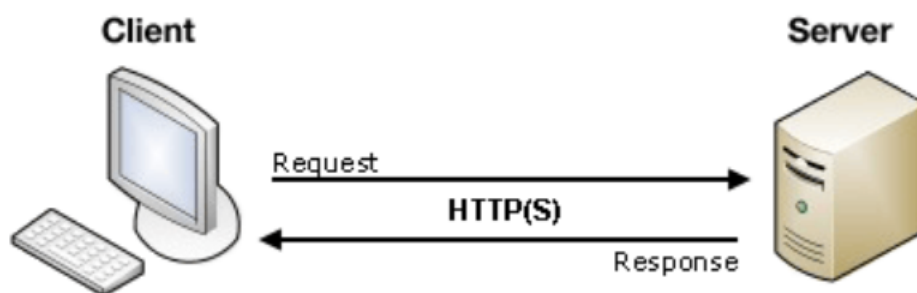


Figura 5.23: Arquitectura cliente-servidor en aplicaciones web Mozilla (s.f.)

Un servidor web basado en *Flask* (en *Python*) actúa como *backend*, mientras que el *frontend* es una interfaz web dinámica (*HTML/JavaScript*) que se ejecuta en el navegador del usuario. Los componentes principales son:

- **Backend Flask:** es la aplicación del lado del servidor que carga el modelo de aprendizaje automático, maneja las solicitudes entrantes, realiza el procesamiento de datos (incluyendo consultas a la base de datos) y devuelve respuestas.
  - **Modelo de Aprendizaje Automático:** se trata de un modelo de clasificación binaria previamente entrenado y almacenado en un archivo `.joblib`. El *backend* lo carga en memoria al iniciar la aplicación y lo emplea para predecir un resultado binario ("degenera" o "no degenera") a partir de las características de entrada proporcionadas por el usuario.
  - **Base de Datos MySQL y Módulo Consultor:** una base de datos relacional *MySQL* almacena los datos relevantes, accesibles a través de una vista denominada `consultor`. El *backend Flask* consulta esta base de datos y envía los resultados al *frontend*, donde se muestran en una tabla *HTML*.
  - **Frontend HTML/JavaScript:** la interfaz de usuario está construida con *HTML* (formularios) y reforzada con un script del lado del cliente (`main.js`). Este script genera dinámicamente los campos del formulario, obtiene datos de referencia desde la *API* del *backend* para rellenar campos *selects*, envía las entradas del usuario para la clasificación y muestra los resultados binarios (0 o 1, interpretados como "No degenera" o "Degenera") al usuario sin recargar la página. Sus responsabilidades incluyen:
    - **Obtención de Metadatos para los Campos del Formulario:** al cargarse la página, el script realiza solicitudes `GET` a rutas bajo `/api/` para obtener los datos necesarios que permiten construir el formulario dinámicamente. Esto garantiza que cualquier cambio en los datos subyacentes (por ejemplo, nuevas edades de *ERG*) se refleje automáticamente en la interfaz, sin necesidad de modificar el código *HTML*. El uso de *AJAX* permite que la página solicite estos datos de forma asíncrona y actualice el *DOM* al recibir la información Zellman (2021).
-

- **Manejo de la Entrada del Usuario y Envío del Formulario:** el script asocia un botón específico de "Clasificar". Recoge todos los valores introducidos por el usuario en los campos y los envía al *backend* mediante una solicitud POST al endpoint `/clasificar`.
- **Comunicación Asíncrona:** la comunicación entre `main.js` y el *backend Flask* se realiza mediante la *API Fetch*, utilizando JSON o datos de formulario como formato de intercambio.
- **Actualización de la UI con los Resultados:** una vez que se reciben los datos JSON, el script actualiza la página web para mostrar el resultado de la clasificación al usuario. La conversión del resultado numérico a una etiqueta de texto se realiza en el cliente para mayor flexibilidad.
- **Errores:** el *JavaScript* también puede manejar casos en que el servidor devuelve un error o la solicitud de red falla, utilizando bloques `catch` en la promesa de `fetch` o verificando los códigos de estado en la respuesta. En una implementación robusta, si ocurre un error (como una entrada faltante o un error del servidor), se mostraría un mensaje de error al usuario. Esto garantiza que la aplicación falle de manera controlada e informe al usuario del problema.
- **Consultor:** la página incluye una estructura de tabla *HTML* con un `<thead>` predefinido para los nombres de columna. Una función *JavaScript* (ubicada en `main.js` u otro archivo) se encarga de obtener los datos del *backend*. Al recibirlos (en formato JSON), el script genera dinámicamente las filas (`<tr>`) y celdas (`<td>`) para cada registro, insertándolas en el `<tbody>` correspondiente. Usando las claves del diccionario JSON, los valores se alinean correctamente con las columnas de la tabla. Esta actualización dinámica garantiza que la tabla refleje siempre el estado más reciente de la base de datos.

Esta arquitectura de alto nivel garantiza un flujo claro: el usuario interactúa con el *frontend* (en su navegador), el cual se comunica con el *backend Flask* mediante solicitudes *HTTP*. El *backend* orquesta las predicciones del modelo de *ML* y las consultas a la base de datos, y responde al cliente, que a su vez actualiza la interfaz según corresponda. En la Figura 5.24

---

se puede observar cómo quedó la página web.

GENERO	ESCOTÓPICO MAX A-W		ESCOTÓPICO MAX B-W		FOTÓPICO MAX A-W
<div>Femenino Masculino Sin especificar</div>	<input type="text" value="OD"/>	<input type="text" value="OI"/>	<input type="text" value="OD"/>	<input type="text" value="OI"/>	<input type="text" value="OD"/>

**Clasificar**  
Resultado

Universidad de Alicante, Ingeniería Biomédica 2025

**Figura 5.24:** Vista final de la interfaz web desarrollada. La aplicación permite al usuario introducir los inputs a través de un formulario dinámico.

### 5.7.2 Integración del modelo en web

La ruta `/clasificar` está diseñada para gestionar solicitudes *HTTP POST* enviadas desde el formulario del *frontend*. Esta espera recibir todos los valores necesarios para que el modelo pueda realizar una predicción.

Una vez recibida la petición, *Flask* recupera los datos enviados usando el objeto `request`. Dado que en esta implementación se utiliza *FormData* (en lugar de *JSON*), los datos se acceden a través de `request.form`, utilizando el nombre de cada campo.

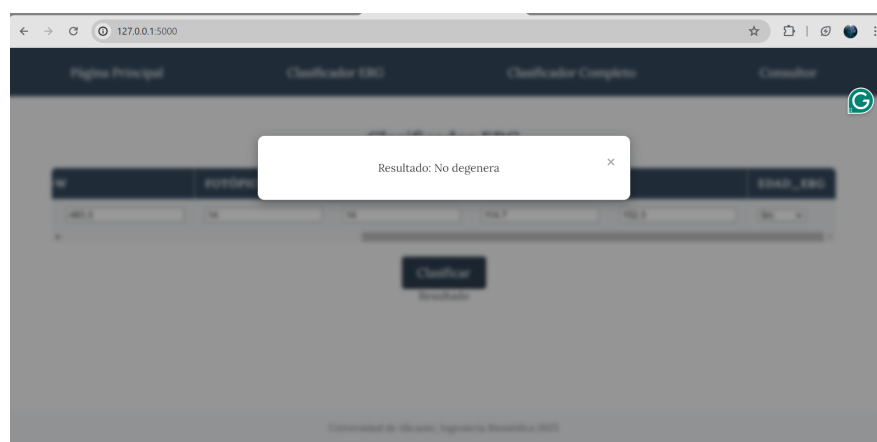
El *backend* lleva a cabo el preprocesamiento de los datos, lo que puede incluir:

- convertir textos a valores numéricos,
- codificar variables categóricas,
- y reorganizar las entradas en el formato esperado por el modelo (por ejemplo, un *array* o un *DataFrame* con el mismo orden de características con el que fue entrenado).

Una vez preparados, los datos se pasan al modelo mediante `model.predict()`, que devuelve una predicción binaria (0 o 1). El resultado se empaqueta en formato *JSON* y se envía al



*frontend* utilizando la función `jsonify()` de *Flask*. Véase en la Figura 5.25.



**Figura 5.25:** Resultado de una clasificación binaria. La interfaz muestra si el caso ingresado presenta o no degeneración

En cuanto a la funcionalidad del módulo *consultor*, el *backend* define una ruta (`/consultor`) que se conecta con la base de datos *MySQL*. Al invocarse mediante una solicitud `GET`, esta ruta ejecuta una consulta para recuperar los registros relevantes.

Antes de enviar los datos al cliente, el servidor asegura que las columnas coincidan con el orden definido en el `<thead>` de la tabla *HTML*. Los datos se devuelven en formato `JSON` para que el *frontend* pueda construir dinámicamente las filas de la tabla. Se presenta el resultado final de la vista *consultor* en la figura 5.26

ID_ANIMAL	SEXO	OD_1	OI_1	OD_2	OI_2	OD_3	OI_3	OD_4	OI_4	OD_5	OI_5	EDAD_OPTO
11696	Masculino	157.8	172.7	366	421.7	5.5	10.7	101	164.6	37.095	182.909	12
11700	Masculino	98.6	150.8	323.9	374.7	4.8	6.1	93.8	108.1	35.274	173.909	12
11702	Feminino	110.9	158.8	332.2	503	9.5	10.2	87.1	150.7	34.915	179	12
11703	Feminino	187.8	127.8	468.9	399	8.1	8.5	140.5	116.6	36	172.97	12
11828	Sin especificar	77.4	60.9	270.6	308	7.5	6.2	67.6	153.6	0	0	0
11928	Masculino	51.1	73	294.3	333.2	0	0	0	0	0	0	0
11929	Masculino	247	197.5	568.7	483.6	34.5	13.1	169.2	165	0	0	6
11932	Masculino	189.1	132.1	436.7	300.6	13.4	9.5	177.1	125.8	0	0	9

**Figura 5.26:** Vista final de la interfaz web en la vista consultor



## 6 Discusión y Líneas futuras

Este Trabajo de Fin de Grado se enmarca en un contexto altamente innovador. No existen estudios previos que apliquen modelos de inteligencia artificial al análisis de datos de degeneración retiniana en ratones genéticamente modificados, lo que convierte a este proyecto en una aproximación pionera dentro de su campo.

A diferencia de los trabajos vistos en el estado del arte que usaban redes convolucionales porque estaban trabajando con imágenes de OCT, en este proyecto se ha usado finalmente la arquitectura XGBoost. Además, no tenían un problema únicamente de clasificación binaria y trabajaban con unos datos con muchas más instancias. Sin embargo, sí que conseguían un rendimiento de los modelos muy bueno como también ocurre en este caso.

En su estadio inicial, el proyecto se enfrentó a diversas limitaciones propias de los estudios experimentales con bases de datos reducidas y poco estructuradas. La base de datos original no estaba diseñada para su uso en aprendizaje automático: presentaba información dispersa, variables no estandarizadas, numerosos valores perdidos y ausencia de una variable objetivo clara. A pesar de ello, se logró consolidar un conjunto de datos funcional y equilibrado, lo cual representa un logro significativo en términos de ingeniería de datos.

### **Entre las principales ventajas del proyecto destacan:**

- La creación de una base de datos propia, estructurada y reutilizable.
- La aplicación de un modelo de clasificación de alta precisión (*XGBoost*, con una precisión de 0.9044 evaluada mediante validación cruzada y de 1.000 mediante la partición *train/test*), con capacidad real de ser aplicado en un entorno de laboratorio.
- La integración final de dicho modelo en una aplicación web funcional, lo que permite su uso por parte de profesionales del grupo de investigación.

**No obstante, también se han identificado limitaciones:**

- El tamaño reducido del conjunto de datos compromete la generalización de algunos modelos, especialmente de las redes neuronales profundas, que no lograron superar un rendimiento estable.
- El sistema aún no permite editar ni actualizar registros de la base de datos de forma dinámica desde la aplicación.
- No se implementaron modelos autoincrementales que puedan aprender a medida que se incorporan nuevos casos.

**En cuanto a mejoras futuras, se propone:**

- Añadir funcionalidades en la aplicación web, como edición y visualización avanzada de los datos.
- Incrementar el número de ratones registrados en la base de datos para mejorar la robustez del modelo.
- Explorar modelos autoincrementales que puedan actualizarse continuamente con nuevos datos sin necesidad de reentrenamiento completo.
- Entrenar el modelo directamente desde la base de datos con nuevos *inputs* y permitir un flujo automatizado de mejora continua.

En resumen, este proyecto representa un primer paso sólido hacia la automatización del análisis clínico en modelos animales, y sienta las bases para futuras investigaciones que puedan extrapolarse al ámbito humano.

---

## 7 Costes

En este apartado se recoge una aproximación al presupuesto requerido para la realización del presente Trabajo de Fin de Grado. La estimación contempla los recursos técnicos utilizados, así como el tiempo dedicado al proyecto.

Concepto	Cantidad	Coste unitario	Coste total
Horas de trabajo del ingeniero	400 h	15 €/h	6 000 €
Equipo informático	1 unidad	700 €	700 €
<b>Total estimado</b>			<b>6 700 €</b>

**Tabla 7.1:** Estimación de costes técnicos y de personal para el desarrollo del proyecto.

Esta valoración permite ofrecer una idea general del esfuerzo técnico y económico que supondría replicar este proyecto en un entorno profesional o de investigación aplicada.



## 8 Conclusiones

- Tras un trabajo de experimentación se ha conseguido aplicar un modelo de inteligencia artificial para clasificar el estado de degeneración retiniana en ratones modificados genéticamente.
- Se ha desarrollado una base de datos funcional con variables extraídas de pruebas experimentales reales, que también se usan en la práctica clínica.
- Con las variables de ERG y el sexo, varios modelos alcanzaron una precisión de 0.7333 aplicando técnicas como *Bootstrap Resampling* (por ejemplo, *SVM*, *MLP*, *DNN*).
- Usando todas las variables del *DataSet* una vez preprocesado, el modelo *XGBoost* alcanzó una precisión media de 0.9044 mediante validación cruzada, y una precisión de 1.000 en la partición *train/test*.
- Las redes neuronales no ofrecieron mejoras significativas, por lo que se descartaron como modelo final.
- La aplicación web es funcional y permite clasificar nuevos casos de forma automática.





## Bibliografía

- Akpinar, M. H., Sengur, A., Faust, O., Tong, L., Molinari, F., y Acharya, U. R. (2024). Artificial intelligence in retinal screening using oct images: A review of the last decade (2013–2023). *Computer Methods and Programs in Biomedicine*, 254. Descargado de <https://doi.org/10.1016/j.cmpb.2024.108253> doi: 10.1016/j.cmpb.2024.108253
- Albertos-Arranz, H., Sánchez-Sáez, X., Martínez-Gil, N., Pinilla, I., Coco-Martin, R. M., Delgado, J., y Cuenca, N. (2021). Phenotypic differences in a prph2 mutation in members of the same family assessed with oct and octa. *Diagnostics*, 11(5). Descargado de <https://doi.org/10.3390/diagnostics11050777> doi: 10.3390/diagnostics11050777
- Analytics Vidhya. (2023, January). *Gradient descent vs backpropagation: What's the difference?* Descargado de <https://www.analyticsvidhya.com/blog/2023/01/gradient-descent-vs-backpropagation-whats-the-difference/> (Recuperado el 19 de abril de 2025)
- Autor desconocido. (s.f.). *Anatomía del ojo*. Descargado de <https://1library.co/article/anatom%C3%ADa-del-ojo-%C3%ADndice.6qmv57wq> (Recuperado el 25 de abril de 2025)
- Balyen, L., y Peto, T. (2019). Promising artificial intelligence-machine learning-deep learning algorithms in ophthalmology. *Asia-Pacific Journal of Ophthalmology*, 8(3), 264–272. Descargado de <https://doi.org/10.22608/APO.2018479> doi: 10.22608/APO.2018479
- Carmona, E. J. (2016). *Tutorial sobre máquinas de vectores soporte (svm)*. Descargado de <https://www.researchgate.net/publication/263817587>
- Chawla, N. V., Bowyer, K. W., Hall, L. O., y Kegelmeyer, W. P. (2002). Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16.

- Chen, T., y Guestrin, C. (2016). Xgboost: A scalable tree boosting system. En *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*.
- Cortes, C., y Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3), 273–297.
- Edreira, D. F., Blanco, J. L., Galdo, B. C., García, V. M., y Sierra, A. P. (2021). La inteligencia artificial como herramienta para la gestión y explotación de datos, informaciones y conocimientos biomédicos en entornos “big data” en la nube. *I+S: Revista de la Sociedad Española de Informática y Salud*, 143, 30–39.
- Efron, B. (1979). Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1), 1–26. doi: 10.1214/aos/1176344552
- Fiuza Pérez, M. D., y Rodríguez Pérez, J. C. (2000). La regresión logística: una herramienta versátil. *Nefrología*, 20(6), 495–500. Descargado de <https://www.revistanefrologia.com/es-la-regresion-logistica-una-herramienta-articulo-X0211699500035664>
- García, I., Dirección, P., Joaquim, J., Ramos, M., y Hortas, M. O. (2023). *Desarrollo de un método de segmentación automática de la retina para la detección de enfermedades neurodegenerativas mediante geometric deep learning*.
- GeeksforGeeks. (s.f.). *How to handle overfitting in pytorch models using early stopping*. Descargado de <https://www.geeksforgeeks.org/how-to-handle-overfitting-in-pytorch-models-using-early-stopping/> (Recuperado el 12 de abril de 2025)
- Hahnloser, R., Sarpeshkar, R., Mahowald, M., Douglas, R., y Seung, H. (2000). Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405, 947–951. doi: 10.1038/35016072
- Heidelberg Engineering. (s.f.). *Spectralis oct – the modular imaging platform*. Descargado de <https://business-lounge.heidelbergengineering.com/us/en/products/spectralis/spectralis/> (Recuperado el 25 de abril de 2025)
- Hemanth, J., Fernando, X., Lafata, P., y Baig, Z. (s.f.). *International conference on intelligent data communication technologies and internet of things (icici) 2018 lecture notes on data*
-

- engineering and communications technologies 26*. Descargado de <http://www.springer.com/series/15362>
- Hernández, J., Ma, O., Ramírez, J., Cèsar, Q., y Ramírez, F. (2004). *Introducción a la minería de datos*.
- IBM. (s.f.). *Xgboost: Potenciando el aprendizaje automático con árboles de decisión*. Descargado de <https://www.ibm.com/es-es/think/topics/xgboost> (Recuperado el 20 de abril de 2025)
- Instead Technologies. (s.f.). *Argos: Un sistema para evaluar la función visual en pequeños animales de investigación*. <https://instead-technologies.com/>. (Recuperado el 28 de marzo de 2025)
- Izaurieta, F., y Saavedra, C. (s.f.). *Redes neuronales artificiales*. (s.f.)
- Krishnamurthy, B. (2024, febrero). *An introduction to the relu activation function*. (Recuperado de <https://builtin.com/machine-learning/relu-activation-function>)
- La Máquina Oráculo. (s.f.). *Neuronas de mcculloch y pitts*. Descargado de <https://lamaquinaoraculo.com/deep-learning/el-modelo-neuronal-de-mcculloch-y-pitts/> (Recuperado el 10 de abril de 2025)
- Le, V. A., y Dik, M. (2024). Topology-preserving scaling in data augmentation. *arXiv preprint, arXiv:2411.19512*.
- Lledó Riquelme, M., y Cuenca Navarro, E. C. M. N. (2010). Transducción visual. *Annals d'Oftalmologia*, 18(3), 130–136.
- López, J. S., y Docampo, G. (1964). Electrorretinografía de la arterioesclerosis retino-coroidea. *Archivos de la Sociedad Oftalmológica Hispano-Americana*, 24(5), 439–449.
- Mares, V., Nehemy, M. B., Bogunovic, H., Frank, S., Reiter, G. S., y Schmidt-Erfurth, U. (2024). Ai-based support for optical coherence tomography in age-related macular degeneration. *International Journal of Retina and Vitreous*, 10(1). Descargado de <https://doi.org/10.1186/s40942-024-00549-1> doi: 10.1186/s40942-024-00549-1
-

- 
- Mozilla. (s.f.). *Sending form data. mdn web docs*. Descargado de [https://developer.mozilla.org/en-US/docs/Learn\\_web\\_development/Extensions/Forms/Sending\\_and\\_retrieving\\_form\\_data](https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Forms/Sending_and_retrieving_form_data) (Recuperado el 5 de mayo de 2025)
- OcuScience. (s.f.). *Hmserg: Un sistema para la evaluación electrofisiológica de la visión en ratones*. <https://ocuscience.us/pages/hmserg-lab-system>. (Recuperado el 28 de marzo de 2025)
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., y Duchesnay, . (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12, 2825–2830.
- Peeters, M. H. C. A., Khan, M., Rooijakkers, A. A. M. B., Mulders, T., Haer-Wigman, L., Boon, C. J. F., ... Collin, R. W. J. (2021). Prph2 mutation update: In silico assessment of 245 reported and 7 novel variants in patients with retinal disease. *Human Mutation*, 42(12), 1521–1547. Descargado de <https://doi.org/10.1002/humu.24275> doi: 10.1002/humu.24275
- Rice, S. O. (1944). Mathematical analysis of random noise. *The Bell System Technical Journal*, 23(3), 282–332.
- Ruiz-Pastor, M. J., Sánchez-Sáez, X., Kutsyr, O., Albertos-Arranz, H., Sánchez-Castillo, C., Ortuño-Lizarán, I., ... Cuenca, N. (2023). Prph2 knock-in mice recapitulate human central areolar choroidal dystrophy retinal degeneration and exhibit aberrant synaptic remodeling and microglial activation. *Cell Death and Disease*, 14(11). Descargado de <https://doi.org/10.1038/s41419-023-06243-8> doi: 10.1038/s41419-023-06243-8
- Rösch, S., Johnen, S., Müller, F., Pfarrer, C., y Walter, P. (2014). Correlations between erg, oct, and anatomical findings in the rd10 mouse. *Journal of Ophthalmology*. Descargado de <https://doi.org/10.1155/2014/874751> doi: 10.1155/2014/874751
- Shorten, C., y Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*.
-

- 
- Spectralis. (s.f.). *Spectralis® oct: combinación de imagen de fondo con láser de escaneo y oct simultáneo*. <https://business-lounge.heidelbergengineering.com/de/de/products/spectralis/>. (Recuperado el 28 de marzo de 2025)
- Taud, H., y Mas, J. F. (2018). Multilayer perceptron (mlp). En (pp. 451–455). doi: 10.1007/978-3-319-60801-3\_27
- Thebault, S. (2011). The retinal pigment epithelium as a blood-retinal barrier component: Involvement in diabetic retinopathy. *Revista Digital Universitaria*, 12(3). Descargado de <http://www.revista.unam.mx/vol.12/num/art31/index.html>
- Tuzsuz, D. (2022, julio). *Sigmoid function*. (Recuperado de <https://www.learndatasci.com/glossary/sigmoid-function/>)
- Using formdata objects*. (s.f.). Descargado de [https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest\\_API/Using\\_FormData\\_Objects](https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest_API/Using_FormData_Objects) (Recuperado de MDN Web Docs)
- Zellman, A. (2021, February). *Build a javascript front end for a flask api*. Descargado de <https://realpython.com/flask-javascript-frontend-for-rest-api/>
-