UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

DIAGNÓSTICO AUTOMATIZADO DE GLAUCOMA MEDIANTE HERRAMIENTAS DE MACHINE LEARNING APLICADAS A DATOS DE PUPILOMETRÍA CROMÁTICA

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO

MANUEL ANTONIO ZAMORANO CANALES

PROFESOR GUÍA:
CARLOS NAVARRO CLAVERIA

PROFESOR CO-GUÍA:
IVÁN PLAZA ROSALES

COMISIÓN:
ANDRÉS CABA RUTTE

SANTIAGO DE CHILE
2024

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: MANUEL ANTONIO ZAMORANO CANALES
FECHA: 2024
PROF. GUÍA: Carlos Navarro Claveria
PROF. CO-GUÍA: Iván Plaza Rosales

DIAGNÓSTICO AUTOMATIZADO DE GLAUCOMA MEDIANTE HERRAMIENTAS DE MACHINE LEARNING APLICADAS A DATOS DE PUPILOMETRÍA CROMÁTICA

El glaucoma es una enfermedad ocular crónica que daña el nervio óptico y constituye la principal causa de ceguera irreversible a nivel mundial, afectando al 3.54 % de la población global entre 40 y 80 años y al 2 % de la población en Chile. Su diagnóstico tardío, debido a la progresión lenta y la ausencia de síntomas en etapas tempranas, resalta la necesidad de técnicas diagnósticas más precisas. La pupilometría cromática, que analiza la respuesta pupilar a estímulos lumínicos, surge como una herramienta prometedora para detectar disfunciones tempranas asociadas al glaucoma. Este trabajo tuvo como objetivo desarrollar una herramienta de diagnóstico automatizado basada en *machine learning*, utilizando datos de pupilometría cromática.

El estudio se realizó con un sistema de pupilometría cromática diseñado con un registro comercial (*PupilLab*) y un sistema de estimulación lumínica propio, aplicando estímulos en formatos *flash* (1 segundo a 250 lux) y rampa (1 minuto, con aumento a 250 lux en 20 segundos) en colores rojo, verde, azul y blanco. Participaron 40 individuos, divididos en 20 pacientes con glaucoma y 20 controles sanos. Se preprocesaron los datos mediante segmentación de señales y filtros, y se extrajeron características estadísticas relevantes, complementadas con técnicas de aumento de datos.

Se probaron modelos de *machine learning* (*XGBoost*) y *deep learning* (FCN y GRU). Para *machine learning*, se evaluaron distintos largos de ventanas de segmentación y tres enfoques experimentales: caso base, selección de ventanas y aislamiento de estímulos. Los mejores resultados se obtuvieron con estímulos tipo rampa, aplicando filtrado y ventanas de 150 muestras de largo, alcanzando un 79.17 % de *accuracy*. Sin embargo, en promedio, los estímulos tipo *flash* generaron resultados más consistentes. Por otro lado, los modelos de *deep learning* no alcanzaron un rendimiento óptimo debido a la limitada cantidad de datos, su alta variabilidad y la falta de generalización.

Este trabajo demuestra que es posible desarrollar herramientas diagnósticas efectivas basadas en pupilometría cromática, sentando las bases para su perfeccionamiento y eventual aplicación clínica.

"Aprender a volar no es fácil. Aprender a caer tampoco.
Pero vivir en el suelo aburre"

Liniers

Agradecimientos

Este trabajo no hubiera sido posible sin las personas que más amo y admiro, que me han otorgado a lo largo de mi vida las cualidades y herramientas que poseo, formándome como la persona que soy hoy. Gracias mamá, papá y tía Titi por toda la sabiduría, el apoyo y amor incondicional que me han dado toda la vida. Gracias hermano, Tuto, por estar siempre a mi lado, escucharme y defenderme. A Anita, por el cariño y el cuidado que me ha dado todos estos años, y a mi Weli, por el amor de abuela. Gracias Anto, por el apañe, la ayuda, el cariño y la paciencia que has tenido conmigo. A todos mis cabros del colegio, mis grandes amigos, mis hermanos, por las risas, los momentos, el apoyo y la compañía en todo momento, ya sean buenos o malos. A mis amigos de la U, por adoptarme e incluirme en nuestros últimos años de estudio y hacerlos mil veces más llevaderos. A mi maestro, por enseñarme a confiar en mí mismo, pararme y seguir peleando. A mis profesores, por las enseñanzas y por confiar en mi trabajo. Y a mi Tío Juan por enseñarme lo que es la resiliencia, y que donde sea que esté, sé que me está sonriendo.
Sin todos ustedes no sería yo. Gracias por todo y ser parte de mi vida.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Identificación y Formulación del Problema

El glaucoma es una enfermedad ocular crónica que daña la cabeza del nervio óptico, punto de salida de las fibras de las células ganglionares de la retina (CGR) [1]. Esta patología es una creciente preocupación en la salud pública a nivel mundial debido a su gravedad y prevalencia. El glaucoma representa la principal causa de ceguera irreversible en todo el mundo [2], afectando a aproximadamente el 3.54 % de la población entre 40 y 80 años [3]. En Chile, esta enfermedad ocupa el primer lugar como causa de ceguera irreversible, afectando al 2 % de la población chilena [4].

La pérdida del campo visual que produce el glaucoma, se caracteriza por ser lenta, indolora y progresiva, haciendo que su diagnóstico sea desafiante. Por lo general, el glaucoma progresa sin síntomas evidentes y puede detectarse cuando ya se ha dañado hasta un 40 % de las CGR [5]. Esta falta de síntomas notorios y la ausencia de una estrategia efectiva para identificar a los pacientes en etapas tempranas de la enfermedad, son desafíos importantes en el manejo de esta patología [4].

En este contexto, se ha despertado un interés creciente en la búsqueda de métodos de detección temprana del glaucoma que sean rentables y de fácil implementación [4]. Una de las áreas prometedoras de investigación es el estudio de la respuesta pupilar a la luz cromática, la cual se centra en la disfunción que se da en la dinámica pupilar al presentar esta enfermedad [6].

Dado el contexto anterior, el presente trabajo de título, se enfoca en el análisis de datos obtenidos a través de un *setup* experimental de pupilometría cromática. Los datos recopilados mediante esta técnica fueron analizados utilizando algoritmos de *Machine Learning* para identificar patrones asociados con el Glaucoma.

1.2. Objetivos del Trabajo de Título

El presente Trabajo de Título tiene como objetivo general desarrollar una herramienta basada en técnicas de *Machine Learning* para diagnosticar el Glaucoma a través de datos obtenidos de pupilometría cromática.

Entre los objetivos específicos, se encuentran:

- Optimizar la preparación de los datos. Esto implica limpiar los datos provistos, aplicar filtros relevantes, seleccionar y organizar la información de manera estructurada, y generar un *dataset* completo y consistente.
- Describir y transformar la data mediante la extracción de características, experimentando con diversas transformaciones, incluyendo la implementación de técnicas de aumento de datos y selección de características, para evaluar su impacto en el rendimiento de los modelos.
- Diseñar y entrenar modelos basados en técnicas de *Machine Learning* y *Deep Learning*, explorando diferentes configuraciones y casos de los datos, con el objetivo de maximizar la precisión y robustez de las predicciones.
- Evaluar, a través de los resultados obtenidos por los modelos predictivos, cuáles estímulos lumínicos y características extraídas contribuyen de manera más efectiva al diagnóstico de glaucoma.

Capítulo 2

Marco Teórico y Estado del Arte

2.1. Marco Teórico

2.1.1. Pupilometría Cromática

La pupilometría cromática es una técnica utilizada para evaluar la respuesta pupilar a estímulos lumínicos, que se basa en la medición de parámetros como la morfología, tamaño y reactividad de la pupila, siendo estos exámenes comunes en la práctica clínica oftalmológica y neurológica [7]. Esta técnica se ha vuelto fundamental en la detección de funciones pupilares anormales, ya que la pupilometría a través de la respuesta pupilar a la luz (PLR) y el periodo inmediato post-estímulo (PIPR) ha demostrado ser una herramienta efectiva [6].

Las anomalías de la pupila, como cambios en el tamaño, la forma y la respuesta a los estímulos luminosos, están estrechamente relacionadas con una serie de complicaciones, entre ellas neuropatías del nervio óptico, como el glaucoma [6]. Numerosos estudios han documentado que la función pupilar detectada mediante PLR y PIPR se encuentra significativamente alterada en pacientes con glaucoma, lo que subraya la importancia de la pupilometría cromática en la detección temprana de esta enfermedad ocular [6]. Entre las células ganglionares de la retina (RGC) afectadas por el glaucoma, se encuentran las células ganglionares de la retina intrínsecamente fotosensibles (IPRGC) [8], las cuales desempeñan un papel crucial en funciones no visuales, como la respuesta pupilar a la luz (PLR) [6].

Es relevante destacar que la respuesta pupilar (PLR y PIPR), se ve afectada en todas las etapas del glaucoma, desde las tempranas hasta las avanzadas, según han concluido diversos estudios [6].

2.1.2. Mirada general a los conceptos de *Machine Learning*

El *Machine Learning*, o Aprendizaje Automático, es una rama de la inteligencia artificial que se centra en el desarrollo de algoritmos y modelos estadísticos que permiten a los sistemas informáticos aprender y mejorar su rendimiento en tareas específicas a partir de la experiencia adquirida a través de datos [9]. Su objetivo fundamental es capacitar a las máquinas para analizar grandes volúmenes de datos y utilizar modelos estadísticos para identificar patrones en estos, lo que les permite producir resultados o tomar decisiones sin la necesidad de instrucciones explícitas [9]. Estos resultados suelen tener asociada una probabilidad de corrección o un grado de confianza.

Se pueden clasificar los métodos de *Machine Learning* en dos categorías principales:

1. **Aprendizaje Supervisado:** en este enfoque, los algoritmos aprenden a resolver problemas utilizando datos de entrada y salida etiquetados. El objetivo es predecir la salida correcta para nuevos conjuntos de datos.

2. **Aprendizaje No Supervisado:** aquí, los algoritmos intentan descubrir patrones ocultos en datos no etiquetados. Este enfoque es más exploratorio y se utiliza para tareas como la segmentación de datos y la detección de anomalías.

Por otra parte, el proceso de creación de soluciones de *Machine Learning*, generalmente, implica dos tareas principales:

- Seleccionar y limpiar un conjunto de datos que se utilizarán para entrenar el modelo.

- Seleccionar un algoritmo o modelo de *Machine Learning* adecuado para el problema específico que se está abordando.

Es importante recalcar que las soluciones de *Machine Learning*, generalmente, requieren conjuntos de datos de entrenamiento que contengan varios cientos o miles de puntos de datos, así como suficiente potencia computacional para ejecutarse eficientemente [9].

Además de estos conceptos básicos, el *Machine Learning* abarca una amplia gama de técnicas y algoritmos, incluidos los modelos de aprendizaje profundo, la optimización de hiperparámetros, la validación cruzada, entre otros. Estas técnicas se aplican en una variedad de campos, desde la medicina hasta la industria financiera, para resolver una variedad de problemas, como el reconocimiento de patrones, la clasificación, la regresión y la generación de recomendaciones.

2.1.3. Aprendizaje supervisado

El aprendizaje supervisado, es una técnica del aprendizaje automático (*Machine Learning*), donde un modelo se entrena utilizando un conjunto de datos etiquetados. Cada ejemplo en el conjunto de datos tiene una etiqueta o valor de salida asociado que el modelo intenta predecir. Según Bishop (2006) [10], este enfoque permite al modelo aprender a partir de datos de entrada conocidos y sus correspondientes salidas para realizar predicciones sobre nuevos datos.

Existen dos tipos principales de problemas abordados mediante aprendizaje supervisado:

- **Clasificación:** la tarea consiste en asignar una etiqueta a una entrada dada. Por ejemplo, clasificar correos electrónicos como "*spam*" o "no *spam*" [11].

- **Regresión:** el objetivo es predecir un valor continuo. Un ejemplo típico es la predicción del precio de una casa basado en sus características [12].

Por otro lado, el proceso del aprendizaje supervisado generalmente sigue estos pasos [13]:

1. **Recolección de Datos:** obtener un conjunto de datos etiquetados.

2. **División de Datos:** dividir los datos en conjuntos de entrenamiento y prueba.

3. **Selección de Modelo:** elegir un algoritmo adecuado para la tarea específica.
4. **Entrenamiento:** ajustar el modelo a los datos de entrenamiento.
5. **Evaluación:** evaluar el rendimiento del modelo en los datos de prueba.
6. **Ajuste:** refinar el modelo para mejorar su desempeño.

Existen diversos algoritmos que se utilizan en el aprendizaje supervisado, cada uno con sus propias ventajas y limitaciones:

- **Regresión Lineal:** utilizada para problemas de regresión, establece una relación lineal entre las variables de entrada y la variable de salida [10].
- **Máquinas de Vectores de Soporte (SVM):** utilizadas tanto para clasificación como para regresión, buscan el hiperplano que mejor separa las clases de datos [11].
- **Árboles de Decisión:** construyen un modelo de decisiones basado en las características de los datos, y se utilizan para clasificación y regresión [12].
- **Redes Neuronales:** inspiradas en el funcionamiento del cerebro humano, se aplican en una amplia variedad de tareas, desde clasificación hasta procesamiento de imágenes [10].
- **k-Vecinos Más Cercanos (k-NN):** clasifica los datos basándose en la mayoría de los vecinos más cercanos en el espacio de características [11].

El aprendizaje supervisado es fundamental en muchos campos, debido a su capacidad para realizar predicciones precisas basadas en datos históricos. Entre sus aplicaciones más comunes se encuentran la detección de fraudes, el reconocimiento de voz y la clasificación de imágenes [13].

2.1.4. Vectores de características

En el campo del *Machine Learning*, un vector de características es una representación numérica de las características (o atributos) relevantes de un objeto, caso o instancia que se usa como entrada para un modelo de aprendizaje. Cada elemento en el vector representa una característica específica y su valor correspondiente. La construcción y selección de estos vectores son cruciales para el rendimiento del modelo, ya que deben capturar toda la información relevante de los datos [10].

Un vector de características se puede considerar como una fila de una matriz donde cada columna representa un atributo diferente de los datos [11]. Por ejemplo, en un problema de clasificación de imágenes, los atributos pueden incluir información sobre los píxeles de la imagen, mientras que en un problema de análisis de texto, los atributos pueden representar la frecuencia de ciertas palabras o n-gramas.

La calidad del vector de características impacta directamente en la efectividad del modelo de *Machine Learning*. Un buen vector de características debe ser capaz de maximizar la información relevante mientras minimiza el ruido. Técnicas como la normalización, estandarización, y selección de características se utilizan para optimizar estos vectores y mejorar el

rendimiento del modelo [10].

Las características pueden ser de varios tipos:

- **Numéricas:** por ejemplo, la edad, el ingreso, la temperatura, etc.
- **Categorías:** por ejemplo, colores, tipos de objetos, clasificaciones, etc.
- **Binarias:** indicadores de presencia o ausencia de una característica, como "sí" o "no", "verdadero" o "falso" [14].

Hay diversas técnicas para la extracción de características que varían según el tipo de datos. Por ejemplo, en imágenes, las técnicas pueden incluir la extracción de bordes, la detección de formas, intensidad de color, magnitud del gradiente y muchos más. En el procesamiento del lenguaje natural, pueden incluir la extracción de palabras clave, la transformación de palabras en vectores a través de *embeddings*, largo de sonidos, nivel de ruido, etc [14].

2.1.5. Características estadísticas

En el análisis de señales, las características estadísticas son cruciales para la extracción de información relevante, que puede ser utilizada en modelos de clasificación y predicción. Estas características incluyen medidas como la media, la desviación estándar, la varianza, la curtosis y la asimetría, entre otras. Estas medidas permiten capturar las propiedades esenciales de las señales, facilitando su análisis y procesamiento en diversas aplicaciones [15].

Entre los tipos de características estadísticas, se encuentran:

1. **Media:** es el promedio de los valores de la señal. La media proporciona una medida central de la señal, ayudando a identificar el valor típico en un conjunto de datos. La media se calcula como:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{2.1}$$

   donde N es el número total de muestras y x_i es el valor de la i-ésima muestra.

2. **Desviación estándar:** mide la dispersión de los valores de la señal respecto a la media. Una alta desviación estándar indica que los valores de la señal están ampliamente dispersos alrededor de la media. La desviación estándar se calcula como:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2} \tag{2.2}$$

3. **Mediana:** es el valor central de la señal cuando los datos están ordenados. Si el número de muestras es par, se toma el promedio de los dos valores centrales:

$$\text{Mediana} = \begin{cases} x_{\left(\frac{N+1}{2}\right)} & \text{si } N \text{ es impar,} \\ \frac{x_{\left(\frac{N}{2}\right)} + x_{\left(\frac{N}{2}+1\right)}}{2} & \text{si } N \text{ es par.} \end{cases} \tag{2.3}$$

4. **Valor máximo:** es el valor más grande de la señal, denotado como:

$$\text{Max}(x) = \max_{i=1,\ldots,N} x_i \tag{2.4}$$

5. **Valor mínimo:** es el valor más pequeño de la señal, denotado como:

$$\text{Min}(x) = \min_{i=1,\ldots,N} x_i \tag{2.5}$$

6. **Rango:** es la diferencia entre el valor máximo y el valor mínimo de la señal:

$$\text{Rango}(x) = \text{Max}(x) - \text{Min}(x) \tag{2.6}$$

7. **Rango intercuartil:** es la diferencia entre el tercer cuartil ($Q3$) y el primer cuartil ($Q1$):

$$\text{IQR}(x) = Q3 - Q1 \tag{2.7}$$

8. **Primer cuartil (Q1):** es el valor que deja por debajo de sí el 25 % de los datos, calculado como:

$$Q1 = x_{\left(\frac{N+1}{4}\right)} \tag{2.8}$$

9. **Tercer cuartil (Q3):** es el valor que deja por debajo de sí el 75 % de los datos, calculado como:

$$Q3 = x_{\left(\frac{3(N+1)}{4}\right)} \tag{2.9}$$

10. **Curtosis:** mide la "apuntalidad" de la distribución de los valores de la señal. Una alta curtosis indica una distribución con picos agudos y colas largas. La curtosis se calcula como:

$$\kappa = \frac{N(N+1)}{(N-1)(N-2)(N-3)} \sum_{i=1}^{N} \left(\frac{x_i - \mu}{\sigma}\right)^4 - \frac{3(N-1)^2}{(N-2)(N-3)} \tag{2.10}$$

11. **Asimetría:** mide la simetría de la distribución de los valores de la señal. Un valor de asimetría positivo indica que la distribución tiene una cola más larga hacia la derecha, mientras que un valor negativo indica una cola más larga hacia la izquierda. La asimetría se calcula como:

$$\gamma = \frac{N}{(N-1)(N-2)} \sum_{i=1}^{N} \left(\frac{x_i - \mu}{\sigma}\right)^3 \tag{2.11}$$

Estas características se utilizan ampliamente en diversas aplicaciones. Por ejemplo, en el estudio de McCall et al. [15], se utilizaron características estadísticas obtenidas de sensores inerciales para clasificar actividades humanas, mejorando la precisión de clasificación mediante la selección de características relevantes para cada macro-clase.

En otro estudio, las características estadísticas de señales de ECG se utilizaron para identificar patrones que indican diferentes condiciones cardíacas. La extracción de estas características permitió el desarrollo de modelos predictivos que mejoraron la precisión de diagnóstico [16].

Estas características estadísticas son esenciales para el análisis y la clasificación de señales, proporcionando una base sólida para el desarrollo de modelos predictivos y la mejora del rendimiento de los mismos en diversas aplicaciones.

2.1.6. Árboles de Decisión

Los árboles de decisión son algoritmos de aprendizaje supervisado no paramétricos utilizados tanto para tareas de clasificación como de regresión. Su estructura jerárquica, similar a un árbol, está compuesta por un nodo raíz, ramas, nodos internos (o nodos de decisión) y nodos hoja (nodos terminales) [17], como se puede ver en la figura 2.1. Estos últimos representan los posibles resultados dentro del conjunto de datos analizado.

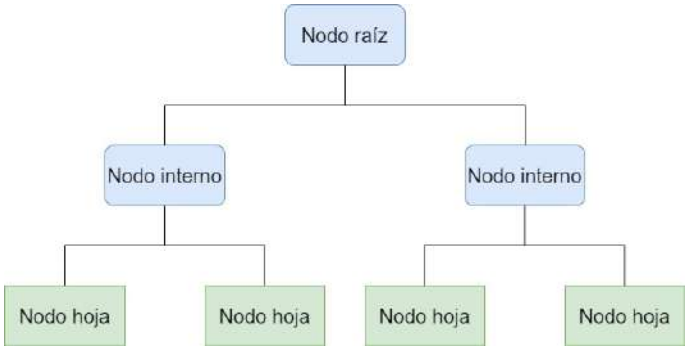

Figura 2.1: Estructura de un árbol de decisión.

El proceso de un árbol de decisión comienza con el nodo raíz, que no tiene ramas entrantes. A partir de este nodo, surgen ramas que conectan con nodos internos, los cuales, a su vez, representan puntos de decisión basados en las características disponibles. Estos nodos realizan evaluaciones que dividen los datos en subconjuntos más homogéneos, y finalmente, las hojas denotan los resultados posibles para las distintas combinaciones de datos [17].

2.1.6.1. Estrategia de Aprendizaje

El aprendizaje en los árboles de decisión sigue una estrategia de *"divide y vencerás"* [17], mediante una búsqueda voraz que identifica los puntos de división óptimos dentro del árbol. Este proceso se repite de manera recursiva desde arriba hacia abajo hasta que la mayoría o la totalidad de los registros son clasificados bajo etiquetas de clase específicas. Sin embargo, la capacidad de clasificar de manera homogénea depende de la complejidad del árbol. Árboles más pequeños tienden a alcanzar nodos hoja más puros, donde los puntos de datos pertenecen a una única clase.

A medida que el árbol crece, se vuelve más difícil mantener esta pureza, lo que a menudo lleva a la fragmentación de datos: cuando quedan muy pocos datos dentro de un subárbol [17]. Este fenómeno puede derivar en un sobreajuste del modelo, donde se ajusta excesivamente a las particularidades de los datos de entrenamiento, pero pierde capacidad de generalización en nuevos datos.

2.1.6.2. Poda y Reducción de Complejidad

Para evitar el sobreajuste, se recurre a técnicas de poda. Este proceso consiste en eliminar ramas que dividen con base en características de baja importancia. Esto reduce la

complejidad del árbol sin sacrificar la precisión del modelo. El ajuste del modelo se evalúa generalmente mediante la técnica de validación cruzada. Además, una manera común de mejorar la precisión de los árboles de decisión es utilizando algoritmos como el de *random forest* [17], donde se construye un conjunto de árboles no correlacionados para generar predicciones más precisas.

Los árboles de decisión son potentes y fáciles de interpretar, pero su tendencia a sobreajustar los datos los hace propensos a perder precisión en nuevas predicciones. Por ello, la simplicidad es una característica clave en su diseño, alineada con el principio de la navaja de Ockham, que sugiere que *"la explicación más simple suele ser la mejor"* [18].

2.1.6.3. Selección del Mejor Atributo en Cada Nodo

Existen varios métodos para determinar el mejor atributo en cada nodo de un árbol de decisión, pero dos de los más utilizados son la ganancia de información y la impureza de Gini. Estos métodos ayudan a evaluar qué tan bien un atributo en particular puede dividir los datos y mejorar la clasificación de las muestras en sus respectivas clases.

2.1.6.3.1. Entropía y Ganancia de Información

La entropía es un concepto tomado de la teoría de la información y se utiliza para medir el nivel de incertidumbre o impureza en un conjunto de datos [17]. Cuanto mayor es la entropía, más desorden hay en los datos. La entropía se define de la siguiente manera:

$$\text{Entropía}(S) = -\sum_{c \in C} p(c) \log_2 p(c)$$

Donde:

- S es el conjunto de datos para el cual se está calculando la entropía.
- C es el conjunto de clases presentes en S.
- $p(c)$ es la probabilidad de que un punto de datos pertenezca a la clase c en el conjunto S.

La entropía puede variar entre 0 y 1. Si todas las muestras pertenecen a la misma clase, la entropía es 0, lo que indica un conjunto de datos perfectamente ordenado. Por otro lado, si los datos están distribuidos equitativamente entre las clases, la entropía alcanza su valor máximo de 1. Al construir un árbol de decisión, el objetivo es seleccionar el atributo que minimice la entropía, ya que esto genera divisiones más puras [17]. La ganancia de información se refiere a la reducción de la entropía después de realizar una división en el árbol, y se calcula como:

$$\text{Ganancia de Información} = \text{Entropía}(S) - \sum_{v \in \text{Valores}(A)} \frac{|S_v|}{|S|} \text{Entropía}(S_v)$$

Donde:

- A es el atributo por el cual se está dividiendo el conjunto de datos.
- Entropía(S) es la entropía antes de la división.
- $\frac{|S_v|}{|S|}$ es la proporción del subconjunto de datos S_v generado por la división.

- Entropía(S_v) es la entropía del subconjunto de datos S_v tras la división.

El atributo que maximiza la ganancia de información es aquel que mejor reduce la impureza de los datos.

2.1.6.3.2. Impureza de Gini

La impureza de Gini mide la probabilidad de clasificar incorrectamente un punto de datos si este fuera etiquetado al azar de acuerdo con la distribución de clases en el conjunto de datos [17]. Al igual que la entropía, si el conjunto de datos es puro (todos los puntos pertenecen a una única clase), la impureza de Gini es 0. La fórmula para calcular la impureza de Gini es:

$$\text{Impureza de Gini} = 1 - \sum_i (p_i)^2$$

Donde p_i es la proporción de puntos de datos que pertenecen a la clase i.

La impureza de Gini es otro criterio útil para decidir las mejores divisiones en un árbol de decisión. Al evaluar las posibles divisiones, el objetivo es seleccionar el atributo que tenga la menor impureza de Gini, ya que este generará subconjuntos de datos más homogéneos [17].

2.1.6.4. Ventajas y desventajas de los Árboles de Decisión

Entre las ventajas, se encuentra [17]:

- **Fácil interpretación:** La lógica booleana y la representación visual de los árboles de decisión los hacen fáciles de comprender y utilizar. La estructura jerárquica que presentan permite identificar con claridad los atributos más relevantes, algo que a veces no es tan evidente en otros algoritmos, como las redes neuronales.
- **Requerimiento de poca o ninguna preparación de datos:** Los árboles de decisión son bastante flexibles en comparación con otros clasificadores. Pueden trabajar con diferentes tipos de datos, ya sean discretos o continuos, y los valores continuos pueden transformarse en categóricos mediante el uso de umbrales. Además, son capaces de manejar valores faltantes, un desafío para otros clasificadores, como el modelo Naïve Bayes.
- **Mayor flexibilidad:** Los árboles de decisión se pueden utilizar tanto para tareas de clasificación como de regresión, lo que los hace más versátiles que algunos otros algoritmos. Otra ventaja es que son insensibles a las relaciones subyacentes entre los atributos; es decir, si dos variables están altamente correlacionadas, el algoritmo seleccionará solo una de ellas para realizar la división.

Entre las desventajas, se encuentra [17]:

- **Propensos al sobreajuste:** Los árboles de decisión muy complejos tienden a ajustarse demasiado a los datos de entrenamiento, lo que dificulta su capacidad para generalizar en nuevos datos. Este problema se puede mitigar mediante la poda (*pruning*). La poda previa (*pre-pruning*) detiene el crecimiento del árbol cuando los datos no son suficientes, mientras que la poda posterior (*post-pruning*) elimina subárboles que no aportan valor una vez que el árbol ha sido construido.
- **Alta varianza:** Pequeñas variaciones en los datos pueden generar árboles de decisión completamente diferentes. Para reducir esta variabilidad, se puede utilizar el método

de *bagging*, que consiste en promediar las estimaciones de múltiples árboles. Sin embargo, este enfoque tiene limitaciones, ya que puede conducir a predictores altamente correlacionados.

- **Mayor costo computacional:** Debido a que los árboles de decisión emplean una búsqueda voraz para identificar los mejores puntos de división, su entrenamiento puede ser más costoso en términos computacionales en comparación con otros algoritmos.

2.1.7. *Gradient Boosting Machines* (GBM)

Gradient Boosting Machines (GBM) se han convertido en algoritmos extremadamente populares, especialmente desde el surgimiento de *XGBoost* en 2016, para tareas de clasificación y regresión [19]. Estos algoritmos combinan múltiples *"weak learners"* o "aprendices débiles", con el fin de mejorar de manera iterativa el rendimiento de los clasificadores o regresores, corrigiendo los errores cometidos en etapas anteriores [20].

El objetivo no es crear clasificadores potentes de inmediato, sino utilizar "aprendices débiles" que se potencian en conjunto [21]. En la mayoría de los casos, los *weak learners* utilizados son árboles de decisión, debido a su simplicidad y versatilidad tanto para tareas de regresión como de clasificación [21]. Se pueden entrenar con una muestra del conjunto total de datos, lo que permite entrenamientos rápidos y eficientes. Al usar árboles de decisión, los problemas asociados con estos, como el sobreajuste, pueden transmitirse al modelo GBM, por lo que es necesario controlar adecuadamente su complejidad [21].

Los árboles de decisión tienden a sobreajustarse cuando se incrementa su profundidad o el número de nodos hijos. Este problema también se presenta en GBM, lo que hace esencial ajustar los hiperparámetros, como la profundidad máxima y el número de nodos, para evitar el sobreajuste.

GBM es un algoritmo de *ensemble*, lo que significa que combina múltiples modelos sencillos para generar un modelo más robusto [21]. Los modelos aditivos son un ejemplo de este tipo de algoritmos, donde la salida de varios modelos independientes se suma para mejorar la predicción:

$$F_M(x) = f_1(x) + ... + f_M(x) = \sum_{m=1}^{M} f_m(x)$$

2.1.7.1. GBM en Clasificación

Para tareas de clasificación, la principal diferencia con respecto a la regresión es la función de pérdida ($\mathcal{L}$) utilizada. En clasificación, se utilizan funciones de pérdida que buscan minimizar el error al predecir una clase en lugar de un valor continuo [21]. Una de las funciones de pérdida más utilizadas es la *cross-entropy* o entropía cruzada [20]:

$$\mathcal{L}_i = -(y_i \log(p_i) + (1 - y_i) \log(1 - p_i))$$

La derivada de esta función es:

$$\frac{\partial \mathcal{L}_i}{\partial \hat{y}} = p_i - y_i$$

Dado que los modelos de clasificación con GBM son esencialmente modelos de regresión, las salidas no pueden ser interpretadas directamente como probabilidades, sino como *scores*, los cuales deben ser transformados. La predicción de y en GBM para clasificación se expresa como:

$$\hat{y} = \log(\text{odds}) = \log\left(\frac{p}{1-p}\right)$$

Para obtener probabilidades finales, se utiliza la función *softmax*:

$$\text{softmax}(\hat{y}) = \frac{1}{1+e^{-\hat{y}}}$$

2.1.7.2. Entrenamiento de GBM

El entrenamiento de GBM se puede visualizar como un proceso iterativo en el cual se realizan ajustes progresivos [21]. La tasa de aprendizaje (ν), también conocida como *shrinkage rate*, es un factor crucial que ajusta el tamaño de las actualizaciones en cada iteración [20]. Esto se refleja en la siguiente ecuación:

$$f_m(x) = f_{m-1}(x) + \nu F_m(x)$$

Dependiendo del problema, se pueden ajustar las funciones de pérdida para mejorar los resultados. Las funciones de pérdida más comunes se presentan en la tabla 2.1.

Tabla 2.1: Ejemplos de funciones de pérdida y sus derivadas

Nombre	**Pérdida**	**Derivada**
Error Cuadrático	$\frac{1}{2}(y_i - f(x_i))^2$	$y_i - f(x_i)$
Error Absoluto	$\lvert y_i - f(x_i)\rvert$	$sgn(y_i - f(x_i))$
Binary Logloss	$\log\left(1 + e^{-y_i f_i}\right)$	$y - \sigma(2f(x_i))$

Finalmente, para evitar el sobreajuste, se puede agregar un término de regularización a la función de pérdida, lo cual penaliza la complejidad del modelo [19]:

$$\mathcal{L}(f) = \sum_{i=1}^{N} \ell(y_i, f(x)) + \Omega(f)$$

donde $\Omega(f)$ es un término de regularización:

$$\Omega(f) = \gamma J + \frac{1}{2}\lambda \sum_{j=1}^{J} w_j^2$$

2.1.7.3. Ventajas y Desventajas de GBM

Dentro de las ventajas, se encuentra [21]:

- Capaz de manejar problemas tanto de regresión como de clasificación.
- Captura relaciones no lineales y características complejas en los datos.
- Funciona con variables numéricas y categóricas.

- Robusto frente a valores atípicos y datos ruidosos.
- Escalable a grandes conjuntos de datos.
- Flexibilidad en la selección de funciones de pérdida.

Dentro de las desventajas, se encuentra [21]:

- Mayor tiempo de entrenamiento y complejidad en comparación con algoritmos más simples.
- Sensible al sobreajuste si no se ajustan correctamente los hiperparámetros.
- Requiere mayor poder computacional.
- Interpretación más compleja debido a la naturaleza de combinación de múltiples árboles.
- Sensible a desequilibrio en las clases de los datos.

2.1.7.4. XGBoost (*Xtreme Gradient Boosting*)

XGBoost es un potente algoritmo de optimización basado en gradientes que utiliza un conjunto de árboles de decisión ensamblados [21]. Su popularidad ha crecido debido a su alto rendimiento y velocidad en diversas tareas de clasificación y regresión [19]. Algunas de sus principales características son [21]:

- **Manejo eficiente de los valores faltantes:** *XGBoost* puede tratar eficientemente los valores faltantes, desviando las muestras hacia una de las ramas por defecto.
- **Soporte para regularización:** El algoritmo incluye mecanismos de regularización L1 (*Lasso*) y L2 (*Ridge*), que ayudan a prevenir el sobreajuste del modelo.
- **Alta flexibilidad:** *XGBoost* permite definir tanto funciones de pérdida personalizadas como funciones de evaluación, lo que lo hace altamente adaptable a diferentes problemas.

XGBoost ha demostrado ser una herramienta de gran utilidad en competiciones de aprendizaje automático, destacándose por su capacidad para gestionar grandes conjuntos de datos, su escalabilidad y su eficacia en la reducción de errores en los modelos predictivos [19].

Algunos de sus hiperparámetros más importantes que lo componen son:

1. ***Learning Rate* (η):** El *learning rate*, o tasa de aprendizaje, es uno de los hiperparámetros más importantes en *XGBoost*. Controla cuánto se ajustan los nuevos árboles que se agregan al modelo durante cada iteración. En otras palabras, el *learning rate* regula la magnitud de las actualizaciones en el modelo. Un valor más bajo de *learning rate* reduce la velocidad de aprendizaje, lo que puede evitar el sobreajuste al forzar al modelo a hacer cambios más pequeños y precisos en cada iteración. Sin embargo, un *learning rate* más bajo también puede requerir más iteraciones para lograr un buen rendimiento. La ecuación que representa este proceso es:

$$f_m(x) = f_{m-1}(x) + \eta \cdot F_m(x)$$

Donde:

- $f_m(x)$ es el nuevo modelo que se ajusta después de la m-ésima iteración.
- $F_m(x)$ es la función de ajuste en la iteración m-ésima.
- η es la tasa de aprendizaje (*learning rate*), que escala el ajuste en cada paso.

2. ***Max Depth*:** El hiperparámetro *max depth* define la profundidad máxima que puede alcanzar cada árbol de decisión en el modelo *XGBoost*, controlando la complejidad del árbol. Una mayor profundidad permite al árbol capturar interacciones más complejas entre las características, pero también aumenta el riesgo de sobreajuste, ya que los árboles muy profundos tienden a ajustar demasiado el modelo a los datos de entrenamiento, afectando su capacidad de generalización a nuevos datos. Una profundidad más baja hace que los árboles sean más simples y eviten el sobreajuste, pero podrían no capturar adecuadamente patrones complejos en los datos.

3. ***Evaluation Metric*:** La *evaluation metric* es la métrica que se utiliza para evaluar el rendimiento del modelo durante el entrenamiento y la validación. *XGBoost* admite diferentes métricas, dependiendo del tipo de problema (clasificación o regresión). Algunas de las métricas comunes son:

   - *Log-loss* (para clasificación binaria): mide la incertidumbre de las predicciones del modelo, penalizando más las predicciones incorrectas con alta confianza.

   $$\mathcal{L}_{\text{log-loss}} = -(y \cdot \log(p) + (1 - y) \cdot \log(1 - p)) \tag{2.12}$$

   Donde y es la etiqueta real y p es la probabilidad predicha.

   - RMSE (*Root Mean Squared Error*) para regresión: mide el error promedio entre las predicciones del modelo y los valores reales.

   $$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \tag{2.13}$$

   La elección de la métrica depende del tipo de tarea. Por ejemplo, para clasificación se puede usar *log-loss*, mientras que para regresión es común usar RMSE. La métrica elegida se optimiza en cada iteración para ajustar el modelo.

2.1.8. Redes neuronales

Las redes neuronales artificiales (RNA) son sistemas computacionales inspirados en la estructura y funcionamiento del cerebro humano. Se definen como redes distribuidas y paralelas de procesadores elementales simples, conocidos como neuronas. Estas redes intentan emular la estructura masivamente paralela del cerebro, con la capacidad de adquirir conocimiento del ambiente a través de un proceso de aprendizaje y almacenar este conocimiento en las conexiones sinápticas entre las neuronas [22].

Las RNA consisten en un conjunto de nodos (neuronas artificiales) conectados por enlaces (sinapsis), cada uno con un peso asociado. Los pesos representan la fuerza de la conexión entre las neuronas, y se ajustan durante el proceso de aprendizaje para mejorar el rendimiento

del modelo [10].

El aprendizaje en una RNA, se lleva a cabo mediante algoritmos que ajustan los pesos sinápticos con el fin de lograr un objetivo específico, como minimizar el error en las predicciones. Uno de los algoritmos más comunes es el de retropropagación (*backpropagation*), que utiliza el descenso de gradiente para actualizar los pesos en función del error cometido por la red [23].

Una característica fundamental de las RNA es su capacidad de generalización, es decir, su habilidad para producir respuestas razonables ante entradas no usadas durante el entrenamiento. Esta capacidad de generalización permite a las RNA aplicar el conocimiento adquirido a situaciones nuevas y no previamente vistas [24].

Una neurona artificial puede ser vista como una unidad procesadora de información con tres elementos básicos [22]:

1. **Pesos Sinápticos:** cada entrada a la neurona está asociada a un peso sináptico, que pondera la importancia de esa entrada. Los pesos se ajustan durante el proceso de aprendizaje. Estos corresponden a los elementos $w_{k1},\ w_{k2},\ ...\ w_{km}$ de la figura 2.2.

2. **Sumador o Combinador Lineal:** la neurona calcula una combinación lineal ponderada de sus entradas. Esto se realiza sumando el producto de cada entrada por su correspondiente peso sináptico. Esta estructura corresponde al elemento $\sum$ de la figura 2.2.

3. **Función de Activación No-Lineal:** la salida del sumador pasa por una función de activación no-lineal, que limita la amplitud de la salida de la neurona. Las funciones de activación más comunes son la sigmoide, la tangente hiperbólica y la ReLU (*Rectified Linear Unit*). Esta función introduce no linealidad en el modelo, permitiendo a la red aprender representaciones complejas [25]. Esta corresponde al elemento $\varphi(\cdot)$ de la figura 2.2.

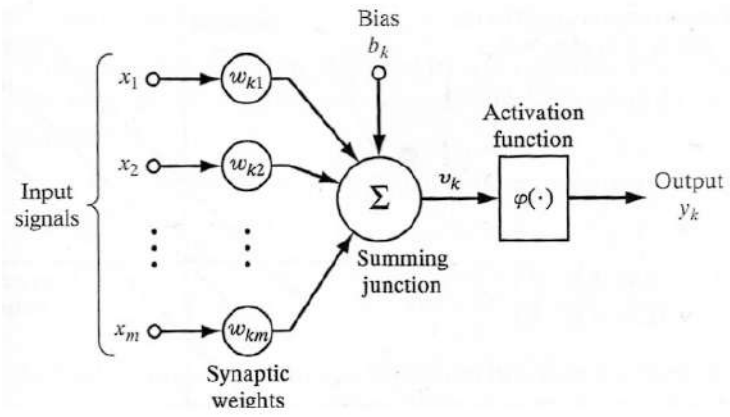

Figura 2.2: Modelo elemental de una neurona artificial.

Las ecuaciones de una neurona son:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \tag{2.14}$$

$$y_k = \varphi(u_k + b_k) \tag{2.15}$$

Donde:

- x_j es la entrada j-ésima.
- w_{kj} es el peso sináptico de la neurona k, asociado a la entrada j-ésima.
- u_k es la salida del combinador lineal.
- b_k es el sesgo o *bias*.
- y_k es la salida no-lineal de la neurona.

Las RNA se utilizan en una amplia variedad de aplicaciones debido a su capacidad para aprender y generalizar. Algunas aplicaciones comunes incluyen:

- **Reconocimiento de Patrones:** las RNA son extremadamente eficaces en tareas como el reconocimiento de voz y la clasificación de imágenes [26].
- **Procesamiento del Lenguaje Natural (NLP):** se utilizan en traducción automática, análisis de sentimientos y generación de texto [27].
- **Predicción y Modelado:** aplicadas en finanzas para la predicción de precios de acciones y en climatología para el pronóstico del tiempo [28].

2.1.9. *Deep Learning*

Deep Learning, o aprendizaje profundo, es una subrama del aprendizaje automático (*Machine Learning*) que se basa en el uso de redes neuronales artificiales con múltiples capas de procesamiento. Estas redes son capaces de aprender representaciones de datos de alto nivel mediante transformaciones no lineales. El término "profundo" se refiere a las numerosas capas que constituyen la red, lo que permite la modelización de patrones complejos y abstractos en los datos [26].

El *Deep Learning* se aplica a una variedad de problemas, entre los que destacan:

- **Reconocimiento de Imágenes:** clasificación y detección de objetos en imágenes, segmentación semántica y reconocimiento facial [29].
- **Procesamiento del Lenguaje Natural (NLP):** tareas como traducción automática, análisis de sentimientos, generación de texto y *chatbots* [30].
- **Reconocimiento de Voz:** conversión de voz a texto, comandos de voz y sistemas de asistencia virtual [31].
- **Juego y Estrategia:** entrenamiento de agentes de inteligencia artificial para juegos complejos como el ajedrez y el Go, donde se superan las capacidades humanas [32].

Algunos conceptos básicos importantes son:

1. **Redes Neuronales Convolucionales (CNN):** utilizadas principalmente en el procesamiento de imágenes, las CNN emplean capas convolucionales para detectar patrones locales en las imágenes [26].

2. **Redes Neuronales Recurrentes (RNN):** adecuadas para datos secuenciales, como el texto y el audio. Las RNN tienen conexiones que forman ciclos, permitiendo que la información persista [33].

3. **Redes de Memoria a Largo Plazo (LSTM):** una variante de las RNN diseñada para aprender dependencias a largo plazo y resolver el problema del desvanecimiento del gradiente [33].

4. **Redes Generativas Antagónicas (GAN)**: compuestas por dos redes neuronales que compiten entre sí para generar datos nuevos y realistas a partir de datos de entrenamiento [34].

El *Deep Learning* ha revolucionado múltiples campos, debido a su capacidad para procesar y entender grandes volúmenes de datos no estructurados. Algunas de las aplicaciones más importantes incluyen:

- **Diagnóstico Médico:** análisis de imágenes médicas para detectar enfermedades como el cáncer y la retinopatía diabética [35].

- **Conducción Autónoma:** sistemas de percepción para vehículos autónomos, incluyendo detección de objetos y navegación [36].

- **Finanzas:** modelos de predicción de mercados, detección de fraudes y análisis de riesgos [37].

- **Creatividad Artificial:** generación de arte, música y diseño a través de algoritmos de aprendizaje profundo [38].

Los avances recientes en *Deep Learning*, incluyen mejoras en arquitecturas de redes, técnicas de regularización y métodos de optimización. Sin embargo, los desafíos persistentes incluyen la necesidad de grandes cantidades de datos etiquetados, el alto costo computacional y la interpretabilidad de los modelos [24].

Deep Learning es una tecnología transformadora que ha ampliado las fronteras del aprendizaje automático. Su capacidad para manejar grandes cantidades de datos y aprender representaciones complejas lo convierte en una herramienta crucial en diversas aplicaciones industriales y científicas. A medida que la investigación avanza, se espera que las técnicas de *Deep Learning* se vuelvan aún más eficientes y accesibles.

2.1.10. *Deep Learning* para la clasificación de series temporales

Las series temporales univariadas, como el tipo más simple de datos de series temporales, representan un punto de partida ideal para estudiar señales temporales [39]. La investigación en aprendizaje de representaciones y clasificación ha revelado numerosas aplicaciones

potenciales en áreas como las finanzas, la industria y el sector sanitario. Si bien las medidas de similitud tradicionales, como el *Dynamic Time Warping* (DTW) y la Distancia Euclidiana (ED), han sido utilizadas durante décadas, los esfuerzos recientes en la ingeniería de características y el diseño de nuevas medidas de distancia han logrado una precisión significativamente mayor en los estándares de clasificación de series temporales [39]. Sin embargo, estos avances suelen venir acompañados de mayores niveles de complejidad y menores niveles de interpretabilidad.

2.1.10.1. *Fully Convolutional Network* (FCN)

Las *Fully Convolutional Networks* (FCN) han demostrado un rendimiento destacado en diversas áreas, como la segmentación semántica de imágenes, la clasificación de series temporales y otros campos relacionados. Este enfoque ha sido ampliamente adoptado debido a su capacidad para ofrecer resultados competitivos en múltiples conjuntos de datos [40].

El diseño básico de una FCN para clasificación de series temporales consta de tres capas convolucionales, una capa de *global pooling* y una capa de clasificación basada en *Softmax* [40], como se muestra en la figura 2.3. La capa *Softmax* se encarga de la clasificación, mientras que las capas convolucionales extraen las características relevantes de los datos. Cada capa convolucional incluye tres elementos fundamentales: una operación de convolución, una capa de normalización por lotes (*batch normalization*) y una activación ReLU, como se describe en la siguiente ecuación [40]:

$$y = W \otimes x + b, \quad s = BN(y), \quad h = ReLU(s) \tag{2.16}$$

donde $\otimes$ representa la operación de convolución, BN es la normalización por lotes y $ReLU$ es la función de activación no lineal.

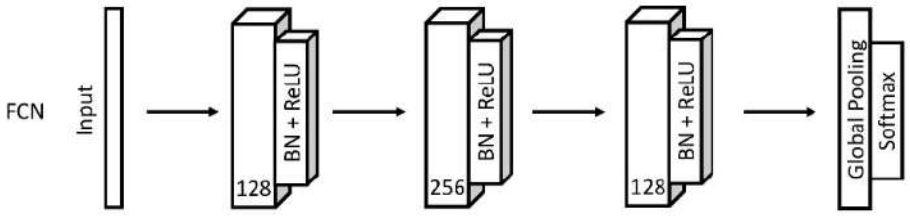

Figura 2.3: Arquitectura de una FCN.

La arquitectura utiliza tres núcleos convolucionales de 1-D con tamaños $\{8, 5, 3\}$, todos con un *stride* de 1. El número de canales por capa es $\{128, 256, 128\}$. Cabe destacar que no se emplean capas de *pooling* entre las convoluciones, lo que permite conservar la resolución temporal completa de las señales y evitar la pérdida de información crítica [40].

Después de las capas convolucionales, se incorpora una capa de *global average pooling* en lugar de una capa completamente conectada (*fully connected layer*). Este enfoque reduce significativamente el número de parámetros de la red, acelera el entrenamiento y mejora la capacidad de generalización del modelo [40]. Finalmente, una capa *Softmax* clasifica las características extraídas por las convoluciones, asignando probabilidades a las distintas clases [40].

Gracias a estas características, las FCN han demostrado ser herramientas eficaces para tareas de clasificación de series temporales, ofreciendo una combinación sólida de precisión, velocidad de entrenamiento y simplicidad arquitectónica.

2.1.11. Redes Neuronales Recurrentes

Una Red Neuronal Recurrente (RNN, por sus siglas en inglés), es un modelo de aprendizaje profundo diseñado para procesar y convertir datos secuenciales en salidas secuenciales específicas. Estos datos secuenciales pueden ser palabras, oraciones o series temporales, donde los componentes interactúan según reglas semánticas y sintácticas complejas [24]. Las RNN son particularmente adecuadas para tareas como la traducción automática, el reconocimiento de voz y la generación de texto, debido a su capacidad para manejar la dependencia temporal entre los datos [41].

Las RNN se distinguen de las redes neuronales tradicionales por su arquitectura en la que existen conexiones recurrentes, lo que les permite mantener una memoria de estados pasados. La estructura básica de una RNN incluye:

1. **Capa de Entrada:** recibe los datos secuenciales. Cada entrada en la secuencia se procesa paso a paso.

2. **Capas Ocultas:** estas capas contienen unidades recurrentes que realizan el procesamiento real. En cada paso de tiempo t, la capa oculta recibe la entrada actual x_t y el estado oculto anterior h_{t-1}, produciendo el estado oculto actual h_t. La actualización del estado oculto se puede describir matemáticamente como:

$$h_t = f(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \tag{2.17}$$

   donde f es una función de activación no lineal, W_{xh} y W_{hh} son matrices de pesos, y b_h es un sesgo (*bias*) [42].

3. **Capa de Salida:** genera la salida basada en el estado oculto actual. La salida y_t se calcula típicamente como:

$$y_t = g(W_{hy}h_t + b_y) \tag{2.18}$$

   donde g es otra función de activación, W_{hy} es la matriz de pesos de la capa de salida, y b_y es el sesgo correspondiente [24].

El principal distintivo de las RNN es su capacidad para "recordar" información a través de sus conexiones recurrentes. Esto les permite utilizar la información de pasos de tiempo anteriores para influir en las predicciones futuras, lo que es esencial para tareas que dependen del contexto histórico.

El aprendizaje en una RNN se realiza mediante el algoritmo de retropropagación a través del tiempo (BPTT, por sus siglas en inglés). Este algoritmo extiende el método de retropropagación utilizado en redes *feedforward* al desenrollar la RNN a través del tiempo y aplicar retropropagación en cada paso de tiempo. Esto permite la actualización de los pesos sinápticos de la red para minimizar el error de predicción [43].

Las RNN tradicionales enfrentan desafíos como el desvanecimiento y el estallido del gradiente, lo que dificulta el aprendizaje de dependencias a largo plazo. Para abordar estos problemas, se han desarrollado variantes como las Redes de Memoria a Largo Plazo (LSTM) y las Unidades de Estado Gated Recurrent (GRU):

1. **LSTM (*Long Short-Term Memory*):** introducidas por Hochreiter y Schmidhuber (1997) [33], las LSTM incluyen "celdas de memoria" y mecanismos de puerta que controlan el flujo de información, permitiendo a la red aprender dependencias a largo plazo.
2. **GRU (*Gated Recurrent Unit*):** introducidas por Cho et al. (2014) [44], las GRU son una simplificación de las LSTM, con menos parámetros pero capacidad similar para capturar dependencias temporales.

Las RNN son fundamentales en muchos campos, debido a su capacidad para modelar datos secuenciales. Algunas de las aplicaciones más destacadas incluyen:

- **Traducción Automática:** las RNN, especialmente en forma de secuencia a secuencia (*seq2seq*), se utilizan para traducir texto de un idioma a otro [41].
- **Reconocimiento de Voz:** modelos de RNN se utilizan para transcribir voz a texto, permitiendo el desarrollo de asistentes virtuales y sistemas de dictado [45].
- **Generación de Texto:** las RNN pueden generar texto de manera coherente basándose en patrones aprendidos de grandes corpus de texto [46].
- **Análisis de Series Temporales:** aplicadas en finanzas y meteorología para la predicción de valores futuros basados en datos históricos [47].

Las Redes Neuronales Recurrentes representan una herramienta poderosa y versátil en el aprendizaje profundo, especialmente adecuadas para el procesamiento de datos secuenciales. Su capacidad para aprender y recordar contextos a través del tiempo las convierte en una tecnología clave en aplicaciones que van desde el procesamiento del lenguaje natural hasta el análisis de series temporales.

2.1.11.1. *Gated Recurrent Units* (GRU)

Las *Gated Recurrent Units* (GRU) son un tipo de red neuronal recurrente (RNN) diseñada para modelar datos secuenciales, introducida por Cho et al. en 2014 [44]. Al igual que las *Long Short-Term Memory* (LSTM), las GRU fueron creadas para abordar el problema del desvanecimiento y explosión del gradiente que afecta a las RNN tradicionales, permitiendo a los modelos capturar dependencias a largo plazo en los datos [24].

Las GRU utilizan un enfoque más simplificado en comparación con las LSTM, lo que las hace más rápidas de entrenar y menos costosas en términos computacionales. Su arquitectura se basa en dos compuertas principales: la compuerta de actualización (*update gate*) y la compuerta de reinicio (*reset gate*), como se muestra en la figura 2.4:

- ***Update Gate:*** Determina cuánto de la información pasada se debe retener para predecir la siguiente salida.
- ***Reset Gate:*** Controla qué parte de la información pasada se debe olvidar, permitiendo al modelo enfocarse en datos recientes si es necesario.

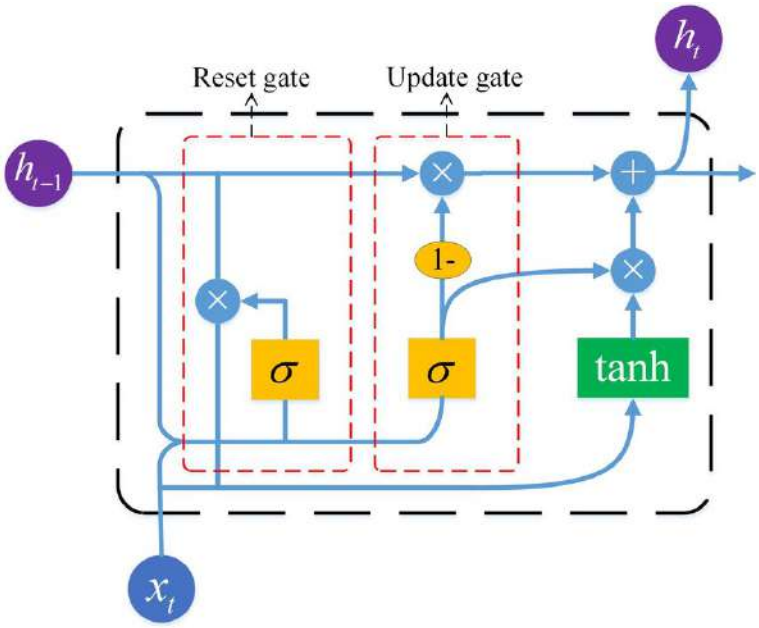

Figura 2.4: Arquitectura de una GRU.

El funcionamiento de las GRU puede representarse mediante las siguientes ecuaciones [44]:

$$z_t = \sigma(W_z \cdot x_t + U_z \cdot h_{t-1} + b_z) \quad \text{(Compuerta de actualización)} \tag{2.19}$$

$$r_t = \sigma(W_r \cdot x_t + U_r \cdot h_{t-1} + b_r) \quad \text{(Compuerta de reinicio)} \tag{2.20}$$

$$\tilde{h}_t = \tanh(W_h \cdot x_t + r_t \odot (U_h \cdot h_{t-1}) + b_h) \quad \text{(Cálculo del estado intermedio)} \tag{2.21}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \quad \text{(Estado actual)} \tag{2.22}$$

Donde:

- x_t: Entrada en el tiempo t.
- h_t: Estado oculto en el tiempo t.
- z_t: Salida de la compuerta de actualización.
- r_t: Salida de la compuerta de reinicio.
- $\tilde{h}_t$: Estado intermedio.
- σ: Función sigmoide.
- tanh: Función tangente hiperbólica.
- W, U, b: Pesos y sesgos aprendidos durante el entrenamiento.

2.1.11.1.1. Ventajas de las GRU

Las GRU ofrecen varias ventajas sobre otros tipos de RNN:

- Menor complejidad computacional que las LSTM, ya que poseen menos parámetros debido a la ausencia de una compuerta de salida [24].

- Rendimiento competitivo en tareas que involucran datos secuenciales, como clasificación de series temporales y procesamiento del lenguaje natural [44].
- Mayor capacidad para generalizar en conjuntos de datos pequeños debido a su arquitectura simplificada.

2.1.11.1.2. Aplicaciones

Las GRU han demostrado ser útiles en una variedad de aplicaciones, incluyendo:

- **Clasificación de series temporales:** Reconocimiento de patrones en datos como señales biomédicas o financieras.
- **Procesamiento de lenguaje natural:** Modelado de texto y traducción automática [44].
- **Predicción de series temporales:** Análisis de tendencias y pronósticos [26].

En comparación con otros enfoques, las GRU equilibran simplicidad y potencia, lo que las hace ideales para problemas donde los recursos computacionales son limitados o los datos son escasos.

2.1.12. Evaluación de modelos

La evaluación de modelos, es una etapa crucial en el proceso de desarrollo de algoritmos de *Machine Learning*. Esta permite determinar la eficacia y eficiencia de un modelo en la tarea para la cual fue diseñado. Aquí se describen los conceptos básicos, las medidas comúnmente utilizadas y sus respectivas ecuaciones.

Entre los conceptos básicos de evaluación de modelos, se encuentran las siguientes instancias:

- **Verdadero Positivo (TP):** instancias en las que el modelo predice correctamente la clase positiva.
- **Falso Negativo (FN):** instancias en las que el modelo no predice la clase positiva cuando debería.
- **Falso Positivo (FP):** instancias en las que el modelo predice la clase positiva cuando no debería.
- **Verdadero Negativo (TN):** instancias en las que el modelo predice correctamente la clase negativa.

Estos valores se utilizan para calcular diferentes métricas que permiten evaluar el rendimiento del modelo.

Las medidas de evaluación, son métricas utilizadas para cuantificar el rendimiento de un modelo de *Machine Learning*. Estas métricas son esenciales para comprender qué tan bien está funcionando un modelo en tareas específicas, permitiendo comparar diferentes modelos y seleccionar el mejor para un problema dado. A continuación, se presentan algunas de las medidas de evaluación más comunes, junto con sus ecuaciones y explicaciones.

1. ***Accuracy* (Exactitud):** el *accuracy* mide la proporción de verdaderas predicciones (tanto positivas como negativas) entre el total de predicciones. Esta medida es útil cuando las clases están balanceadas, es decir, cuando el número de instancias de cada clase es similar. El *accuracy* se define mediante la siguiente ecuación:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.23}$$

2. ***Precision* (Precisión):** la precisión indica qué proporción de las instancias predichas como positivas son realmente positivas. Esta medida es útil en problemas donde el costo de los falsos positivos es alto. Se define mediante la siguiente ecuación:

$$Precision = \frac{TP}{TP + FP} \tag{2.24}$$

3. ***Recall* o Tasa de Verdaderos Positivos (TPR):** el *recall* mide la proporción de instancias positivas que fueron correctamente identificadas por el modelo. Esta medida es crucial en problemas donde es importante no perder instancias positivas, como en diagnósticos médicos. Se calcula mediante la siguiente ecuación:

$$Recall = \frac{TP}{TP + FN} \tag{2.25}$$

4. **Especificidad (Tasa de Verdaderos Negativos):** la especificidad mide la proporción de instancias negativas que fueron correctamente identificadas. Esta medida es relevante en problemas donde el costo de los falsos negativos es alto. Se calcula mediante la siguiente ecuación:

$$Specificity = \frac{TN}{TN + FP} \tag{2.26}$$

5. ***F1 Score*:** el *F1 Score* es la media armónica de la precisión y el *recall*, y se utiliza cuando se busca un balance entre ambos. Se calcula mediante la siguiente ecuación:

$$F1\ Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{2.27}$$

2.1.12.1. Curva ROC (*Receiver Operating Characteristic*)

La curva ROC, es una herramienta gráfica utilizada para evaluar el rendimiento de un modelo de clasificación binaria. Muestra la relación entre la Tasa de Verdaderos Positivos (*True Positive Rate*, TPR o *recall*) y la Tasa de Falsos Positivos (*False Positive Rate*, FPR) a diferentes umbrales de decisión [48]. La Tasa de Falsos Positivos se calcula de la siguiente manera:

$$\text{FPR} = \frac{FP}{FP + TN} \tag{2.28}$$

El eje vertical de la curva ROC representa la TPR, mientras que el eje horizontal representa la FPR. Al variar el umbral de decisión del modelo, se obtienen diferentes pares de valores (TPR, FPR), los cuales se grafican para formar la curva ROC [48]. La figura 2.5 muestra los distintos casos del comportamiento de esta curva, en donde una línea diagonal indica un clasificador totalmente aleatorio. Bajo aquella línea el clasificador demostraría un

comportamiento peor, mientras que sobre esta demostraría un mejor comportamiento. Un clasificador perfecto se encontraría en la esquina izquierda superior.

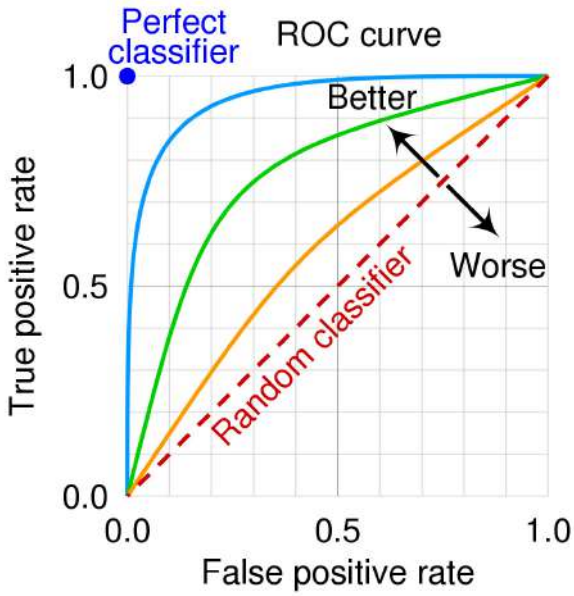

Figura 2.5: Comportamiento de una curva ROC [49].

Un aspecto importante de la curva ROC, es el **Área Bajo la Curva (AUC)**, que proporciona una medida agregada del rendimiento del modelo en todos los umbrales posibles. Un AUC de 1 indica un modelo perfecto, mientras que un AUC de 0.5 indica un modelo sin capacidad discriminativa. Este se calcula a través de la siguiente ecuación:

$$\text{ROC-AUC} = \int_0^1 \text{TPR}\, d(\text{FPR}) \tag{2.29}$$

2.1.12.2. Curva *Precision-Recall*

La curva *Precision-Recall*, es otra herramienta gráfica utilizada para evaluar el rendimiento de un modelo de clasificación, especialmente útil en conjuntos de datos desbalanceados donde las clases positivas son mucho menos frecuentes que las clases negativas [50]. Esta curva se construye graficando la precisión contra el *recall* para diferentes umbrales de decisión [51], como se puede observar en la figura 2.6.

Figura 2.6: Comportamiento de una curva *Precision-Recall* [52].

Al igual que con la curva ROC, el área bajo la curva *Precision-Recall* (PR AUC) también puede proporcionar una medida agregada del rendimiento del modelo [51]. Sin embargo, en escenarios con datos desbalanceados, la curva *Precision-Recall* suele ser más informativa que la curva ROC, ya que refleja mejor el rendimiento del modelo en la clase minoritaria [50].

2.1.12.3. *Cross-Validation*

La validación cruzada es una técnica ampliamente utilizada para evaluar cómo un modelo se generaliza a un conjunto de datos independiente. Esta técnica resulta especialmente valiosa cuando se trabaja con conjuntos de datos pequeños, en los que cada observación es importante tanto para el entrenamiento como para la validación. En lugar de reservar una sola parte de los datos para pruebas, la validación cruzada divide el conjunto en varias particiones o "*folds*" para asegurar una evaluación más exhaustiva y confiable del modelo [53].

El procedimiento típico implica dividir los datos en k subconjuntos aproximadamente iguales. En cada iteración, el modelo se entrena utilizando $k-1$ particiones, mientras que la partición restante se utiliza para probar el modelo. Este proceso se repite k veces, cambiando la partición de prueba en cada iteración. Al final, se promedian las métricas de rendimiento obtenidas en cada iteración, lo que proporciona una estimación más robusta del error del modelo [10].

La validación cruzada es muy útil no solo para la selección de modelos y la estimación de su rendimiento, sino también para obtener información sobre la estabilidad y la variabilidad de las predicciones. Al analizar la variación en las métricas (por ejemplo, precisión, error cuadrático medio), se puede evaluar la consistencia del rendimiento del modelo en diferentes subconjuntos de datos, identificando posibles problemas de sobreajuste o subajuste [20].

El esquema general es el siguiente:

1. Dividir el conjunto de datos en k particiones o "*folds*".
2. Para cada iteración i:
    - Entrenar el modelo utilizando $k-1$ particiones.
    - Evaluar el modelo con la partición restante i.
3. Calcular el error total como la media de los errores en las k particiones.

Esta técnica es particularmente ventajosa en situaciones donde se necesita obtener una estimación confiable del error sin desperdiciar datos, proporcionando una visión más completa del rendimiento del modelo en comparación con un simple particionado en entrenamiento y prueba [20].

2.1.13. Selección y Filtrado de Características

La selección de características, es un paso crucial en el preprocesamiento de datos en *Machine Learning*, ya que influye directamente en el rendimiento de los modelos predictivos. A continuación, se describen varios conceptos clave relacionados con la selección de características.

2.1.13.1. Correlación de Características

La correlación de características, es una medida estadística que describe la relación entre dos variables. Es importante identificar y gestionar las características correlacionadas para evitar redundancias y mejorar la eficiencia del modelo. La correlación se calcula utilizando la siguiente ecuación:

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} \tag{2.30}$$

donde $\rho_{X,Y}$ es el coeficiente de correlación entre las variables X e Y, $cov(X,Y)$ es la covarianza entre X e Y, y σ_X y σ_Y son las desviaciones estándar de X e Y [18].

2.1.13.2. Maldición de la Dimensionalidad

La maldición de la dimensionalidad, se refiere a varios fenómenos que ocurren cuando se analizan y organizan datos en espacios de alta dimensión. A medida que aumentan las dimensiones, el volumen del espacio de datos crece exponencialmente, lo que puede dificultar la clasificación y el análisis de datos [54]. Un ejemplo visual de este fenómeno puede encontrarse en la figura 2.7.

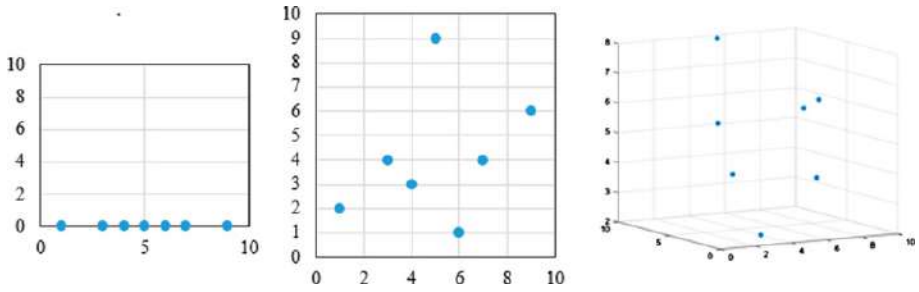

Figura 2.7: Ejemplo del efecto de la Maldición de la Dimensionalidad al ser proyectado en una dimensión, dos dimensiones y tres dimensiones (de izquierda a derecha respectivamente) [54].

2.1.13.3. Filtrado de Características

El filtrado de características es una técnica utilizada para seleccionar o eliminar características basándose únicamente en sus propiedades estadísticas, sin involucrar algoritmos de *Machine Learning* complejos. Este enfoque ayuda a abordar tres problemas principales:

- Características que aportan la misma información.
- Características que no aportan información relevante.
- Un gran número de dimensiones que pueden entorpecer la clasificación.

La eliminación de características constantes es un ejemplo de métodos de filtrado, ya que se fundamenta en la ausencia de variabilidad en dichas características. Incorporar estos pasos tempranos en el flujo de trabajo de ciencia de datos, mejora la eficiencia del proceso, y sienta las bases para construir modelos más robustos y precisos [55].

Cuando se trabaja con conjuntos de datos con numerosas *features* (características), es común encontrar correlaciones entre múltiples características. La selección inteligente de *features* implica decidir cuáles conservar basándose en varios criterios:

- **Rendimiento del modelo:** algunas *features* generan modelos con un rendimiento superior.
- **Variabilidad y cardinalidad:** las *features* con mayor variabilidad o cardinalidad proporcionan más información.
- **Datos faltantes:** las *features* con menos datos faltantes suelen ser más confiables e informativas.

Estas estrategias ayudan a mantener las características más relevantes y eliminar redundancias, mejorando así el rendimiento del modelo [56].

2.1.14. Interpretabilidad de Modelos

El problema con los modelos de *Machine Learning*, es que estos responden principalmente al "¿Qué?", enfocándose solo en los resultados [57]. A menudo, es común preocuparse únicamente por métricas como *accuracy* o *F1-score*, lo que es una descripción incompleta del

mundo real, ya que una respuesta correcta resuelve solo parte del problema [58].

Por lo general, se evita responder al "¿Por qué?" [57], pregunta que busca una explicación de la salida obtenida por nuestros modelos, y cuya respuesta nos puede llevar a aprender más sobre el problema que se intenta resolver [59].

2.1.14.1. Impacto de una Buena Interpretación en *Machine Learning*

La interpretabilidad de los modelos de *Machine Learning* proporciona un mejor conocimiento del problema a trabajar, transformándolo en una fuente de conocimiento [57]. A continuación, se señalan algunos de los puntos más importantes que aporta la interpretabilidad en *Machine Learning*:

- **Predicciones seguras:** al desarrollar un modelo capaz de manejar un automóvil automáticamente, es esperable que las predicciones/clasificaciones del modelo sean seguras para el usuario. Por esto, la comprensión del "¿cómo?" está tomando decisiones aquel modelo es crucial, ya que de lo contrario podría tener consecuencias catastróficas [60].
- **Detección de sesgo:** la interpretabilidad permite reconocer sesgos en los modelos, descubriendo si están inclinados hacia una cierta población, excluyendo a un grupo minoritario [61].
- **Aceptación social y manejo de interacciones sociales:** la integración de los modelos de *Machine Learning* en el día a día exige un aumento de confianza en ellos [62].
- **Depuración (*debugging*):** la comprensión de un modelo permite una mejor depuración, ya que se sabría cómo las componentes impactan en las salidas del mismo [63].

2.1.14.2. ¿Cuándo no es Necesaria la Interpretabilidad?

Si bien resulta interesante y beneficiosa la interpretabilidad en los modelos, en algunos casos no es necesaria [57]:

- Cuando un proyecto de *Machine Learning* no posee un impacto significativo.
- Si el problema está bien estudiado, la interpretabilidad deja de ser relevante.
- Cuando el algoritmo de *Machine Learning* permite a las personas manipular el sistema.

2.1.14.3. Modelos Intrínsecos

Los modelos intrínsecos son aquellos considerados interpretables debido a su estructura simple [57]. Ejemplos incluyen regresiones lineales y árboles de decisión [64].

2.1.14.4. Cajas Negras (*Black Boxes*)

En un modelo de caja negra, los datos de entrada se procesan a través de múltiples capas o algoritmos complejos que generan las predicciones o resultados finales. La forma en que el modelo llega a esas predicciones puede ser desconocida o difícil de explicar [57]. Esto puede presentar desafíos en términos de transparencia, interpretabilidad y confianza en las decisiones tomadas por el modelo [64].

Aunque los modelos de caja negra pueden ofrecer un alto rendimiento predictivo, plantean preocupaciones sobre la explicabilidad de las decisiones y la identificación de posibles sesgos o discriminación en los resultados [64].

2.2. Estado del Arte

2.2.1. Diagnóstico de Glaucoma a través de *Machine Learning*

En el campo de la detección del glaucoma, se han desarrollado numerosas técnicas por parte de investigadores con el objetivo de mejorar la precisión y la automatización del diagnóstico. Entre estas técnicas, destacan aquellas basadas en el uso de *Machine Learning*, que permiten extraer características relevantes de los datos y clasificarlos utilizando una variedad de clasificadores [65].

En la actualidad, se han utilizado diferentes modelos de *Deep Learning*, como las redes neuronales convolucionales (CNN), para diagnosticar enfermedades como el glaucoma de manera más automatizada [66]. Estos modelos han demostrado su eficacia en estudios donde se han utilizado imágenes de tomografías de coherencia óptica (OCT), alcanzando valores de precisión del 96.27 % mediante el uso de modelos pre-entrenados [67]. Además, se han empleado imágenes de fondo ocular (*fundus images*) junto con técnicas de *transfer learning* y *data augmentation* [68], logrando una precisión del 98.48 % [66]. En las figuras 2.8 y 2.9 se pueden observar ejemplos de imágenes OCT y de fondo ocular, respectivamente.

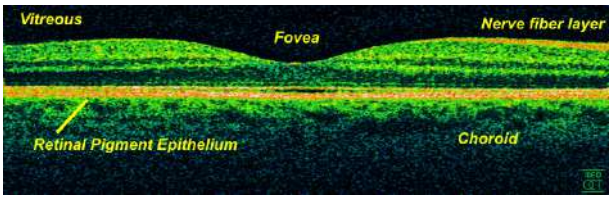

Figura 2.8: Ejemplo de imagen OCT.

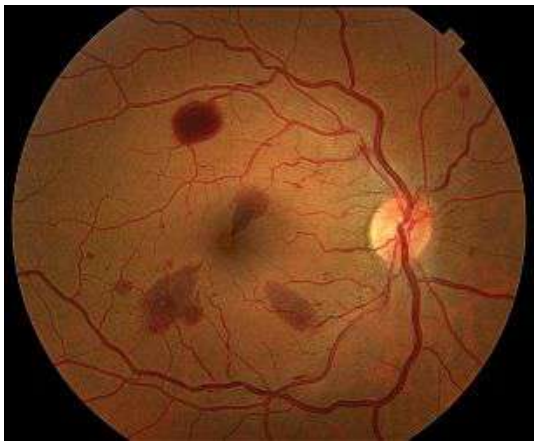

Figura 2.9: Ejemplo de imagen de fondo ocular.

A pesar de los resultados prometedores obtenidos con estas metodologías, es importante tener en cuenta que la obtención de este tipo de imágenes, ya sea de OCT o de fondo ocular, se realiza a través de dispositivos costosos, no portátiles y de difícil implementación, que requieren ser operados por expertos. Además, estas técnicas no son completamente efectivas para identificar a todos los pacientes en etapas tempranas de la enfermedad, lo que subraya

la necesidad de seguir investigando en el desarrollo de herramientas más accesibles y precisas para el diagnóstico temprano del glaucoma.

2.2.2. Pupilometría Cromática en Glaucoma

En un estudio en preparación dirigido por el Dr. Iván Plaza Rosales del Laboratorio de Neurosistemas de la Universidad de Chile, se ha implementado un innovador *setup* experimental con el propósito de evaluar la respuesta pupilar a distintos estímulos lumínicos en pacientes diagnosticados con glaucoma, así como en individuos sanos [4]. El objetivo fundamental es establecer el potencial de la pupilometría cromática como biomarcador en el diagnóstico precoz de esta enfermedad ocular.

Este *setup* experimental (ver figura 2.10), comprende un dispositivo capaz de generar estímulos lumínicos cromáticos, que está conectado a un sistema de *PupilLabs*, permitiendo la grabación y seguimiento ocular de cada individuo participante. Los estímulos lumínicos incluyen *flashes* de luz azul, roja, verde y blanca, así como una rampa de estos mismos colores, lo que suma un total de 8 estímulos distintos.

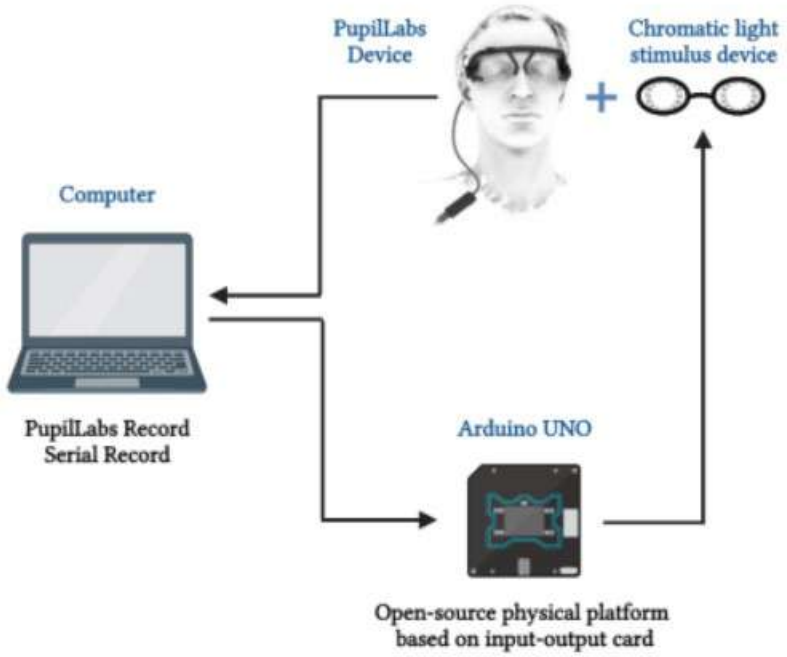

Figura 2.10: *Setup* experimental [4].

Los experimentos realizados han revelado alteraciones significativas en la respuesta pupilar en los sujetos diagnosticados con glaucoma en comparación con los controles sanos, como se observa en los gráficos de la figura 2.11. Específicamente, se desarrolló un potencial *test de screening* basado en la respuesta pupilar a los estímulos de *flash* azul, que permite registrar de manera exitosa la respuesta pupilar generada por las células ganglionares de la retina intrínsecamente fotosensibles (IPRGC) mediante pupilometría cromática. Estos resultados demuestran la capacidad de este método para detectar la función de las IPRGC en pacientes

con glaucoma [4].

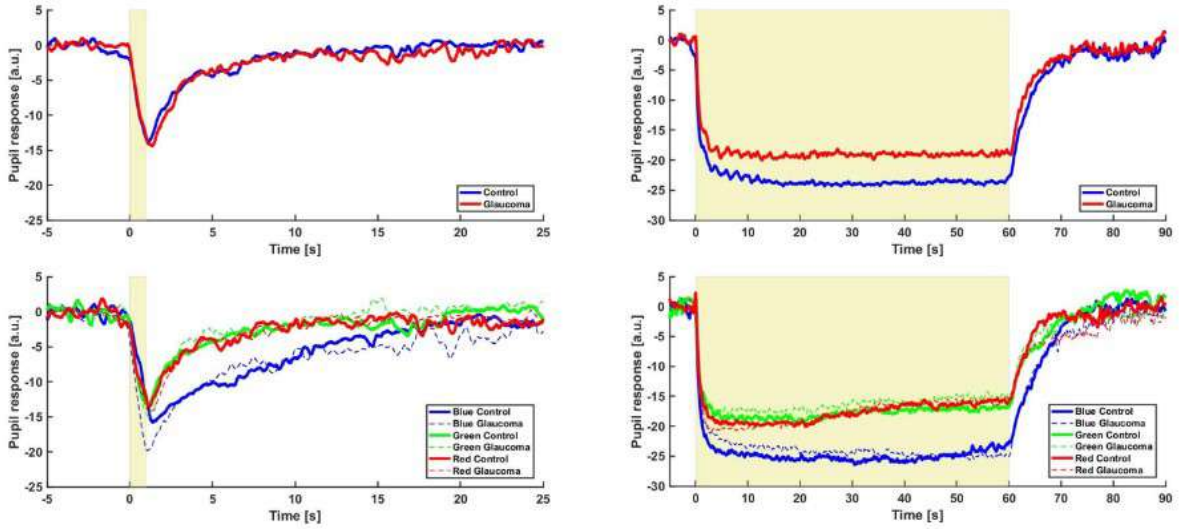

Figura 2.11: Diferencias en función de la longitud de onda del estímulo. Izquierda: respuesta pupilar a *flashes*. Derecha: respuesta pupilar a rampas de luz [4].

A pesar del éxito observado en estos resultados preliminares, hasta el momento no se ha puesto mucho énfasis en desarrollar un clasificador de glaucoma basado en la respuesta pupilar a la luz utilizando técnicas de *Machine Learning*, pese a la disponibilidad de los datos recopilados. Dado lo anterior, para este trabajo de título, se toma este estudio en preparación junto a los resultados obtenidos del *setup* experimental.

2.2.3. Técnicas de aprendizaje automático para mejorar la detección del glaucoma a través del técnicas de pupilometría cromática

Hasta la fecha, el uso de técnicas de pupilometría cromática para detectar glaucoma ha recibido poca atención en la literatura científica. Sin embargo, un trabajo reciente publicado durante la redacción de este informe titulado "*Evaluating machine learning techniques for enhanced glaucoma screening through Pupillary Light Reflex analysis*" [69] aborda este tema. El estudio evalúa diversas técnicas de aprendizaje automático para mejorar la detección de glaucoma a partir de análisis del reflejo pupilar ante estímulos lumínicos.

El conjunto de datos de esta investigación incluye 512 vídeos, cada uno con una duración de 4 minutos y 5 segundos, capturados a 30 cuadros por segundo, sumando un total de 7350 cuadros por vídeo. Estos vídeos documentan la respuesta pupilar a estímulos de luz en 250 voluntarios, de los cuales 113 eran sujetos sanos y 137 padecían glaucoma. Cabe destacar que algunos voluntarios realizaron múltiples grabaciones por ojo.

Los estímulos lumínicos aplicados fueron de tipo *flash*, utilizando luces de colores rojo, verde, azul y blanco. El protocolo del experimento (ver figura 2.12) consistió en presentar a cada sujeto un *flash* de 1 segundo de duración, seguido de un periodo de descanso de 59

segundos, antes de presentar el siguiente estímulo. Todos los vídeos fueron capturados con la misma frecuencia de muestreo y duración.

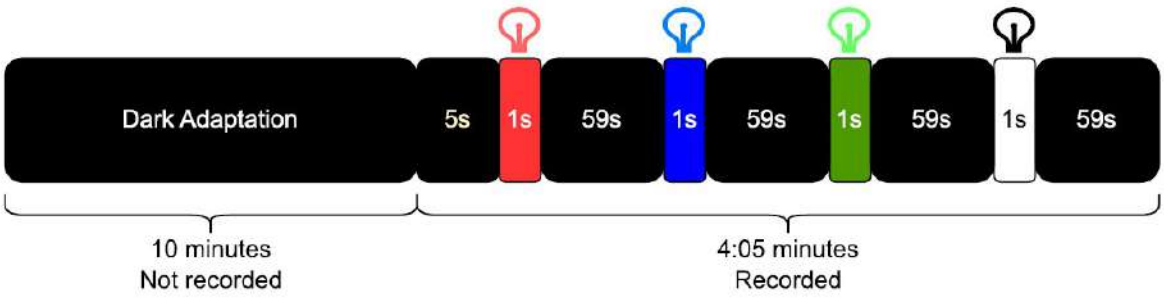

Figura 2.12: Protocolo utilizado en el experimento [69].

El estudio se desarrolló en varias etapas:

1. **Segmentación y preprocesamiento:** Se procesaron los vídeos utilizando técnicas de procesamiento de imágenes para segmentar la pupila a lo largo del tiempo. Posteriormente, se eliminaron las secciones de las señales no relevantes, como los periodos de oscuridad. Un ejemplo de las señales resultantes se muestra en la figura 2.13.

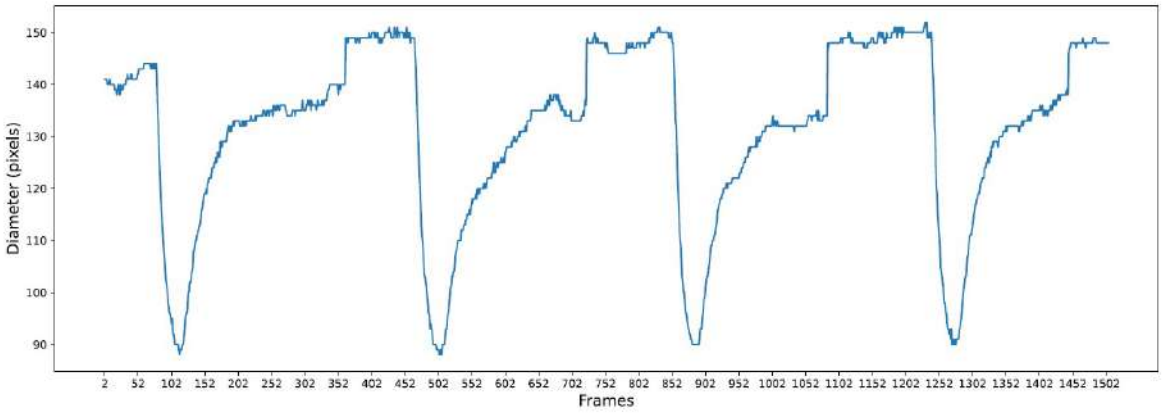

Figura 2.13: Señal pupilar después del procesamiento y eliminación de secciones redundantes [69].

2. **Filtrado:** Se aplicaron diferentes métodos de filtrado para eliminar el ruido presente en las señales. Se probaron tres métodos de filtrado: sin filtro, un filtro personalizado y un filtro de paso bajo.

3. **Extracción de características:** Se siguieron dos enfoques: el primero utilizó las señales completas sin extracción de características, y el segundo se centró en la extracción de características relevantes de la literatura. Entre estas, se encuentran:

   a) Diámetro inicial

   b) Contracción máxima

c) Tiempo hasta la contracción máxima

d) Amplitud de contracción absoluta

e) Amplitud de contracción relativa

f) Velocidad de contracción

g) Estado latente

h) Diámetro de redilatación después de 6 s

i) Tiempo de redilatación

j) Complejidad fractal de Higuchi

k) Diámetro al final del examen (aplicado a la señal completa)

l) Complejidad fractal de Higuchi en la señal (aplicado a la señal completa)

4. **Balanceo del *dataset*:** Se aplicaron técnicas de submuestreo y sobremuestreo para equilibrar las clases en el conjunto de datos.

5. **Selección de características:** Se implementaron tres métodos de selección de características: *SelectKBest* (basado en filtros), *Recursive Feature Elimination* (RFE, basado en envoltura) y *LassoCV* (método integrado).

6. **Segmentación de señales:** Las señales se dividieron en cuatro segmentos según el color de la estimulación: rojo, azul, verde y blanco.

7. **Clasificación:** Se evaluó la eficacia de varias técnicas de clasificación, incluyendo:

   - *Linear Discriminant Analysis (LDA)*
   - *K-Nearest Neighbors (KNN)*
   - *Decision Tree*
   - *Random Forest*
   - *Extra Trees*
   - *AdaBoost*
   - *Gradient Boosting*
   - *Naive Bayes*
   - *Support Vector Machine (SVM)*
   - *Redes Neuronales*

En cuanto a los resultados, el modelo que mejor rendimiento mostró fue el clasificador *Linear Discriminant Analysis* (LDA), logrando una precisión del 73.9 %. La matriz de confusión del mejor modelo se puede observar en la figura 2.14, mientras que la curva ROC se presenta en la figura 2.15.

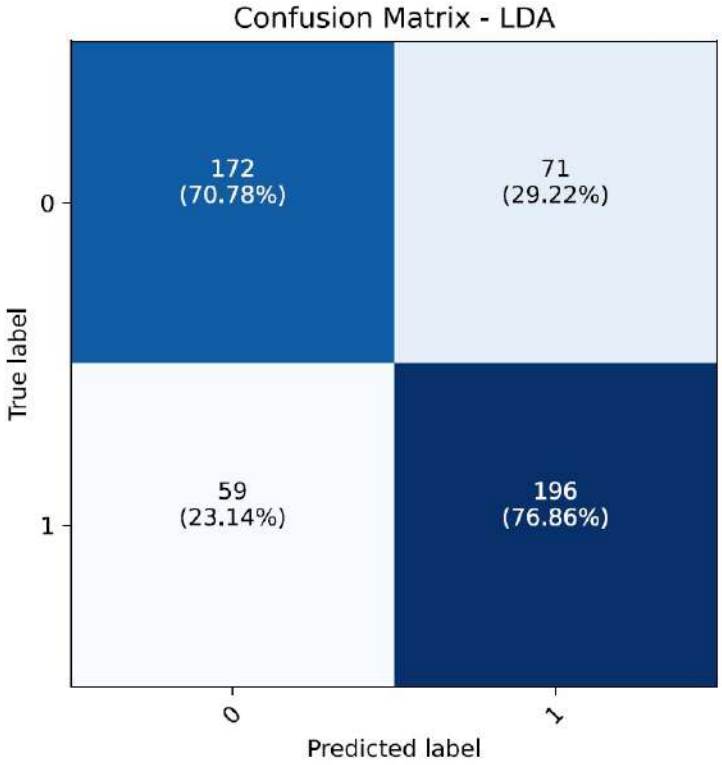

Figura 2.14: Matriz de confusión del mejor modelo y experimento hecho en el estudio [69].

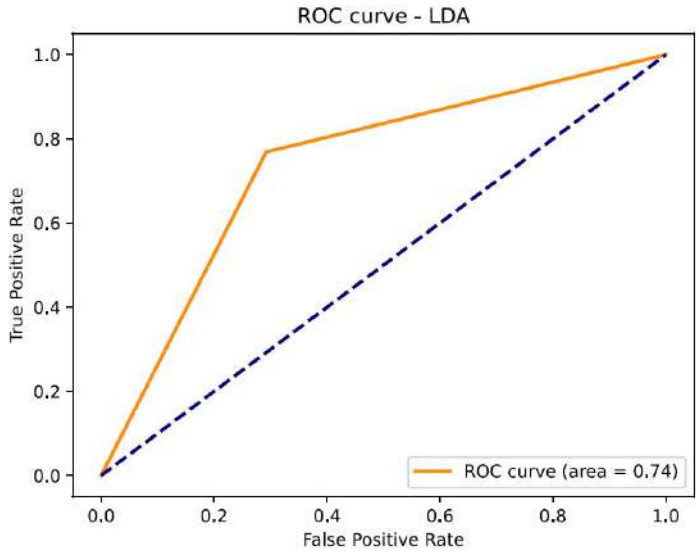

Figura 2.15: Curva ROC del mejor modelo y experimento hecho en el estudio [69].

Capítulo 3

Metodología

En este capítulo se presenta, de manera detallada y estructurada, la metodología empleada para el desarrollo de este trabajo. Cada sección aborda un paso específico, desde la obtención de los datos experimentales hasta el diseño, implementación y evaluación de los modelos predictivos.

3.1. Resumen de los pasos metodológicos

Antes de profundizar en los detalles de la metodología implementada, en la figura 3.1 se presenta un cuadro resumen que sintetiza los principales pasos realizados en este trabajo. Este resumen tiene como objetivo proporcionar una visión general de todo el proceso, desde la obtención de datos hasta la evaluación de los modelos.

En las secciones siguientes, se describen cada uno de estos pasos de manera detallada, destacando las técnicas utilizadas, las configuraciones específicas y los resultados obtenidos en cada etapa. Este desglose permitirá entender cómo se implementaron y optimizaron las diferentes estrategias para cumplir con los objetivos del trabajo.

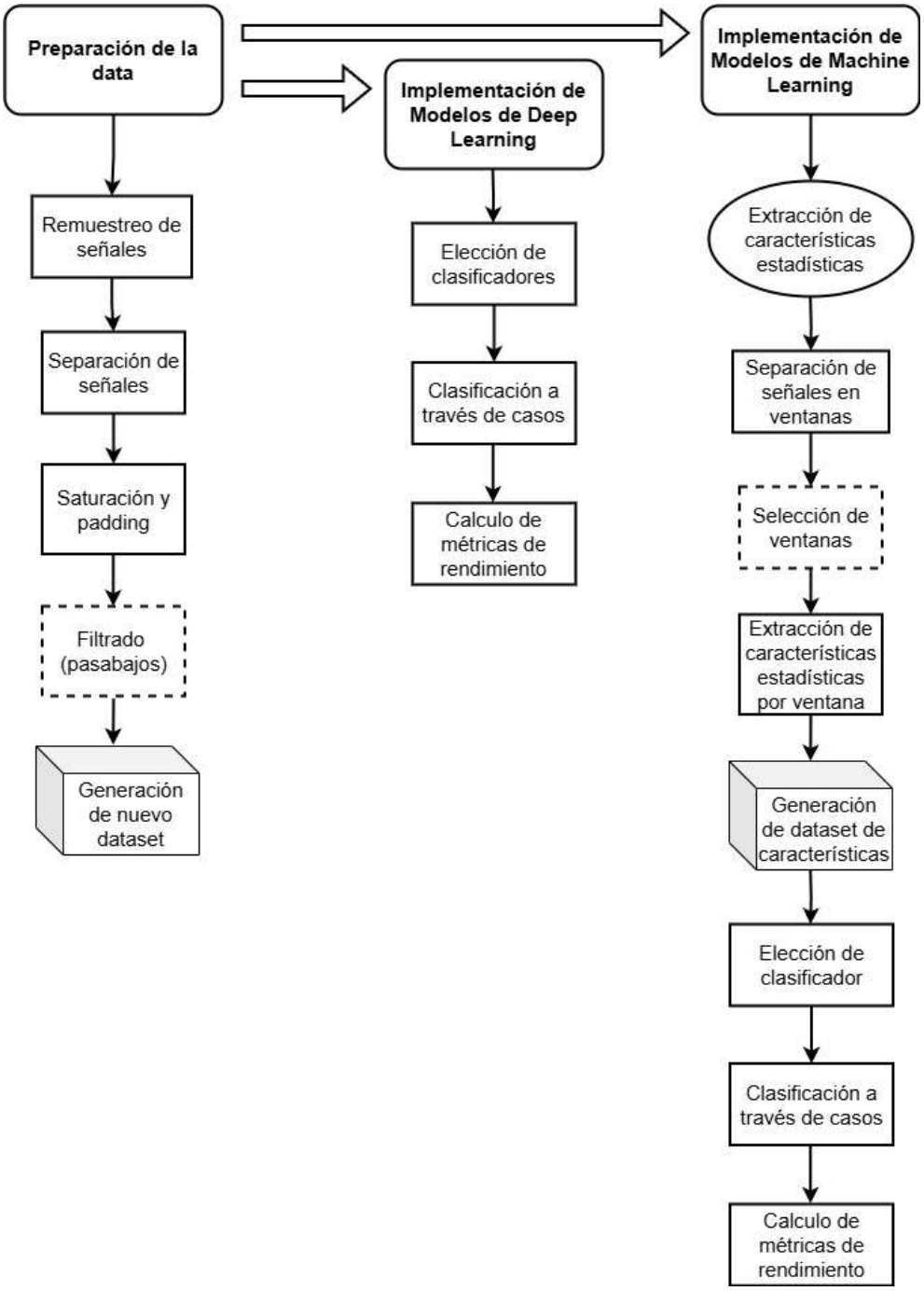

Figura 3.1: Cuadro resumen metodológico.

3.2. Obtención de datos del experimento de pupilometría cromática

En el experimento de pupilometría cromática realizado por el Dr. Iván Plaza Rosales [4], participaron 40 individuos en total: 20 pacientes controles sanos y 20 pacientes con diferentes niveles de glaucoma.

Cada individuo fue evaluado en el *setup* experimental mostrado en la figura 2.10, el cual consiste de un sistema de registro comercial (*PupilLabs*) y un sistema de estimulación lumínica de confección propia. A cada paciente se le aplicaron todos los tipos de estímulos: *flashes* y rampas de luz azul, roja, verde y blanca, lo que resultó en un total de 8 estímulos distintos. La duración de cada experimento fue de entre 13 y 16 minutos, en los que los pacientes fueron expuestos a todos los estímulos en un orden aleatorio y sin interrupciones. Los estímulos de flash tuvieron una duración de 1 segundo (con una intensidad de 250 lux), mientras que los estímulos de rampa duraron 1 minuto (20 segundos de incremento gradual de la intensidad y 40 segundos a máxima intensidad, correspondiente a 250 lux). Para todos los experimentos, se incluyó un período inicial de 2 minutos de oscuridad, seguido de 1 minuto de oscuridad entre cada estímulo, y 1 minuto final de oscuridad. El orden general de los estímulos se muestra en la figura 3.2.

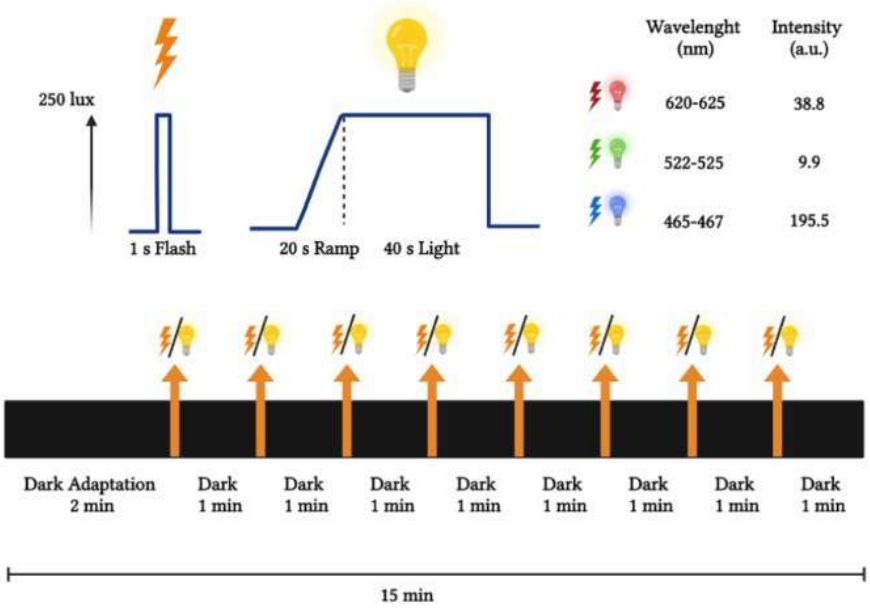

Figura 3.2: Orden general de los estímulos del experimento [4].

Se utilizó un dispositivo *PupilLabs*, equipado con dos cámaras que grababan ambos ojos del sujeto durante el experimento. Los vídeos capturados fueron posteriormente procesados con el software del dispositivo *PupilLabs*, el cual analizó los vídeos y extrajo información relacionada con la ubicación de los ojos y pupilas, así como el diámetro de las pupilas a lo largo del tiempo. El *output* final, por paciente, es un archivo `.csv`, llamado `pupil_positions.csv`, el cual contiene las mediciones realizadas durante el experimento de pupilometría cromática. Este contiene los siguientes campos clave:

- `timestamp`: Marca de tiempo asociada a cada fotograma de la imagen fuente (en segundos), que indica cuándo fue capturada la medición.
- `world_index`: Identificador del fotograma del vídeo del mundo más cercano al evento. Este índice facilita la asociación de las mediciones con los eventos registrados en el vídeo.
- `id`: Indica el ojo al que corresponde la medición. El valor puede ser `0` o `1`, donde `0` corresponde al ojo derecho y `1` al ojo izquierdo (desde el punto de vista de quien lleva puesto el dispositivo).
- `confidence`: Representa una evaluación por parte del detector de pupilas sobre el nivel de confianza en la medición. Un valor de `0` indica ausencia de confianza en la medición (por lo que debe ignorarse). Un valor de `1` indica confianza completa en la medición. En general, los datos útiles suelen tener un valor de confianza superior a 0.6.
- `norm_pos_x`: Posición de la pupila en el eje x, en coordenadas normalizadas, dentro del marco de la imagen del ojo.
- `norm_pos_y`: Posición de la pupila en el eje y, en coordenadas normalizadas, dentro del marco de la imagen del ojo.
- `diameter`: Diámetro de la pupila en píxeles de la imagen, tal como se observa en el marco de la imagen del ojo. Este valor no está corregido por la perspectiva.
- `method`: Indica qué detector de pupilas fue utilizado para realizar la detección en cada medición. Se trata de un *string* que describe el método aplicado.

Este archivo proporciona la base de datos cruda para cada paciente, que debe ser procesada antes de poder analizarse en profundidad. Las columnas más importantes, como el `timestamp`, el `id` y el `diameter`, permiten obtener información crítica para el análisis del comportamiento pupilar bajo distintos estímulos lumínicos.

Dado que los estímulos se presentaron en un orden aleatorio, junto con los archivos `.csv`, se generó un archivo `.txt` para cada paciente que detallaba el orden de los estímulos aplicados. Cabe mencionar que el archivo `.csv` no contiene información directa sobre los estímulos lumínicos, sino que presenta únicamente las características físicas de los ojos de los sujetos a lo largo del tiempo. La característica física más importante, y con la que se trabajó, es el diámetro del iris (en píxeles) a través del tiempo. Un ejemplo de este a través de diversos *timestamps* se puede ver en la figura 3.3, donde se grafica la respuesta pupilar ante diferentes estímulos lumínicos.

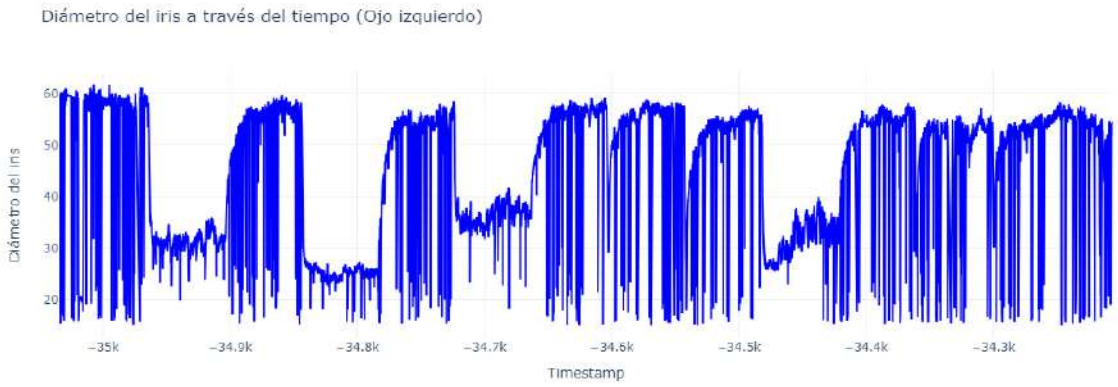

Figura 3.3: Respuesta pupilar a través de los diversos *timestamps* del ojo izquierdo de un paciente.

3.3. Preparación de los datos

Para implementar herramientas de clasificación en un proyecto de *Data Science*, el primer paso es contar con una base de datos sólida con la cual trabajar. En este caso, para desarrollar una herramienta que permita el diagnóstico del glaucoma a través del análisis de la respuesta pupilar a estímulos lumínicos cromáticos, es esencial preparar todos los datos crudos obtenidos del *setup* experimental y generar una base de datos organizada y legible.

El primer paso fue desarrollar un *dataframe* utilizando la librería `pandas`, que contiene toda la información relevante de manera estructurada, facilitando su manipulación y análisis de forma simple e intuitiva. Para lograrlo, se separó la respuesta pupilar de cada estímulo lumínico, tratándola como un dato individual. En la figura 3.4 se muestra un ejemplo de esto, donde se segmenta la respuesta pupilar del mismo ejemplo presentado en la figura 3.3, pero ahora separando cada respuesta pupilar ante los distintos estímulos. Los segmentos 1, 2, 3 y 6 corresponden a respuestas pupilares a estímulos de tipo rampa, mientras que los segmentos 4, 5, 7 y 8 corresponden a respuestas pupilares a estímulos de tipo *flash*.

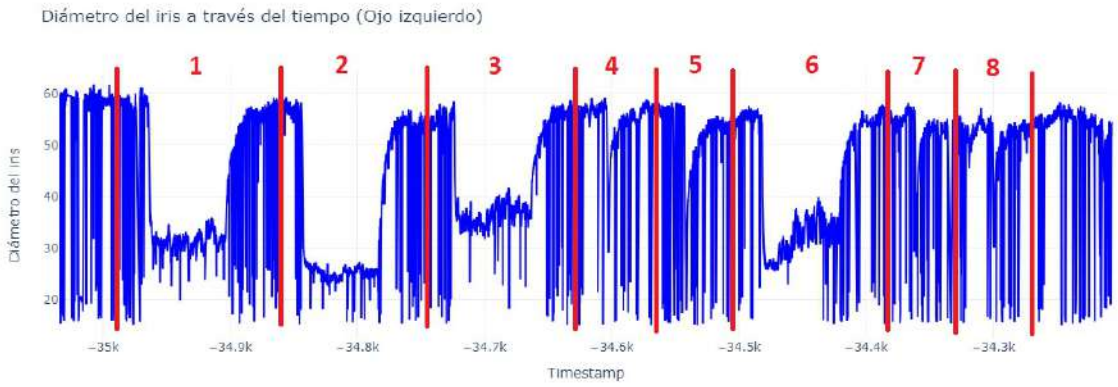

Figura 3.4: Respuesta pupilar segmentada por tipos de estímulos lumínicos.

El nuevo *dataset* generado contiene varias columnas: un ID que identifica el código del paciente, el ojo correspondiente (izquierdo o derecho), el tipo de estímulo, el color del estímulo y si el paciente padece de glaucoma. Esta estructura proporciona una base de datos más robusta, lo que facilita un trabajo más fluido y eficiente con los datos.

El objetivo de esta sección es describir los principales problemas encontrados durante el preprocesamiento de los datos y detallar cómo estos fueron resueltos para generar el *dataset* descrito.

3.3.1. Problemas de la data

Tras un análisis del comportamiento de las señales en los datos originales, junto con la revisión detallada del proceso experimental en el que se obtuvieron, se identificaron las siguientes dificultades:

1. **No hay ninguna indicación en los datos que especifique en qué momento se presenta cada estímulo**. Solo se dispone de archivos `.txt` que contienen el orden de

los estímulos para cada paciente.

2. La generación de estímulos lumínicos no está vinculada directamente (en términos de código) con la grabación de los vídeos ni con la obtención de los datos. Es decir, la grabación se inició de manera manual y luego se ejecutó un código que generaba los estímulos lumínicos. **Esto implica que tanto el inicio como el final de los datos generados son variables**.

3. Los vídeos de los pacientes presentan **longitudes diferentes** y **frecuencias de muestreo variables** entre cada uno de ellos.

4. Aunque los vídeos de cada paciente (ojo izquierdo y ojo derecho) tienen la misma duración, existe una **discrepancia en la cantidad de muestras** entre ambos. Esto implica que poseen **frecuencias de muestreo diferentes** y, por lo tanto, ***timestamps* distintos**.

La principal dificultad que surge de los problemas 1 y 2 es que, para separar cada respuesta pupilar por tipo de estímulo, se debe verificar manualmente la señal. En otras palabras, es necesario observar la señal y estimar el momento exacto en que comienza el experimento, ya que no se dispone de un tiempo inicial y final fijo.

Por otro lado, el principal obstáculo derivado de los problemas 3 y 4 es que las respuestas pupilares tienen diferentes longitudes, aunque temporalmente la duración de estas sea la misma. Esto complica la generación de un *dataset*, ya que todos los datos deben tener una longitud fija, es decir, todas las filas deben tener la misma cantidad de columnas. Además, trabajar exclusivamente con los *timestamps* es problemático, ya que los valores de estos difieren entre los pacientes, e incluso entre los ojos del mismo paciente en un mismo experimento. Esto hace que coordinar el inicio preciso del experimento para separar las respuestas pupilares sea complicado y consuma mucho tiempo.

3.3.2. Generación del *dataset*

Para la generación del *dataset* y solucionar los problemas identificados en la sección 3.3.1, se tomaron las siguientes acciones:

1. **Remuestreo de las señales:** Se remuestrearon las señales con respecto a la variable `world_index`. Como se mencionó anteriormente, este identifica el fotograma del vídeo del mundo más cercano al evento registrado en `timestamp`. Los vídeos de ambos ojos tenían una tasa de muestreo mayor que el vídeo del mundo (que graba el entorno). En promedio, los vídeos de los ojos se registraban con aproximadamente 350,000 muestras, mientras que el del mundo lo hacía con alrededor de 24,500 muestras. Para solucionar el problema de tener diferentes tasas de muestreo, se aproximó cada muestra de los vídeos de los ojos a la muestra más cercana del vídeo del mundo. Se desarrolló una función que filtra el *dataframe* para que solo mantenga una fila por cada valor de `world_index`, seleccionando la fila con el valor más alto de `confidence`.

2. **Generación de archivo que indica el inicio del experimento:** Para poder separar las reacciones pupilares por estímulos, es necesario indicar el número del fotograma donde comienza el experimento. Combinando esta información con el archivo `.txt` de cada paciente (que contiene el orden de los estímulos) y sabiendo la duración de cada

estímulo, es posible realizar esta separación. Se observó manualmente cada experimento y se estimó el índice del *dataframe* inicial entre 8 a 10 segundos antes de la aparición de la primera respuesta pupilar. Con esta información, se generó un archivo `.csv` que contiene el ID del paciente y el índice del fotograma inicial del experimento.

3. **Generación del *dataset* vacío:** Se generó un *dataset* vacío con las siguientes columnas:

   - `id`: Código del paciente.
   - `eye`: Corresponde al ojo del paciente. Los valores pueden ser `left` o `right`.
   - `stimuli`: Indica el tipo de estímulo. Los valores posibles son `ramp` o `flash`.
   - `color`: Color del estímulo. Los valores posibles son `red`, `green`, `blue`, o `white`.
   - `glaucoma`: Indica si el paciente padece glaucoma o no. Los valores pueden ser `yes` o `no`.
   - `sample_1`, `sample_2`, ..., `sample_2770`: Correspondientes a 2770 columnas, donde se registran los valores del diámetro pupilar (en píxeles) en cada muestra. Se fijó un largo de 2770 para todas las respuestas pupilares de los estímulos, correspondiente a aproximadamente 92 segundos por reacción pupilar.

Con los pasos descritos anteriormente, se generó una función para rellenar el nuevo *dataset*, utilizando también el archivo `.csv` que indica el inicio del experimento. Esta función realiza las siguientes operaciones:

1. Toma los códigos de todos los pacientes.
2. Busca su archivo `pupil_positions.csv`.
3. Separa los datos entre ojo izquierdo y derecho.
4. Remuestra las señales como se indicó anteriormente.
5. Determina si el paciente tiene glaucoma o no (utilizando su ID/código).
6. Calcula la tasa de muestreo, que es fija (29.83 muestras por segundo) para todos los pacientes gracias al remuestreo.
7. Calcula la duración de los estímulos.
8. Rellena el *dataframe* con la información de los estímulos.
9. Pasa al siguiente estímulo.
10. Una vez completada la información del paciente, pasa al siguiente.

De esta forma, se generó un *dataset* con 2775 columnas y 656 filas, como se muestra en la figura 3.5.

	id	eye	stimuli	color	glaucoma	sample_1	sample_2	sample_3	sample_4	sample_5	...	sample_2761	sample_2762	sample_2763	sample_2764	sample_2765	sample_2766	sample_2767	sample_2768	sample_2769	sample_2770
0	C001	left	ramp	white	no	11.421809	51.632774	49.136442	44.595915	10.767278	...	55.015610	55.195877	55.131669	55.220485	54.954781	55.044311	54.868641	55.012999	55.104108	55.182648
1	C001	left	ramp	blue	no	56.805336	56.765305	56.750474	56.175521	56.930798	...	27.476767	52.824345	53.477177	55.411936	56.438331	56.553257	56.073525	56.063473	55.947377	55.893032
2	C001	left	ramp	green	no	54.326952	54.440178	54.714699	55.499070	56.050834	...	56.577864	56.609074	56.819729	56.770057	56.185188	13.059161	17.648903	55.145064	14.972549	56.430405
3	C001	left	flash	white	no	55.787624	55.933980	55.942581	56.724411	56.978560	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	C001	left	flash	blue	no	59.850759	57.968526	33.227293	58.630396	57.587477	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	C001	left	ramp	red	no	56.235004	55.917564	56.105497	55.888826	56.176029	...	54.264717	54.313963	54.272126	54.166878	53.452621	50.630194	11.557823	27.985472	43.879836	52.511971
6	C001	left	flash	red	no	54.601431	54.581684	54.679883	54.665344	54.648270	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7	C001	left	flash	green	no	56.719736	56.402336	55.909666	56.221283	56.131363	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
8	C001	right	ramp	white	no	46.833160	43.629848	17.446286	18.221857	17.880424	...	50.076435	50.019839	49.955989	51.081802	50.188717	50.096776	50.389408	50.865040	50.432724	49.922916
9	C001	right	ramp	blue	no	51.172456	52.257381	50.993540	43.809630	0.000000	...	51.965121	53.887000	53.470181	52.908485	52.168674	51.882730	52.820683	51.970182	52.340378	52.610977
10	C001	right	ramp	green	no	53.261269	52.815006	51.921143	52.301464	53.014683	...	52.574112	52.401651	53.042474	19.478518	28.944765	25.240248	21.512200	52.318325	52.771439	40.539242
11	C001	right	flash	white	no	53.116306	52.223270	52.493837	52.843510	52.894299	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
12	C001	right	flash	blue	no	51.817524	42.810749	41.951134	0.000000	27.703615	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13	C001	right	ramp	red	no	50.785607	51.976555	50.998297	51.408707	51.415897	...	49.713729	50.439243	11.433658	17.887014	29.686583	0.000000	44.712772	47.161194	13.785289	10.872181
14	C001	right	flash	red	no	50.601871	50.910453	52.519806	52.129833	51.873520	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
15	C001	right	flash	green	no	52.114094	51.123128	50.891659	52.954445	53.138733	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
16	C002	left	flash	blue	no	46.992462	47.032223	47.291660	47.331293	46.999798	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
17	C002	left	flash	green	no	43.740337	43.954097	44.026813	43.975315	44.020817	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
18	C002	left	ramp	white	no	43.590279	43.563667	43.585438	43.768256	43.739961	...	46.119160	46.380466	46.258817	46.609200	46.584278	46.363099	46.287899	46.125832	46.038406	46.237620
19	C002	left	ramp	green	no	44.828008	45.172432	45.174828	45.136540	45.150245	...	44.558022	45.057167	45.337010	45.502069	45.465844	45.411225	45.613084	45.829338	45.780842	45.746876
20	C002	left	flash	white	no	47.000584	47.043674	46.748494	46.854923	46.747399	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
21	C002	left	flash	red	no	47.655375	47.399848	47.908897	47.760640	47.348320	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
22	C002	left	ramp	blue	no	47.135171	47.145790	47.206020	47.369992	47.380413	...	46.100193	46.182297	46.181812	46.059040	45.831283	45.630150	43.862882	44.192247	38.703911	45.278918
23	C002	left	ramp	red	no	45.207959	45.147304	45.294198	45.155163	45.344329	...	46.011948	45.323382	35.470947	43.049096	43.078188	41.886142	37.832001	39.852316	40.017342	16.012384
24	C002	right	flash	blue	no	47.300781	47.429856	47.295403	47.256710	47.217690	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Figura 3.5: Primeras 25 filas del dataset diseñado.

3.3.2.1. Saturación de señales

Además de que las señales contienen bastante ruido, se observan *peaks* en los datos que son extremadamente altos, como se puede ver en la figura 3.6. Este ejemplo corresponde a la respuesta pupilar ante un estímulo de tipo rampa, de color rojo. La respuesta pupilar a un estímulo rampa, en general, debería tener la forma mostrada en la figura 3.7.

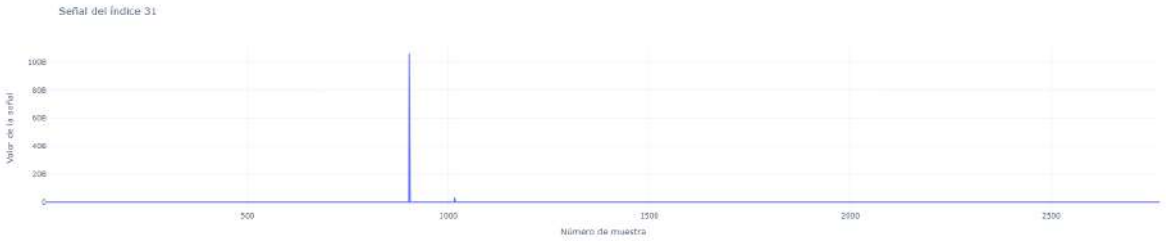

Figura 3.6: Respuesta pupilar ante un estímulo tipo rampa, con un *peak* muy elevado.

Figura 3.7: Respuesta pupilar general ante un estímulo tipo rampa.

En las figuras 3.6 y 3.7 se puede apreciar que la presencia de *peaks* dificulta la visualización adecuada de estas señales y, además, podrían afectar significativamente cualquier análisis o cálculo que se quiera realizar para procesar los datos. Dado esto, se propuso saturar las señales con un valor máximo de diámetro pupilar de 80 píxeles. Este valor fue propuesto de

manera visual, ya que en todas las señales analizadas no se observaron valores importantes (que no sean *peaks*) por encima de este umbral.

Una vez saturadas las señales del *dataset*, se puede observar el mismo ejemplo de la figura 3.6, ahora saturado, en la figura 3.8.

Figura 3.8: Respuesta pupilar general ante un estímulo tipo rampa correspondiente a la misma señal de la figura 3.6, pero saturada con un valor máximo de 80 píxeles.

Por otro lado, se aplicó un proceso similar para las muestras que tomaban el valor 0. Como se puede observar en la figura 3.9, algunas señales presentan valores de 0 píxeles, lo cual es fisiológicamente imposible, ya que la pupila no puede cerrarse completamente.

Figura 3.9: Respuesta pupilar ante un estímulo tipo rampa, con valores 0.

Dado lo anterior, se decidió saturar estos valores a un mínimo de 10 píxeles. Al igual que en la saturación realizada con los valores máximos, este umbral se definió de manera visual, ya que en todas las señales analizadas no se encontraron valores relevantes por debajo de este límite.

Una vez aplicados estos ajustes, se puede observar el mismo ejemplo de la figura 3.9, ahora con la saturación de valores mínimos aplicada, en la figura 3.10.

Figura 3.10: Respuesta pupilar general ante un estímulo tipo rampa correspondiente a la misma señal de la figura 3.9, pero saturada con un valor mínimo de 10 píxeles.

3.3.2.2. Implementación de *padding* a señales tipo *flash*

Como se observa en la figura 3.5, existen valores `NaN` en los datos correspondientes a la clase `flash`. Esto se debe a que, como se explicó en la sección 3.3.2, cada respuesta pupilar ante estímulos lumínicos tiene una duración de 2770 muestras, es decir, 92 segundos. Este valor es adecuado para las respuestas ante estímulos de tipo rampa, que tienen una duración de 1 minuto, más otro de descanso. Sin embargo, los *flashes* solo duran 1 segundo, seguido de 60 segundos de descanso. Por lo tanto, una vez transcurrido ese tiempo, no existen más muestras con las cuales completar el *dataset*, lo que resulta en la aparición de valores nulos (`NaN`). Un ejemplo visual de esta situación se muestra en la figura 3.11.

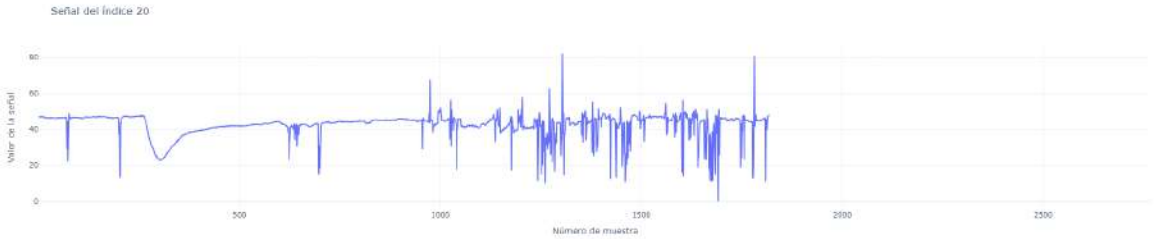

Figura 3.11: Respuesta pupilar general ante un estímulo tipo flash, de color blanco en este caso.

Para evitar problemas en los cálculos que se requieren para procesar los datos, la solución fue implementar una técnica de *padding*, es decir, rellenar artificialmente los datos faltantes. En este caso, se propuso rellenar los valores nulos repitiendo las últimas 300 muestras reales de cada respuesta pupilar ante estímulos de tipo *flash*. Esto permite completar todas las columnas con valores. Al aplicar esta técnica a todo el *dataset*, específicamente a las señales correspondientes a respuestas ante estímulos de tipo *flash*, se obtiene un *dataset* sin valores `NaN`. El ejemplo mostrado en la figura 3.11, después de aplicar *padding*, se puede ver en la figura 3.12.

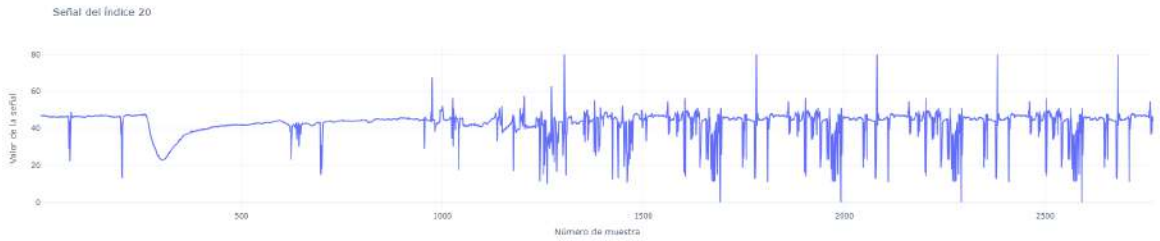

Figura 3.12: Respuesta pupilar general ante un estímulo tipo flash correspondiente al mismo ejemplo visto en la figura 3.11, aplicando *padding*.

3.3.2.3. Filtrado

Como se observa en la figura 3.13, las señales presentan un nivel considerable de ruido. Por ello, en este trabajo se experimentó con dos tipos de *datasets*: uno sin filtrar y otro en el que se aplicó un filtro a las señales.

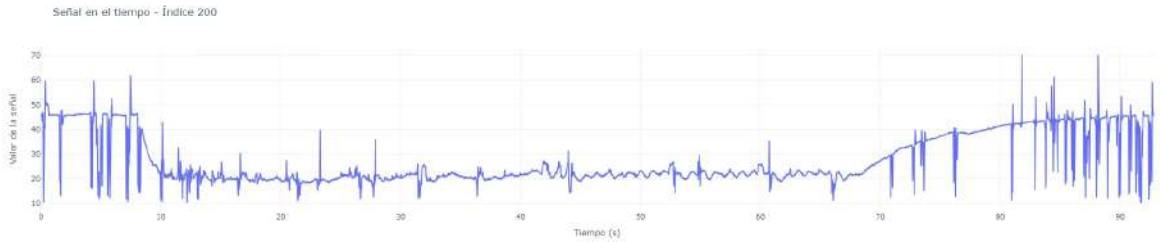

Figura 3.13: Ejemplo de respuesta pupilar general ante un estímulo tipo rampa, sin filtrar.

Para filtrar cada señal del *dataset*, se utilizó un filtro pasabajo con una *frecuencia de Nyquist* igual a 0.5 veces la frecuencia de muestreo (que es 29.84 Hz para todos los casos), una frecuencia de corte de 0.8 Hz y un orden de 5. La misma señal mostrada en la figura 3.13, pero una vez filtrada, se puede observar en la figura 3.14.

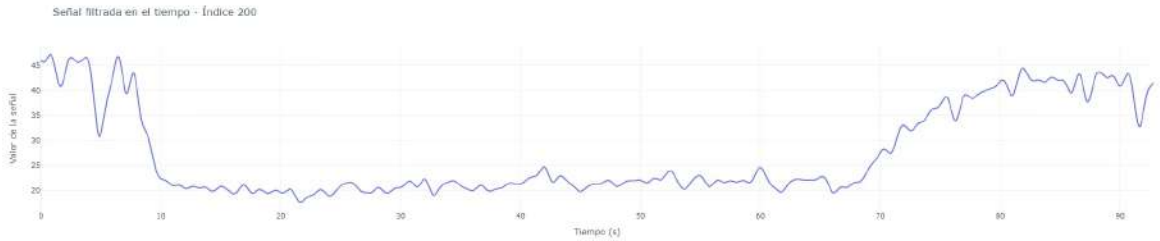

Figura 3.14: Ejemplo de respuesta pupilar general ante un estímulo tipo rampa, correspondiente al mismo ejemplo visto en la figura 3.13, aplicando el filtro pasabajos.

3.4. Implementación de modelos a base de *Machine Learning*

Esta sección describe en detalle el proceso de implementación de los modelos de *Machine Learning*, incluyendo la extracción de características utilizadas para el entrenamiento de distintos modelos. Asimismo, se presentan los resultados obtenidos en cada una de las etapas del desarrollo y evaluación.

3.4.1. Extracción de características

En esta sección se describen las características extraídas del *dataset* generado en la sección 3.3, con el fin de construir vectores de características que representen de manera precisa el comportamiento de las señales pupilares ante estímulos lumínicos.

3.4.1.1. Características estadísticas

Como se mencionó en la sección 2.1.5, las características estadísticas juegan un rol clave en la extracción de información relevante en señales temporales. Además, su implementación se facilita considerablemente utilizando lenguajes de programación especializados en matemáticas estadísticas, como *Python*, que proporciona funciones nativas para estos cálculos [16].

Las características estadísticas utilizadas son:

1. **Promedio:** Es el valor medio de una ventana de datos. Se utiliza para tener una medida general de la tendencia central de la señal en cada ventana.
2. **Desviación estándar:** Mide la cantidad de variación o dispersión de los datos respecto al promedio. Se usa para evaluar la estabilidad de la señal, identificando si los valores están concentrados o dispersos alrededor del promedio.
3. **Mediana:** Es el valor central de los datos ordenados. Se utiliza cuando se desea una medida robusta frente a valores atípicos, ya que es menos sensible a estos que el promedio.
4. **Valor máximo:** El valor más alto dentro de una ventana de datos. Es útil para identificar *peaks* o eventos inusuales en la señal.
5. **Valor mínimo:** El valor más bajo dentro de una ventana de datos. Se utiliza para detectar caídas o puntos bajos significativos en la señal.
6. **Rango:** Es la diferencia entre el valor máximo y el valor mínimo de una ventana. Se emplea para medir la amplitud de variación de la señal en una ventana específica.
7. **Rango intercuartil:** Es la diferencia entre el tercer cuartil (Q3) y el primer cuartil (Q1). Es una medida de dispersión que no se ve afectada por valores extremos, proporcionando una idea del rango en el que se encuentran la mayoría de los datos.
8. **Primer cuartil (Q1):** Es el valor por debajo del cual se encuentra el 25 % de los datos. Se utiliza para evaluar la parte inferior de la distribución de la señal.
9. **Tercer cuartil (Q3):** Es el valor por debajo del cual se encuentra el 75 % de los datos. Se utiliza para evaluar la parte superior de la distribución de la señal.

10. **Curtosis:** Mide la "agudeza" de la distribución de los datos en la ventana. Se utiliza para identificar si los datos tienen muchos valores en los extremos (valores atípicos) o si están más concentrados alrededor de la media.

11. **Asimetría:** Mide la asimetría de la distribución de los datos. Indica si los valores tienden a concentrarse más hacia la derecha o hacia la izquierda de la media, lo que puede revelar patrones sesgados o tendencias específicas en la señal.

Para realizar la extracción de estas características, cada señal fue segmentada en ventanas de tiempo de longitud fija. Posteriormente, para cada ventana, se calcularon las 11 características estadísticas mencionadas. En la figura 3.15, se muestra un ejemplo de la segmentación de una señal en $i+1$ ventanas (N), cada una con una longitud de n. Para cada ventana, se extrajeron las características previamente mencionadas. En este estudio, se realizaron pruebas utilizando ventanas de 25 muestras (0.83 segundos), 50 muestras (1.67 segundos) y 100 muestras (3.35 segundos).

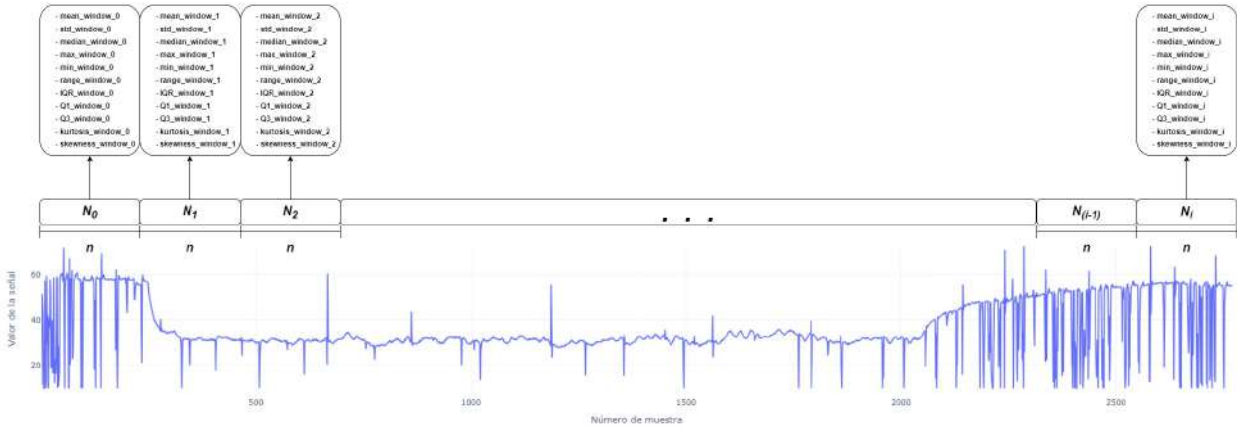

Figura 3.15: Ejemplo de segmentación y extracción de características estadísticas en una señal.

Adicionalmente, se implementó una combinación de ventanas de diferentes longitudes en un mismo *dataset*. En la figura 3.16, se muestra cómo se combinaron ventanas de longitud n (N) y m (M), proporcionando así un análisis más completo de la señal. En este caso, se experimentó con la combinación de ventanas de 50 y 100 muestras.

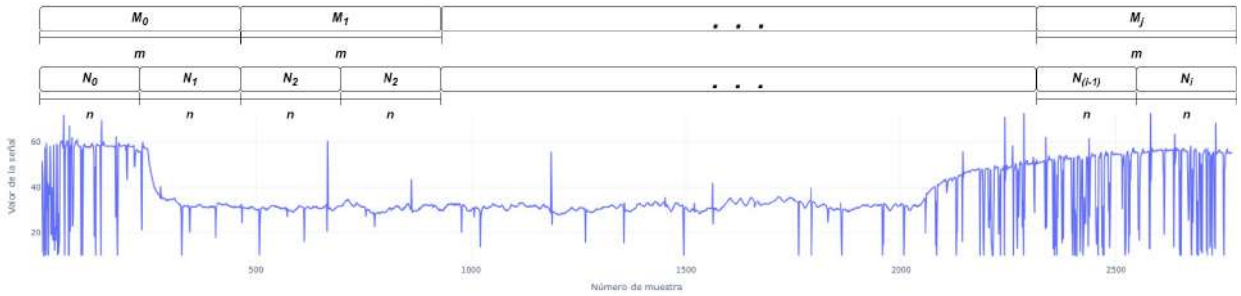

Figura 3.16: Ejemplo de segmentación y extracción de características estadísticas en una señal, usando distintos largos de ventanas.

Para todos los casos, las características obtenidas por cada señal fueron añadidas a nuevos *datasets* de características, compuestos por:

- **`id`:** Código del paciente.
- **`eye`:** Corresponde al ojo del paciente. Los valores pueden ser `left` o `right`.
- **`stimuli`:** Indica el tipo de estímulo. Los valores posibles son `ramp` o `flash`.
- **`color`:** Color del estímulo. Los valores posibles son `red`, `green`, `blue`, o `white`.
- **`glaucoma`:** Indica si el paciente padece glaucoma o no. Los valores pueden ser `yes` o `no`.
- **Características estadísticas:** Correspondientes a las características estadísticas mencionadas en esta sección.

3.4.1.2. Selección de ventanas

Con el objetivo de experimentar con diferentes métodos de preprocesamiento y mejorar el rendimiento de los modelos, se implementaron dos estrategias de selección de ventanas. La primera consiste en seleccionar las características correspondientes a los primeros 20 segundos de la reacción pupilar. La segunda, en cambio, selecciona ventanas de 20 segundos de señal de manera aleatoria.

Para el primer caso, utilizando ventanas de 50 muestras de longitud, se seleccionaron las primeras 12 ventanas. Al trabajar con ventanas de 100 muestras de longitud, se utilizaron las primeras 6 ventanas. Finalmente, en el caso de las ventanas de 25 muestras, se seleccionaron las primeras 24 ventanas.

En el segundo caso, la cantidad de ventanas seleccionadas de forma aleatoria varía según la longitud de la ventana, de manera similar al primer caso.

3.4.1.3. Aumento de datos

Otro enfoque utilizado para mejorar el rendimiento de los modelos fue la técnica de aumento de datos. En este caso, se generó un nuevo *dataset* de características mediante el aumento artificial de la data original. Para ello, se duplicó la matriz de características y se añadió ruido gaussiano a la matriz duplicada. El nivel de ruido agregado fue de 0.01, lo que equivale a un 1 % del valor original de cada característica. Esta estrategia busca aumentar la diversidad de los datos y mejorar la capacidad de generalización de los modelos.

3.4.1.4. Aislamiento de características

Con el objetivo de identificar qué tipos de estímulos lumínicos podrían ser más efectivos para el diagnóstico del glaucoma, se realizaron dos experimentos modificando el *dataset*. En el primer experimento, se utilizaron únicamente las respuestas pupilares ante estímulos de tipo *flash*, seleccionando los primeros 20 segundos de cada señal. En el segundo experimento, se emplearon exclusivamente las respuestas pupilares ante estímulos de tipo rampa, utilizando la señal completa. Estos experimentos permiten evaluar y comparar la efectividad de cada tipo de estímulo en la detección del glaucoma.

3.4.2. Descripción de casos

Como se mencionó anteriormente, para evaluar el rendimiento de los modelos en función de las distintas configuraciones del *dataset*, se definieron y analizaron tres enfoques experimentales:

1. **Caso base:** Se utilizó el conjunto de datos de características original, sin modificaciones adicionales.
2. **Selección de ventanas:**
   a) Se seleccionaron las características correspondientes a los primeros 20 segundos de la reacción pupilar.
   b) Se extrajeron ventanas aleatorias, acumulando un total de 20 segundos de señal.
3. **Aislamiento de características:**
   a) Se seleccionaron exclusivamente los estímulos tipo *flash*, limitados a los primeros 20 segundos.
   b) Se seleccionaron sólo las respuestas a estímulos tipo rampa.

Para cada caso, se evaluaron tres configuraciones del *dataset*:

- Con datos filtrados.
- Con datos aumentados artificialmente.
- Combinando el filtrado y el aumento de datos.

Adicionalmente, en cada configuración se variaron los largos de las ventanas utilizadas para la extracción de características, explorando así cómo estas afectan el rendimiento de los modelos.

3.4.3. Preprocesamiento

Antes de entrenar cualquier modelo, es necesario llevar a cabo una serie de pasos adicionales para garantizar que los modelos de *machine learning* funcionen correctamente. Aunque los vectores de características ya fueron generados, es preciso realizar modificaciones adicionales para adaptar estos datos a los modelos que se utilizarán.

El primer paso, como se explicó en la sección 2.1.3, consiste en dividir el conjunto de datos en datos de entrenamiento y datos de prueba. El conjunto de entrenamiento se utiliza para ajustar los parámetros del modelo, mientras que el conjunto de prueba se emplea para evaluar su rendimiento. En este caso, la división se realizó utilizando una proporción de 80/20, es decir, el 80 % de los datos se utilizó para entrenar los modelos, y el 20 % restante para probar su desempeño. Para evitar *data leakage* (fuga de información), se utilizó el código de cada paciente como criterio de separación, asegurando que las señales correspondientes a un mismo paciente no se distribuyeran en conjuntos distintos. Tras esta división, se eliminó la columna `id` del *dataset*.

Como se mencionó en la sección 3.4.1, los conjuntos de datos contienen tanto características estadísticas como información relevante, como el código del paciente, si este padece glaucoma, el tipo de estímulo lumínico, entre otras. En este contexto, es necesario definir la *etiqueta* o variable objetivo a predecir. Dado que el objetivo es predecir la presencia de glaucoma, la etiqueta seleccionada fue la columna `glaucoma`, una variable categórica que toma los valores "`yes`" o "`no`". Para que los modelos de *machine learning* puedan procesar esta variable, se realizó una binarización, asignando 0 a los pacientes sin glaucoma y 1 a aquellos con glaucoma.

Asimismo, otras variables categóricas como `eye`, `stimuli` y `color` fueron transformadas para ser interpretadas adecuadamente por los modelos. Para evitar que las variables categóricas sean tratadas como continuas (lo cual podría ocurrir con la codificación ordinal), se empleó la técnica de *OneHot Encoding*. Esta técnica transforma cada categoría en una nueva columna binaria que toma el valor de 1 cuando la categoría está presente, tal como se muestra en el ejemplo visual de la codificación de la variable `color` en la figura 3.17.

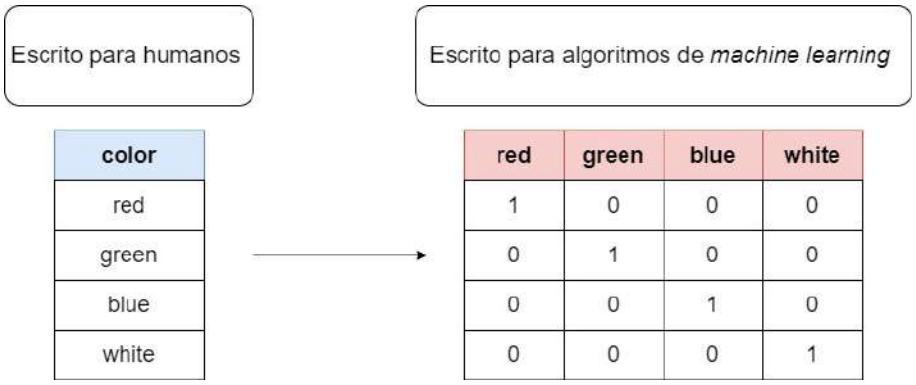

Figura 3.17: Codificación *Onehot* ejecutada en la característica `color`.

Un paso adicional fundamental es el escalamiento de las variables numéricas. El objetivo de este procedimiento es asegurar que todas las variables numéricas tengan un rango similar, lo que mejora el rendimiento de los modelos. En este caso, se utilizó un escalado *min-max*, que conserva la proporción y forma de los datos originales pero es sensible a valores extremos. La ecuación para esta transformación es:

$$x_i' = \frac{x_i - \min(x)}{\max(x) - \min(x)} \tag{3.1}$$

donde x_i representa el valor original de la columna x a escalar, y x_i' es el valor transformado. Esta transformación ajusta los datos para que estén comprendidos en el intervalo [0, 1], facilitando su uso en algoritmos que no requieren que los datos sigan una distribución normal.

Todos estos pasos fueron implementados utilizando un *pipeline*, una herramienta que permite organizar el preprocesamiento y el entrenamiento de los modelos de forma eficiente y reproducible.

3.4.4. Descripción del modelo

Entre los modelos iniciales que se exploraron y probaron se encuentran *XGBoost*, *Random Forest* y *Support Vector Machine* (SVM). Todos estos modelos fueron evaluados inicialmente utilizando la data segmentada en ventanas de 50 muestras, y empleando técnicas de *cross-validation* en el conjunto de entrenamiento para su validación. Tras las pruebas, el modelo *XGBoost* demostró un rendimiento superior, por lo que se decidió emplearlo para las pruebas posteriores.

Los principales hiperparámetros de *XGBoost* con los que se trabajó fueron los siguientes:

1. ***Learning Rate*** ***(η)*****:** Controla el tamaño de las actualizaciones en cada iteración del modelo, ayudando a evitar el sobreajuste.
2. ***Max Depth*****:** Define la profundidad máxima de los árboles y controla la complejidad del modelo. Un valor alto puede llevar al sobreajuste, mientras que un valor bajo puede limitar la capacidad del modelo para capturar patrones complejos.
3. ***Evaluation Metric*****:** Determina la métrica a optimizar durante el entrenamiento.

Además, con el objetivo de usar de referencia, se implementó un modelo *"dummy"*, el cual clasifica los datos sin leerlos, de manera aleatoria.

3.4.5. Entrenamiento

Antes de entrenar el modelo, se utilizó *cross-validation* en el conjunto de entrenamiento, junto con una búsqueda en grilla (*grid search*) para identificar los mejores hiperparámetros. Los valores óptimos encontrados fueron:

- `learning_rate = 0.3`
- `max_depth = 6`
- `eval_metric = ["error", "logloss"]`

Con los hiperparámetros ya definidos, se procedió a entrenar el modelo en el conjunto de entrenamiento. Posteriormente, el modelo fue evaluado utilizando el conjunto de prueba y se calcularon diversas métricas de evaluación para medir su rendimiento en las pruebas.

3.4.6. Resultados *Dummy model*

Probando el modelo aleatorio, se llegó a los resultados de la tabla 3.1.

Tabla 3.1: Resultados al entrenar un modelo *dummy*.

	Precision	**Recall**	**F1-score**	**Support**
0	0.53	0.47	0.50	80
1	0.42	0.47	0.44	64
Accuracy			0.47	144
Macro average	0.47	0.47	0.47	144
Weighted average	0.48	0.47	0.47	144

Las curvas ROC y *precision-recall* se pueden observar en las figuras A.1 y A.2 respectivamente, mientras que la matriz de confusión se puede ver en la figura A.3.

Teniendo estos resultados como base, se procedió a utilizar el modelo *XGBoost*.

3.4.7. Resultados: caso base

En esta sección se muestran los resultados obtenidos al experimentar con las distintas configuraciones de la *data*, utilizando todo el *dataset*.

Los resultados de *accuracy* obtenidos en los distintos casos se presentan en la tabla 3.2, donde **Normal** se refiere a la utilización de las características sin filtrado ni aumento de datos, **Filtrado** corresponde a la utilización de características con filtrado, **D.A.** representa el uso de la matriz de características con aumento de datos, y **Filtrado + D.A.** indica la combinación de ambas técnicas (filtrado y aumento de datos).

Tabla 3.2: *Accuracy* obtenido utilizando todo el *dataset* (Caso base).

	50	**100**	**25**	**50-100**
Normal	0.64	0.65	0.67	0.65
Filtrado	0.60	0.62	0.58	0.65
D.A.	**0.69**	0.62	0.64	0.66
Filtrado + DA	0.61	**0.69**	0.58	0.64

Los dos resultados de mejor rendimiento se obtuvieron en el caso de *D.A.* con ventanas de 50 muestras de longitud y en el caso de *Filtrado + D.A.* con ventanas de 100 muestras de longitud. En el primer caso, se alcanzó un *accuracy* del 68.75 %, mientras que en el segundo caso se logró un *accuracy* del 69.44 %. Para el primer caso, las figuras B.1 a B.5 muestran su matriz de confusión, curva ROC, curva *precision-recall*, así como las curvas de pérdida y error a lo largo de las épocas, respectivamente. De manera similar, para el segundo caso, las figuras B.6 a B.10 ilustran su matriz de confusión, curva ROC, curva *precision-recall*, y las gráficas de pérdida y error a través de las épocas correspondientes, respectivamente. En el anexo D.1, se pueden encontrar todas las matrices de confusión de este caso.

3.4.8. Resultados: selección de ventanas

En esta sección se muestran los resultados obtenidos al seleccionar ventas.

3.4.8.1. Primeros 20 segundos

Al utilizar sólo los primeros 20 segundos de cada señal, se obtuvieron los valores de *accuracy* mostrados en la tabla 3.3.

Tabla 3.3: *Accuracy* obtenido utilizando los primeros 20 segundos de cada señal.

	50	**100**	**25**	**50-100**
Normal	0.62	0.57	0.61	0.65
Filtrado	0.57	0.61	0.62	0.62
D.A.	0.62	0.62	**0.69**	**0.67**
Filtrado + DA	0.62	0.65	0.60	0.66

Los dos resultados de mejor rendimiento se obtuvieron en el caso de *D.A.* con ventanas de 25 muestras de longitud y en el caso de *D.A.* con ventanas de 50 y 100 muestras de longitud. En el primer caso, se alcanzó un *accuracy* del 68.75 %, mientras que en el segundo caso se logró un *accuracy* del 67.36 %. Para el primer caso, las figuras B.11 a B.15 muestran su matriz de confusión, curva ROC, curva *precision-recall*, así como las curvas de pérdida y error a lo largo de las épocas, respectivamente. De manera similar, para el segundo caso, las figuras B.16 a B.20 ilustran su matriz de confusión, curva ROC, curva *precision-recall*, y las gráficas de pérdida y error a través de las épocas correspondientes, respectivamente. En el anexo D.2.1, se pueden encontrar todas las matrices de confusión de este caso.

3.4.8.2. Ventanas aleatorias

Al utilizar ventanas aleatorias correspondiente a 20 segundos de cada señal, se obtuvieron los valores de *accuracy* mostrados en la tabla 3.4.

Tabla 3.4: *Accuracy* obtenido utilizando ventanas aleatorias.

	50	**100**	**25**	**50-100**
Normal	0.53	0.59	**0.67**	0.57
Filtrado	0.56	0.55	0.60	0.62
D.A.	0.51	0.54	**0.66**	0.56
Filtrado + DA	0.58	0.56	0.63	0.62

Los dos resultados de mejor rendimiento se obtuvieron en el caso de *Normal* con ventanas de 25 muestras de longitud y en el caso de *D.A.* con ventanas de 25 muestras de longitud. En el primer caso, se alcanzó un *accuracy* del 66.67 %, mientras que en el segundo caso se logró un *accuracy* del 65.97 %. Para el primer caso, las figuras B.21 a B.25 muestran su matriz de confusión, curva ROC, curva *precision-recall*, así como las curvas de pérdida y error a lo largo de las épocas, respectivamente. De manera similar, para el segundo caso, las figuras B.26 a B.30 ilustran su matriz de confusión, curva ROC, curva *precision-recall*, y las gráficas de pérdida y error a través de las épocas correspondientes, respectivamente. En el anexo D.2.2, se pueden encontrar todas las matrices de confusión de este caso.

3.4.9. Resultados: aislamiento de características

En esta sección se muestran los resultados obtenidos al sólo usar un tipo de estímulo por cada prueba.

3.4.9.1. Respuestas ante estímulos tipo *flash*

Al utilizar sólo las respuestas pupilares ante estímulos lumínicos tipo *flash*, se obtuvieron los valores de *accuracy* mostrados en la tabla 3.5. En este caso, se hizo una prueba adicional en donde se utilizaron ventanas de 15 muestras de largo debido a la corta duración de este tipo de estímulo.

Tabla 3.5: *Accuracy* obtenido utilizando estímulos lumínicos tipo *flash*.

	50	**100**	**25**	**50-100**	**15**
Normal	0.65	0.57	**0.72**	0.69	0.58
Filtrado	0.65	0.61	0.62	0.65	0.67
D.A.	0.69	0.56	0.65	0.68	0.62
Filtrado + DA	**0.72**	0.65	0.61	0.65	0.65

Los dos resultados de mejor rendimiento se obtuvieron en el caso de *Normal* con ventanas de 25 muestras de longitud y en el caso de *Filtrado + D.A.* con ventanas de 50 muestras de longitud. En el primer caso, se alcanzó un *accuracy* del 72.22 %, mientras que en el segundo caso se logró un *accuracy* del 72.22 %. Para el primer caso, las figuras B.31 a B.35 muestran su matriz de confusión, curva ROC, curva *precision-recall*, así como las curvas de pérdida y error a lo largo de las épocas, respectivamente. De manera similar, para el segundo caso, las figuras B.36 a B.40 ilustran su matriz de confusión, curva ROC, curva *precision-recall*, y las gráficas de pérdida y error a través de las épocas correspondientes, respectivamente. En el anexo D.3.1, se pueden encontrar todas las matrices de confusión de este caso.

3.4.9.2. Respuestas ante estímulos tipo rampa

Al utilizar sólo las respuestas pupilares ante estímulos lumínicos tipo rampa, se obtuvieron los valores de *accuracy* mostrados en la tabla 3.6. En este caso, se hizo una prueba adicional en donde se utilizaron ventanas de 150 muestras de largo debido a la larga duración de este tipo de estímulo.

Tabla 3.6: *Accuracy* obtenido utilizando estímulos lumínicos tipo rampa.

	50	**100**	**25**	**50-100**	**150**
Normal	0.56	0.64	0.61	0.61	0.60
Filtrado	0.62	0.69	0.53	0.61	**0.79**
D.A.	0.53	0.58	0.58	0.64	0.57
Filtrado + DA	0.56	**0.76**	0.56	0.62	0.75

Los dos resultados de mejor rendimiento se obtuvieron en el caso de *Filtrado* con ventanas de 150 muestras de longitud y en el caso de *Filtrado + D.A.* con ventanas de 100 muestras de longitud. En el primer caso, se alcanzó un *accuracy* del 79.17 %, mientras que en el segundo caso se logró un *accuracy* del 76.39 %. Para el primer caso, las figuras B.41 a B.45 muestran su matriz de confusión, curva ROC, curva *precision-recall*, así como las curvas de pérdida y error a lo largo de las épocas, respectivamente. De manera similar, para el segundo caso, las figuras B.46 a B.50 ilustran su matriz de confusión, curva ROC, curva *precision-recall*, y las

gráficas de pérdida y error a través de las épocas correspondientes, respectivamente. En el anexo D.3.2, se pueden encontrar todas las matrices de confusión de este caso.

3.5. Implementación de modelos a base de *Deep Learning*

En esta sección se describe el proceso de implementación y los resultados obtenidos al aplicar modelos basados en *Deep Learning*. Para esta implementación, se trabajó directamente con las señales obtenidas (descritas en la sección 3.3), es decir, sin realizar extracción de características. En total, se llevaron a cabo seis experimentos con distintas configuraciones de los datos:

1. **Caso base:** uso del *dataset* completo de señales con saturación y *padding*, sin aplicar filtro.
2. **Caso filtrado:** uso del *dataset* completo de señales con saturación y *padding*, aplicando un filtro pasabajos.
3. **Caso rampa:** uso exclusivo de las señales correspondientes a respuestas pupilares ante estímulos de tipo rampa, con saturación y *padding*, sin aplicar filtro.
4. **Caso *flash*:** uso exclusivo de las señales correspondientes a respuestas pupilares ante estímulos de tipo *flash*, con saturación y *padding*, sin aplicar filtro.
5. **Caso rampa filtrado:** uso exclusivo de las señales correspondientes a respuestas pupilares ante estímulos de tipo rampa, con saturación y *padding*, aplicando un filtro pasabajos.
6. **Caso *flash* filtrado:** uso exclusivo de las señales correspondientes a respuestas pupilares ante estímulos de tipo *flash*, con saturación y *padding*, aplicando un filtro pasabajos.

Se experimentó con dos tipos de modelos de *Deep Learning*: una *Fully Convolutional Network* (FCN) y una red neuronal recurrente con un mecanismo de compuerta de Unidades Recurrentes Cerradas (GRU), evaluando el rendimiento en cada una de las configuraciones mencionadas.

3.5.1. Entrenamiento

Para entrenar los modelos, se dividió el *dataset* en tres conjuntos: un 60 % de los datos se utilizó para el entrenamiento, un 20 % para la validación y el 20 % restante se reservó para la prueba final del modelo.

El modelo *Fully Convolutional Network* (FCN) se entrenó a lo largo de 1000 épocas, comenzando con una tasa de aprendizaje inicial de 0.001. Esta tasa se redujo en un factor de 0.5 cada 50 épocas, alcanzando un mínimo de 0.0001. Se utilizó el modelo con el mayor *accuracy* en el conjunto de validación para seleccionar la mejor versión del modelo durante el entrenamiento. El tamaño de lote (*batch size*) se fijó en 16.

Por otro lado, el modelo basado en unidades recurrentes de tipo GRU (*Gated Recurrent Unit*) se entrenó durante 100 épocas, con una tasa de aprendizaje inicial de 0.001, que se redujo en un factor de 0.5 cada 50 épocas hasta alcanzar un mínimo de 0.0001. Para este modelo, se seleccionó la versión con la menor pérdida en el conjunto de validación. Al igual que en el entrenamiento de FCN, se utilizó un *batch size* de 16.

3.5.2. Resultados: FCN

A continuación, se presentan los resultados obtenidos al entrenar el modelo *Fully Convolutional Network* (FCN) en las distintas configuraciones de datos.

3.5.2.1. Caso base

En el caso base, el modelo FCN alcanzó un *accuracy* de 75.72 % en el conjunto de validación y un *accuracy* de 50.00 % en el conjunto de prueba. La matriz de confusión para el conjunto de validación se presenta en la figura C.1, y la correspondiente al conjunto de prueba se muestra en la figura C.2. Además, se incluyen las métricas adicionales para evaluar el rendimiento del modelo: la curva ROC en la figura C.3, la curva *precision-recall* en la figura C.4, las curvas de pérdida a lo largo de las épocas en la figura C.5, y las curvas de error a través de las épocas de entrenamiento en la figura C.6.

3.5.2.2. Caso filtrado

En el caso filtrado, el modelo FCN alcanzó un *accuracy* de 69.90 % en el conjunto de validación y un *accuracy* de 50.69 % en el conjunto de prueba. La matriz de confusión para el conjunto de validación se presenta en la figura C.7, y la correspondiente al conjunto de prueba se muestra en la figura C.8. Además, se incluyen las métricas adicionales para evaluar el rendimiento del modelo: la curva ROC en la figura C.9, la curva *precision-recall* en la figura C.10, las curvas de pérdida a lo largo de las épocas en la figura C.11, y las curvas de error a través de las épocas de entrenamiento en la figura C.12.

3.5.2.3. Caso rampa

En el caso rampa, el modelo FCN alcanzó un *accuracy* de 80.76 % en el conjunto de validación y un *accuracy* de 58.33 % en el conjunto de prueba. La matriz de confusión para el conjunto de validación se presenta en la figura C.13, y la correspondiente al conjunto de prueba se muestra en la figura C.14. Además, se incluyen las métricas adicionales para evaluar el rendimiento del modelo: la curva ROC en la figura C.15, la curva *precision-recall* en la figura C.16, las curvas de pérdida a lo largo de las épocas en la figura C.17, y las curvas de error a través de las épocas de entrenamiento en la figura C.18.

3.5.2.4. Caso *flash*

En el caso *flash*, el modelo FCN alcanzó un *accuracy* de 80.76 % en el conjunto de validación y un *accuracy* de 54.16 % en el conjunto de prueba. La matriz de confusión para el conjunto de validación se presenta en la figura C.19, y la correspondiente al conjunto de prueba se muestra en la figura C.20. Además, se incluyen las métricas adicionales para evaluar el rendimiento del modelo: la curva ROC en la figura C.21, la curva *precision-recall* en la figura C.22, las curvas de pérdida a lo largo de las épocas en la figura C.23, y las curvas de error a través de las épocas de entrenamiento en la figura C.24.

3.5.2.5. Caso rampa filtrado

En el caso rampa filtrado, el modelo FCN alcanzó un *accuracy* de 92.00 % en el conjunto de validación y un *accuracy* de 43.05 % en el conjunto de prueba. La matriz de confusión para el conjunto de validación se presenta en la figura C.25, y la correspondiente al conjunto de prueba se muestra en la figura C.26. Además, se incluyen las métricas adicionales para evaluar el rendimiento del modelo: la curva ROC en la figura C.27, la curva *precision-recall* en la figura C.28, las curvas de pérdida a lo largo de las épocas en la figura C.29, y las curvas de error a través de las épocas de entrenamiento en la figura C.30.

3.5.2.6. Caso *flash* filtrado

En el caso *flash* filtrado, el modelo FCN alcanzó un *accuracy* de 82.69 % en el conjunto de validación y un *accuracy* de 59.72 % en el conjunto de prueba. La matriz de confusión para el conjunto de validación se presenta en la figura C.31, y la correspondiente al conjunto de prueba se muestra en la figura C.32. Además, se incluyen las métricas adicionales para evaluar el rendimiento del modelo: la curva ROC en la figura C.33, la curva *precision-recall* en la figura C.34, las curvas de pérdida a lo largo de las épocas en la figura C.35, y las curvas de error a través de las épocas de entrenamiento en la figura C.36.

3.5.3. Resultados: GRU

A continuación, se presentan los resultados obtenidos al entrenar el modelo *Gated Recurrent Unit* (GRU) en las distintas configuraciones de datos.

3.5.3.1. Caso base

En el caso base, el modelo GRU alcanzó un *accuracy* de 75.72 % en el conjunto de validación y un *accuracy* de 57.63 % en el conjunto de prueba. La matriz de confusión para el conjunto de validación se presenta en la figura C.37, y la correspondiente al conjunto de prueba se muestra en la figura C.38. Además, se incluyen las métricas adicionales para evaluar el rendimiento del modelo: la curva ROC en la figura C.39, la curva *precision-recall* en la figura C.40, las curvas de pérdida a lo largo de las épocas en la figura C.41, y las curvas de error a través de las épocas de entrenamiento en la figura C.42.

3.5.3.2. Caso filtrado

En el caso filtrado, el modelo GRU alcanzó un *accuracy* de 60.19 % en el conjunto de validación y un *accuracy* de 61.11 % en el conjunto de prueba. La matriz de confusión para el conjunto de validación se presenta en la figura C.43, y la correspondiente al conjunto de prueba se muestra en la figura C.44. Además, se incluyen las métricas adicionales para evaluar el rendimiento del modelo: la curva ROC en la figura C.45, la curva *precision-recall* en la figura C.46, las curvas de pérdida a lo largo de las épocas en la figura C.47, y las curvas de error a través de las épocas de entrenamiento en la figura C.48.

3.5.3.3. Caso rampa

En el caso rampa, el modelo GRU alcanzó un *accuracy* de 71.15 % en el conjunto de validación y un *accuracy* de 47.22 % en el conjunto de prueba. La matriz de confusión para el conjunto de validación se presenta en la figura C.49, y la correspondiente al conjunto de prueba se muestra en la figura C.50. Además, se incluyen las métricas adicionales para evaluar el rendimiento del modelo: la curva ROC en la figura C.51, la curva *precision-recall* en la

figura C.52, las curvas de pérdida a lo largo de las épocas en la figura C.53, y las curvas de error a través de las épocas de entrenamiento en la figura C.54.

3.5.3.4. Caso *flash*

En el caso *flash*, el modelo GRU alcanzó un *accuracy* de 57.69 % en el conjunto de validación y un *accuracy* de 47.22 % en el conjunto de prueba. La matriz de confusión para el conjunto de validación se presenta en la figura C.55, y la correspondiente al conjunto de prueba se muestra en la figura C.56. Además, se incluyen las métricas adicionales para evaluar el rendimiento del modelo: la curva ROC en la figura C.57, la curva *precision-recall* en la figura C.58, las curvas de pérdida a lo largo de las épocas en la figura C.59, y las curvas de error a través de las épocas de entrenamiento en la figura C.60.

3.5.3.5. Caso rampa filtrado

En el caso rampa filtrado, el modelo GRU alcanzó un *accuracy* de 61.53 % en el conjunto de validación y un *accuracy* de 58.33 % en el conjunto de prueba. La matriz de confusión para el conjunto de validación se presenta en la figura C.61, y la correspondiente al conjunto de prueba se muestra en la figura C.62. Además, se incluyen las métricas adicionales para evaluar el rendimiento del modelo: la curva ROC en la figura C.63, la curva *precision-recall* en la figura C.64, las curvas de pérdida a lo largo de las épocas en la figura C.65, y las curvas de error a través de las épocas de entrenamiento en la figura C.66.

3.5.3.6. Caso *flash* filtrado

En el caso *flash* filtrado, el modelo GRU alcanzó un *accuracy* de 69.23 % en el conjunto de validación y un *accuracy* de 54.16 % en el conjunto de prueba. La matriz de confusión para el conjunto de validación se presenta en la figura C.67, y la correspondiente al conjunto de prueba se muestra en la figura C.68. Además, se incluyen las métricas adicionales para evaluar el rendimiento del modelo: la curva ROC en la figura C.69, la curva *precision-recall* en la figura C.70, las curvas de pérdida a lo largo de las épocas en la figura C.71, y las curvas de error a través de las épocas de entrenamiento en la figura C.72.

3.5.4. Resumen de resultados

En esta sección se presentan de manera resumida los resultados obtenidos, destacando únicamente los valores de *accuracy* en los distintos casos al implementar los modelos de *Deep Learning*. Estos valores se pueden observar en la tabla 3.7.

Tabla 3.7: *Accuracy* obtenido en cada caso al implementar los modelos a base de *Deep Learning*.

	FCN		**GRU**	
	Validación (%)	**Prueba (%)**	**Validación (%)**	**Prueba (%)**
Caso base	75.72	50.00	75.72	57.63
Caso filtrado	69.90	50.69	60.19	61.11
Caso rampa	80.76	58.33	71.15	47.22
Caso flash	80.76	54.16	57.69	47.22
Caso rampa filtrado	92.00	43.05	61.53	58.33
Caso flash filtrado	82.69	59.72	69.23	54.16

Capítulo 4

Análisis y discusión

En este capítulo se presentan y analizan los resultados obtenidos, considerando tanto los distintos modelos implementados como las configuraciones de datos y los diferentes casos de estudio aplicados. El análisis busca identificar patrones de desempeño y evaluar el impacto de cada configuración sobre los resultados finales, brindando una comprensión más profunda del efecto de los estímulos lumínicos en el diagnóstico automatizado de glaucoma.

4.1. Análisis General de Machine Learning

En las tablas 3.2, 3.3, 3.4, 3.5 y 3.6, se puede observar una variabilidad considerable en los resultados obtenidos, los cuales varían según la longitud de las ventanas empleadas y las diferentes configuraciones del *dataset*. Estos factores parecen influir de manera diversa en el rendimiento de los modelos, reflejándose en métricas de precisión que muestran diferencias notables entre los distintos experimentos realizados.

Si analizamos las distribuciones de precisión para cada configuración del *dataset* en forma global, como se ilustra en la figura 4.1, observamos que no existe una correlación fuerte entre la configuración específica del *dataset* y los valores de precisión obtenidos. Por ejemplo, en la configuración **Filtrado**, la mediana es comparable a la de otras configuraciones, pero con una menor variabilidad. A pesar de esta menor dispersión, esta configuración presenta valores atípicos que incluyen tanto uno de los peores como uno de los mejores casos registrados en el análisis, lo que sugiere una sensibilidad significativa a ciertas combinaciones de parámetros.

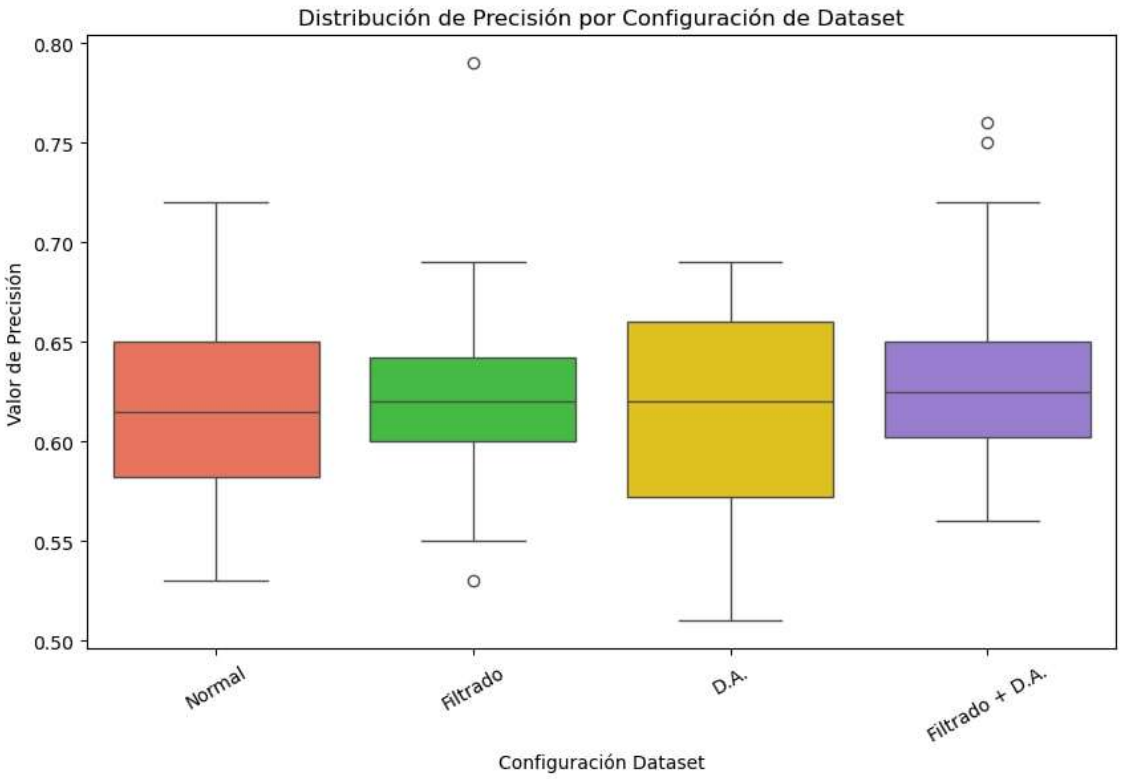

Figura 4.1: Distribución de precisión por tipo de configuración del *dataset*.

Esta variabilidad sugiere que, aunque la configuración del *dataset* tiene un impacto, no es el único factor determinante. Esto implica que el análisis detallado de los resultados es más efectivo al examinar cada caso en particular en lugar de basarse solo en el análisis general. En las siguientes secciones, se explora el comportamiento específico de cada configuración en función de sus resultados de precisión y se identifican los elementos que contribuyen al rendimiento de cada configuración.

Al analizar la distribución de precisión obtenida para cada caso, como se muestra en la figura 4.2, se observan diferencias notables en los resultados. Las distribuciones del **Caso base** y el caso de **Primeros 20 segundos** presentan similitudes, aunque su mediana varía ligeramente, siendo la del **Caso base** marginalmente superior. Por otro lado, el caso de **Ventanas aleatorias** muestra el rendimiento más bajo en términos de precisión general, con una distribución significativamente inferior y la mediana más baja de todos los casos evaluados.

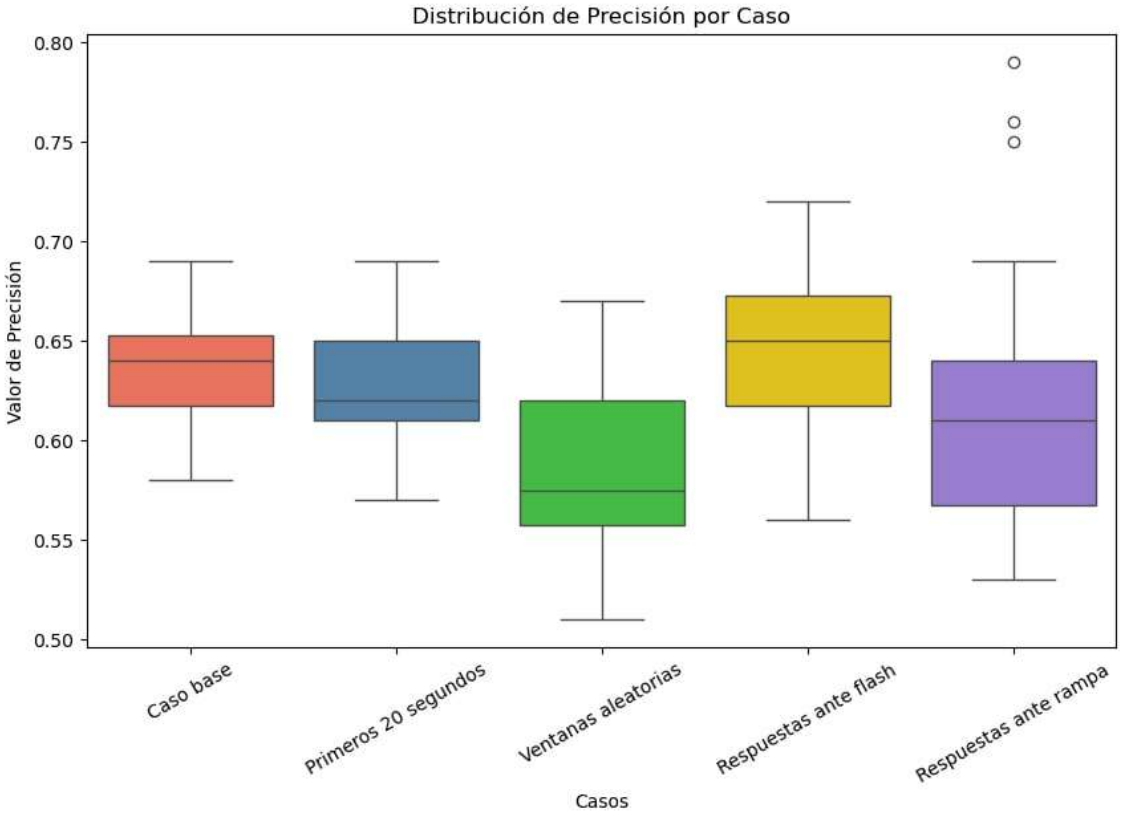

Figura 4.2: Distribución de precisión por cada caso implementado.

El caso de **Respuestas ante flash** destaca como el más efectivo, con la distribución más alta en precisión y alcanzando la mediana más elevada, lo que sugiere que trabajar exclusivamente con respuestas a estímulos de tipo *flash* podría conducir a mejores resultados en promedio. En cambio, el caso de **Respuestas ante rampa** muestra una distribución más dispersa, con una mediana relativamente baja en comparación con otros casos. Sin embargo, este caso contiene valores atípicos que corresponden a algunos de los mejores resultados individuales, lo que indica que, aunque menos consistente, puede ofrecer casos de alta precisión en situaciones específicas.

Al analizar la distribución de precisión en función del largo de las ventanas, como se muestra en la figura 4.3, se observa una considerable variabilidad entre los distintos tamaños de ventana. Esta dispersión puede explicarse debido a la inclusión de todos los casos posibles en el análisis, lo cual introduce diferencias en las características de las señales capturadas en cada ventana. Esta variabilidad sugiere que el largo de la ventana puede tener un impacto significativo en el rendimiento del modelo y que es importante considerar este factor de manera detallada al seleccionar el tamaño de ventana óptimo para el análisis de precisión.

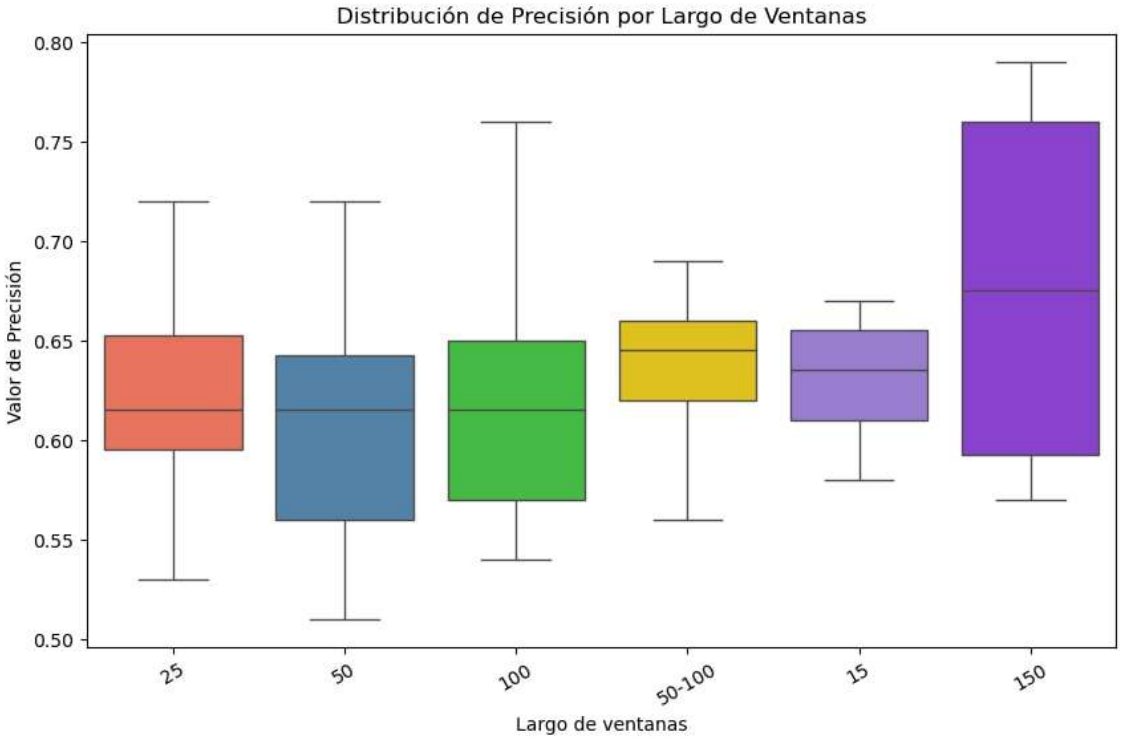

Figura 4.3: Distribución de precisión por largo de ventanas para todos los casos.

4.1.1. Análisis por casos

En esta sección se presentan los resultados de cada caso de manera individual, permitiendo un análisis detallado y específico. Esta aproximación facilita la identificación de patrones, fortalezas y debilidades en cada configuración implementada, brindando una comprensión más profunda del comportamiento de los modelos en diferentes escenarios.

4.1.1.1. Caso Base

Al observar la tabla 3.2, se identifica que los mejores resultados en términos de precisión se obtuvieron al aplicar técnicas de aumento de datos ($D.A.$) y/o filtrado de las señales. Estas configuraciones mejoran la capacidad predictiva del modelo en comparación con el uso de datos sin tratamiento adicional.

En las figuras B.1 y B.6 se presentan las matrices de confusión para los casos con aumento de datos y con aumento más filtrado, respectivamente. En ambas configuraciones, se observa una mayor precisión en la predicción de los pacientes de control en comparación con los pacientes con glaucoma. Esto sugiere que el modelo tiene una tendencia a clasificar mejor los casos de control, posiblemente debido a las características específicas de las señales de los pacientes con glaucoma.

Las curvas ROC, presentadas en las figuras B.7 y B.2, muestran una concavidad sobre la diagonal de referencia, lo que indica un rendimiento superior al azar para ambos modelos. La forma cóncava sugiere que el modelo mantiene una capacidad razonable para distinguir entre ambas clases, aunque el área bajo la curva (AUC) no alcanza valores óptimos, lo que

deja margen para mejoras adicionales en sensibilidad y especificidad.

En las figuras B.8 y B.3 se observan las curvas *Precision-Recall*. Estas curvas evidencian un equilibrio entre precisión y sensibilidad en los modelos, lo cual es favorable para un problema en el que ambas métricas son importantes. Sin embargo, la precisión y el *recall* no alcanzan los valores más altos posibles, lo que señala limitaciones en la capacidad del modelo para recuperar correctamente todas las instancias de glaucoma sin perder precisión.

Las curvas de pérdida para los conjuntos de entrenamiento y prueba, mostradas en las figuras B.9 y B.4, evidencian una falta de convergencia en ambos casos. Esta tendencia sugiere que los modelos no generalizan bien, posiblemente debido a la variabilidad de las señales y a la poca cantidad de datos.

Las figuras B.10 y B.5 muestran las curvas de error en los conjuntos de entrenamiento y prueba. En ambos casos, el error en el conjunto de prueba se mantiene elevado, lo cual refuerza la observación de que los modelos enfrentan dificultades para generalizar.

4.1.1.2. Caso Primeros 20 segundos

Como se observa en la tabla 3.3, los mejores resultados corresponden al uso de ventanas de 25 muestras con aumento de datos y a la combinación de ventanas de 50 y 100 muestras también con aumento de datos. Para el primer caso, este rendimiento superior puede atribuirse a que las ventanas más pequeñas capturan eficientemente la información clave en los primeros momentos de la reacción pupilar, caracterizada por su relativa rapidez y cambio. Por otro lado, la combinación de ventanas de diferentes tamaños permite una visión general y detallada simultáneamente, capturando patrones tanto específicos como generales de la señal.

Dado que los análisis de los parámetros evaluados (matrices de confusión, curvas ROC, y curvas *precision-recall*) presentan un comportamiento similar al del Caso Base, en esta sección se omitirán los detalles específicos de cada métrica. En general, los resultados reflejan tendencias análogas, con un equilibrio entre precisión y sensibilidad en los modelos entrenados bajo esta configuración.

4.1.1.3. Caso Ventanas aleatorias

Como se mencionó anteriormente en la figura 4.2, este caso presenta la distribución más baja de precisión entre los distintos experimentos realizados. Sin embargo, es importante destacar que los mejores resultados obtenidos en esta configuración son comparables a los de otros casos, como se muestra en la tabla 3.4. En este caso, los mejores desempeños se lograron con ventanas de 25 muestras, tanto en la configuración sin aumento de datos ni filtrado como en la configuración con aumento de datos.

Dado que este experimento sigue un enfoque aleatorio, su interpretación es menos directa, y los resultados pueden variar significativamente. Este caso se implementó principalmente como una base de comparación con el Caso de Primeros 20 segundos, permitiendo observar cómo la selección aleatoria de ventanas influye en el rendimiento del modelo en contraste con una selección sistemática.

4.1.1.4. Caso Respuesta ante *flash*

Los resultados obtenidos en el análisis de estímulos tipo *flash* reflejan un comportamiento interesante en términos de precisión y distribución. Según la tabla 3.5, los mejores desempeños se lograron utilizando el conjunto de datos sin alteraciones con ventanas de 25 muestras de largo, así como aplicando filtrado y aumento de datos con ventanas de 50 muestras de largo. Cabe destacar que estos casos corresponden a los de mejor distribución de precisión, como se muestra en la figura 4.2, aunque no representan necesariamente los valores de precisión más altos absolutos.

El buen desempeño de las ventanas de menor tamaño puede atribuirse a la naturaleza de los estímulos *flash*, los cuales son más breves en duración. Esto permite que las ventanas más pequeñas capturen de manera eficiente las características más relevantes de la señal, enfocándose en los momentos críticos de la respuesta pupilar. Sin embargo, como se observa en los resultados obtenidos con ventanas de 15 muestras de largo, existe un límite inferior en el tamaño de ventana que puede ser utilizado sin comprometer la calidad de la descripción de la señal. Este límite se manifiesta en una pérdida de información crítica que resulta en un desempeño subóptimo.

En términos generales, este caso pone de manifiesto la importancia de ajustar el tamaño de las ventanas a la duración y naturaleza de los estímulos lumínicos, maximizando la capacidad del modelo para capturar patrones relevantes en los datos. Esto resalta que una configuración bien equilibrada entre el tamaño de ventana y el preprocesamiento aplicado es esencial para obtener los mejores resultados posibles.

4.1.1.5. Caso Respuesta ante rampa

Como se observa en la tabla 3.6, los mejores resultados en este caso corresponden a dos configuraciones específicas: (1) la utilización de filtrado y aumento de datos con ventanas de 100 muestras de largo, y (2) la utilización de filtrado con ventanas de 150 muestras de largo. Entre estas configuraciones, destaca especialmente el segundo caso, ya que logra el mejor desempeño entre todos los experimentos realizados, alcanzando una precisión de 0.79. No obstante, como se discutió anteriormente en la figura 4.2, estos resultados corresponden a valores atípicos dentro de una distribución más amplia. Esto podría atribuirse a la variabilidad inherente de los datos o al impacto combinado del filtrado y el tamaño de las ventanas.

El mejor resultado se obtiene utilizando ventanas de 150 muestras con filtrado, lo cual resulta coherente, dado que las respuestas a estímulos tipo rampa tienen una duración relativamente larga y estable en comparación con otros estímulos. Este tamaño de ventana permite capturar patrones completos y detallados de la señal, optimizando así la capacidad del modelo para identificar características relevantes.

La figura B.41 presenta la matriz de confusión correspondiente a este caso, mostrando que el modelo tiene una notable capacidad para diferenciar entre pacientes control y pacientes con glaucoma. En particular, se observan 34 clasificaciones correctas para el grupo control (85.00 %) y 23 clasificaciones correctas para el grupo glaucoma (71.88 %), con un número reducido de errores en ambas categorías (6 y 9, respectivamente).

En la figura B.42, se aprecia una curva ROC con un AUC de 0.82, lo que indica una

excelente discriminación entre clases. La curva se mantiene significativamente alejada de la diagonal de aleatoriedad, lo que reafirma la robustez del modelo en esta configuración. De igual manera, la figura B.43 muestra una curva *Precision-Recall* con un AUC-PR de 0.82, destacando un balance sólido entre precisión y recall, lo que resulta crucial para garantizar una identificación precisa de pacientes con glaucoma.

El éxito del modelo en este caso puede explicarse por varios factores. Por un lado, las ventanas de 150 muestras capturan patrones completos y estables, lo que resulta especialmente útil para estímulos tipo rampa debido a su mayor duración. Por otro lado, el filtrado de las señales reduce el ruido presente en los datos, facilitando la extracción de características relevantes y, en consecuencia, mejorando la precisión del modelo.

A pesar del alto desempeño alcanzado, es importante destacar que la figura 4.3 evidencia una amplia distribución de precisiones para las ventanas de 150 muestras. Esto sugiere que no todos los experimentos logran este nivel de rendimiento, lo que subraya la necesidad de seguir evaluando configuraciones adicionales para mejorar la estabilidad y la consistencia de los resultados obtenidos.

4.1.2. Análisis de características

En esta sección se analizan las características más relevantes de los mejores casos, evaluando tanto las ventanas temporales como los tipos de métricas estadísticas más influyentes en el desempeño de los modelos. Este análisis se basa en la extracción de la importancia de las características, medida en forma de peso, para cada caso. Posteriormente, se elaboraron dos rankings: uno para identificar las ventanas temporales más importantes y otro para determinar los tipos de características estadísticas más relevantes. Esto permite extraer información clave sobre cómo los modelos priorizan distintos aspectos de la señal pupilar para realizar sus predicciones.

4.1.2.1. Caso Base

El mejor resultado para el Caso Base se obtuvo con ventanas de 100 muestras de largo, utilizando filtrado y aumento de datos. La figura 4.4 destaca que las ventanas más relevantes identificadas por el modelo son las número 2, 3 y 1, correspondientes a los intervalos de tiempo de 10.05 a 13.40 s, 13.40 a 16.76 s y 6.70 a 10.05 s, respectivamente. Esto se refleja claramente en las figuras 4.5 y 4.6, que muestran respuestas pupilares ante estímulos de tipo rampa y *flash*, respectivamente.

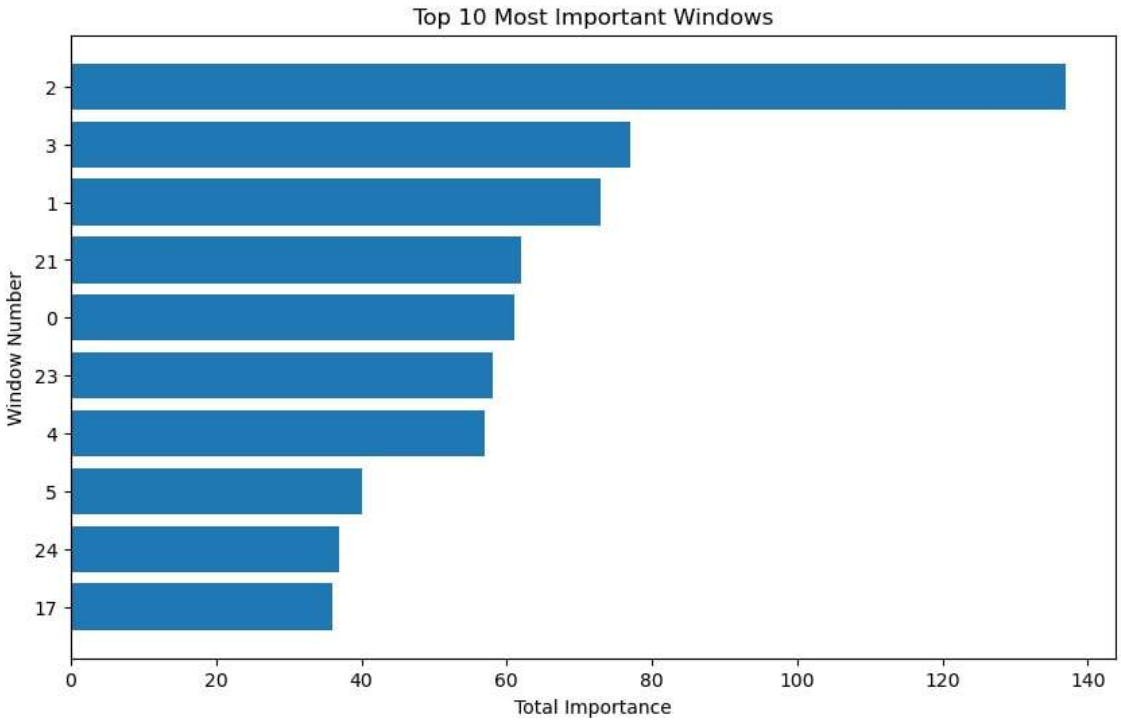

Figura 4.4: Ventanas más importantes para el mejor modelo de Caso Base.

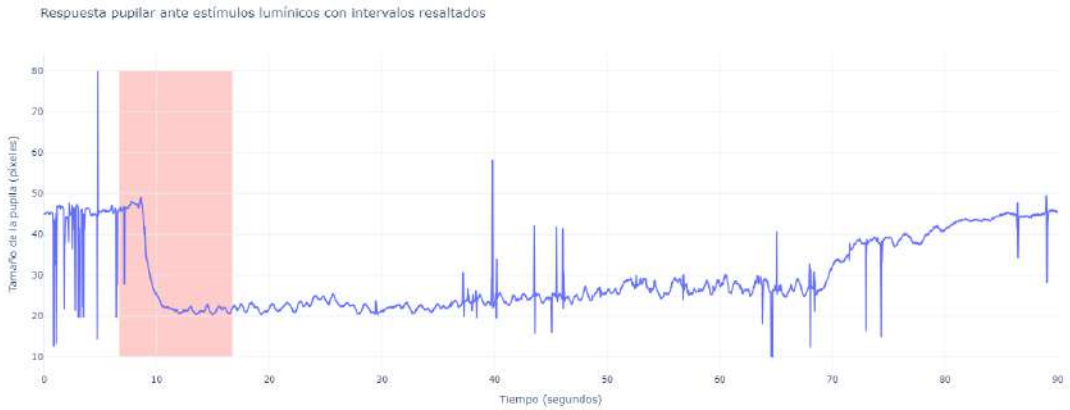

Figura 4.5: Ventanas más importantes del mejor modelo del Caso Base sobre una señal de reacción pupilar ante un estímulo tipo rampa.

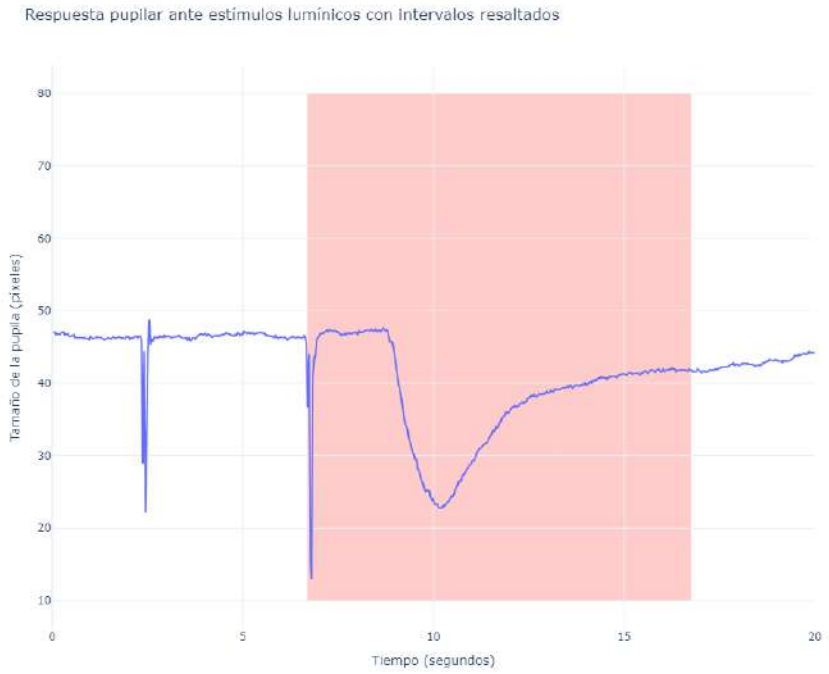

Figura 4.6: Ventanas más importantes del mejor modelo del Caso Base sobre una señal de reacción pupilar ante un estímulo tipo *flash*.

En cuanto a las métricas estadísticas, la figura 4.7 resalta la importancia de la *kurtosis*, *skewness* e IQR, métricas que describen la forma, asimetría y dispersión central de las señales, elementos clave para diferenciar patrones en las respuestas pupilares.

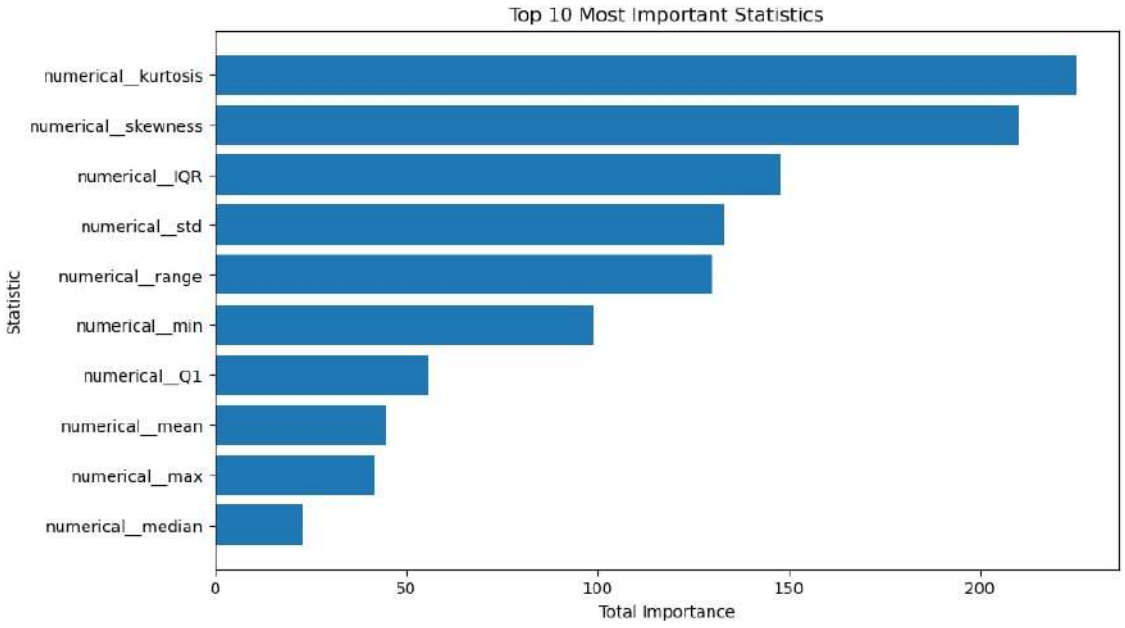

Figura 4.7: Características estadísticas más importantes del mejor modelo de Caso Base.

4.1.2.2. Caso Primeros 20 segundos

El mejor desempeño en este caso corresponde a ventanas de 25 muestras de largo, utilizando aumento de datos. La figura 4.8 indica que las ventanas más importantes son las número 11, 10 y 14, cubriendo los intervalos de tiempo de 10.05 a 10.89 s, 9.21 a 10.05 s y 12.57 a 13.40 s, como se observa en las figuras 4.9 (rampa) y 4.10 (*flash*).

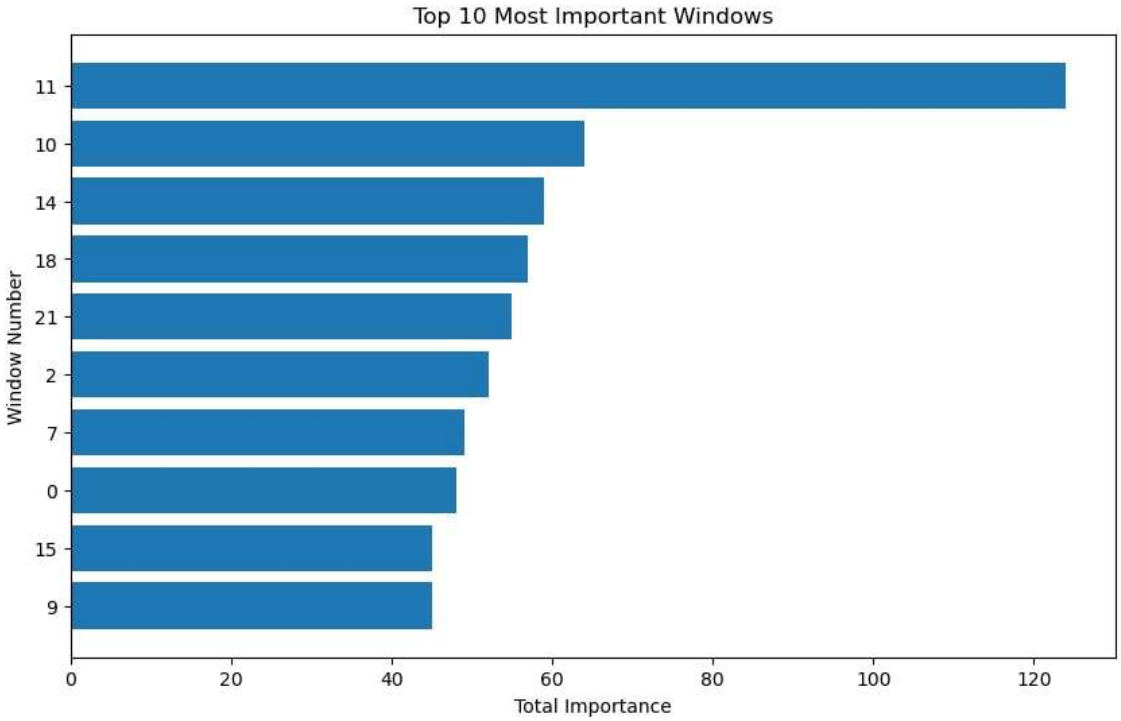

Figura 4.8: Ventanas más importantes para el mejor modelo de Caso Primeros 20 segundos.

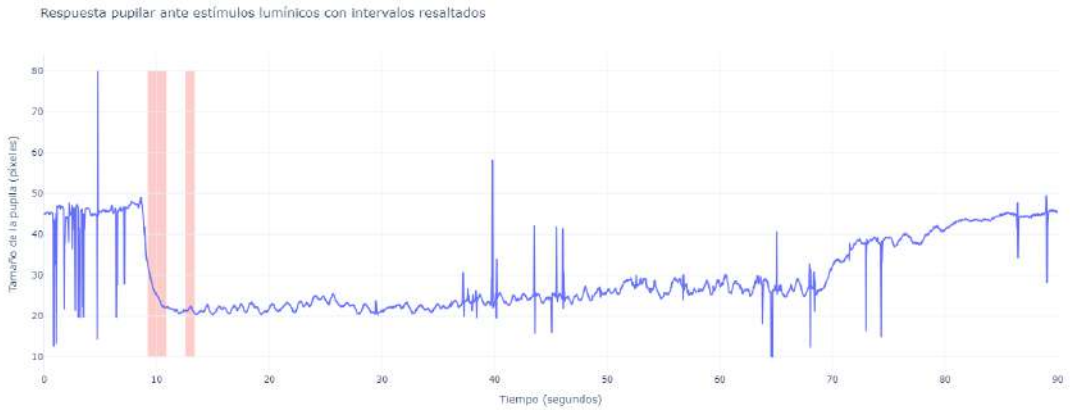

Figura 4.9: Ventanas más importantes del mejor modelo del Caso Primeros 20 segundos sobre una señal de reacción pupilar ante un estímulo tipo rampa.

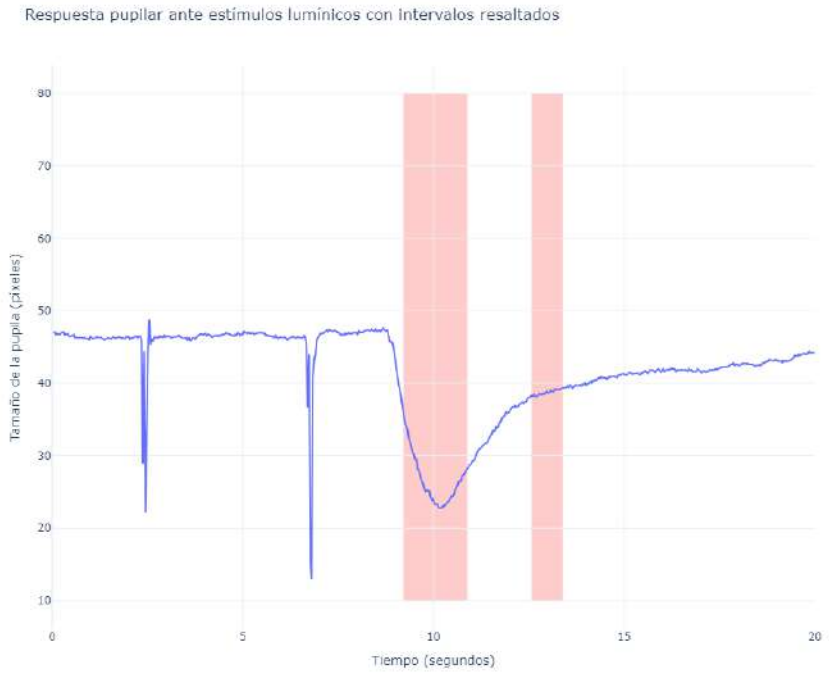

Figura 4.10: Ventanas más importantes del mejor modelo del Caso Primeros 20 segundos sobre una señal de reacción pupilar ante un estímulo tipo *flash*.

Las características estadísticas más relevantes, según la figura 4.11, son el mínimo, IQR y desviación estándar, lo que sugiere que la variabilidad y los valores extremos de la señal desempeñan un papel central en este caso.

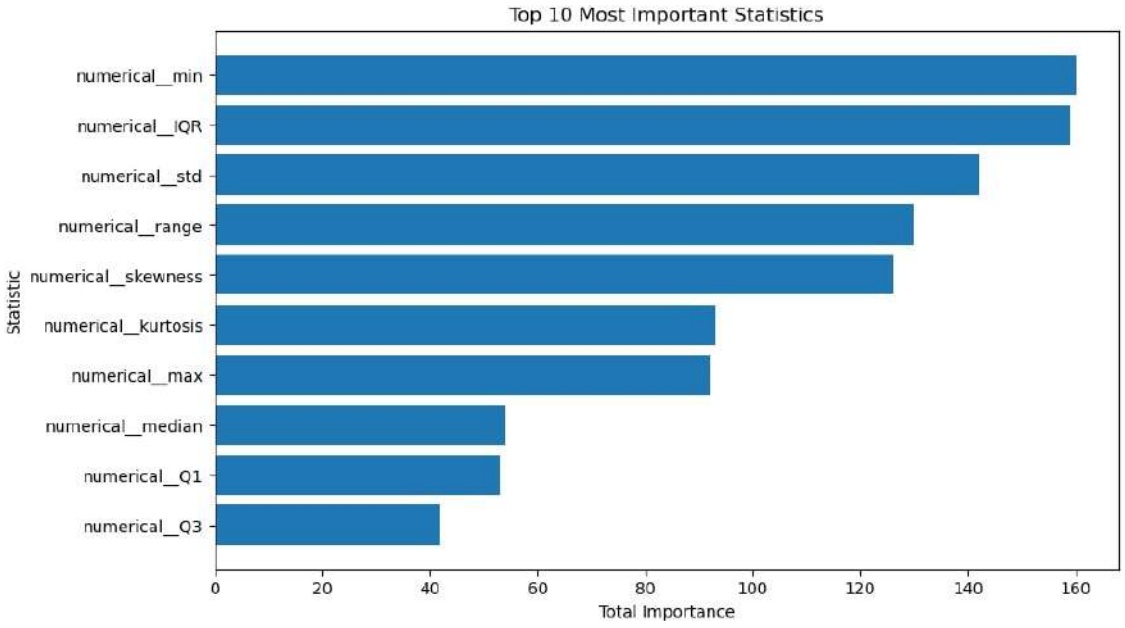

Figura 4.11: Características estadísticas más importantes del mejor modelo de Caso Primeros 20 segundos.

4.1.2.3. Caso Respuesta ante *flash*

Para este caso, el mejor resultado se logró con ventanas de 25 muestras sin modificar el *dataset*. Las ventanas más importantes, según la figura 4.12, son las número 11, 2 y 10, asociadas a los intervalos de tiempo de 10.05 a 10.89 s, 2.51 a 3.35 s y 9.21 a 10.05 s, respectivamente, como se muestra en la figura 4.13.

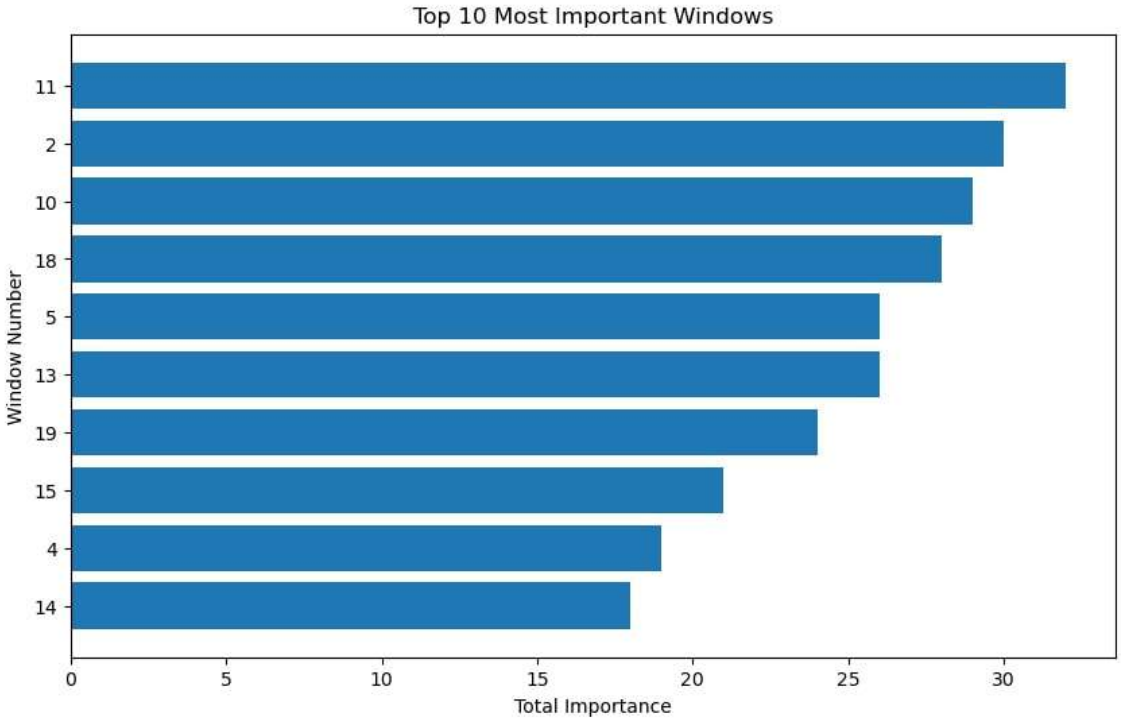

Figura 4.12: Ventanas más importantes para el mejor modelo de Caso Respuesta ante *flash*.

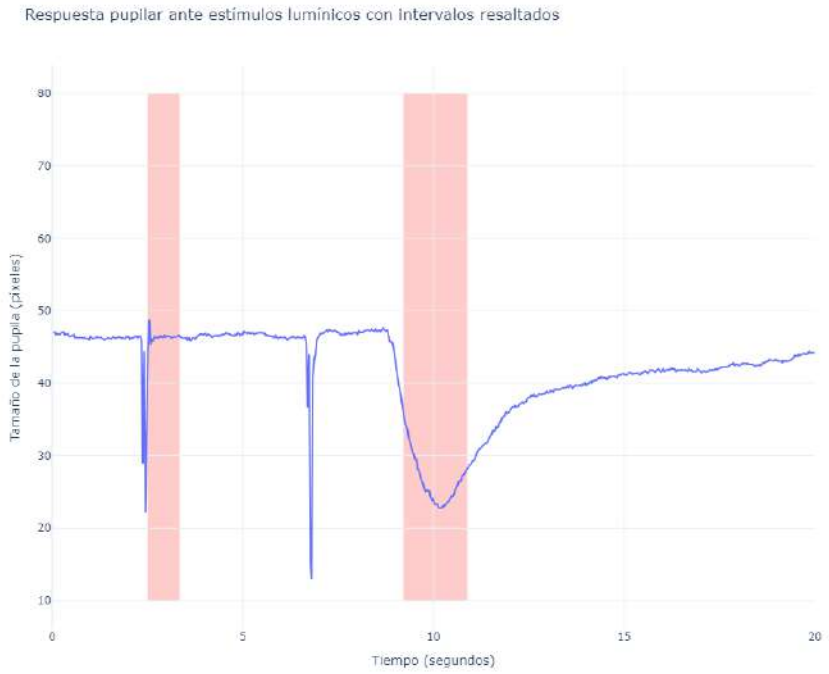

Figura 4.13: Ventanas más importantes del mejor modelo del Caso Respuesta ante *flash* sobre una señal de reacción pupilar ante un estímulo tipo *flash*.

En términos de características estadísticas, la figura 4.14 resalta la importancia de skewness, mínimo e IQR, lo que refuerza la relevancia de analizar la forma y la dispersión de la distribución de la señal pupilar.

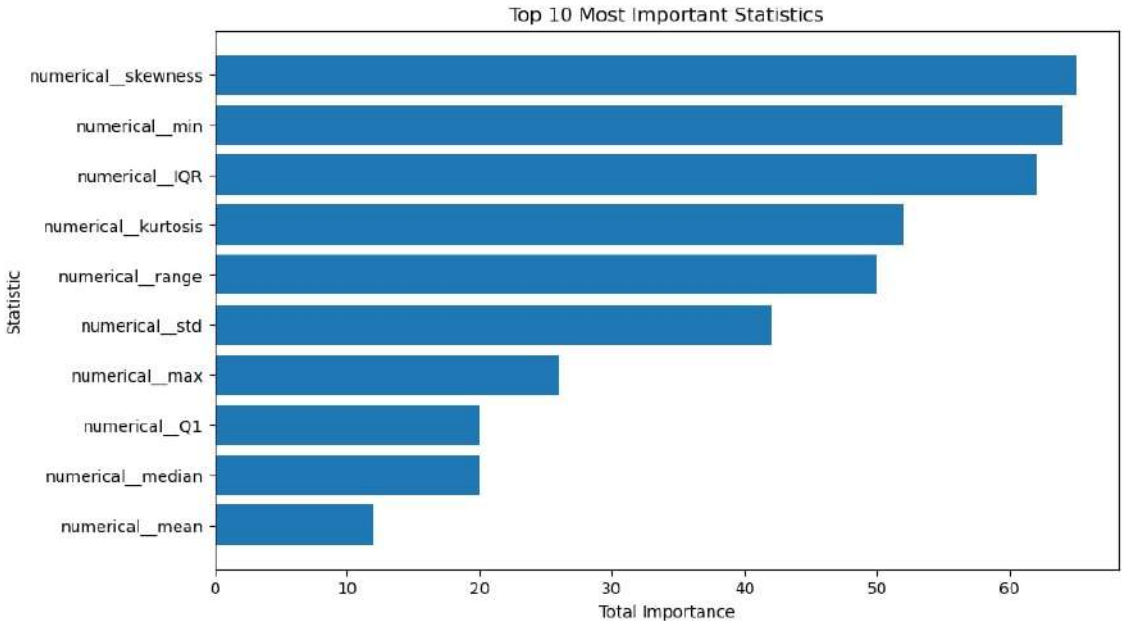

Figura 4.14: Características estadísticas más importantes del mejor modelo de Caso Respuesta ante *flash*.

4.1.2.4. Caso Respuesta ante rampa

El mejor resultado en este caso se obtuvo con ventanas de 150 muestras de largo, utilizando filtrado. La figura 4.15 identifica las ventanas más importantes como las número 2, 14 y 5, correspondientes a los intervalos de 15.08 a 20.11 s, 75.42 a 80.45 s y 30.17 a 35.19 s, como se observa en la figura 4.16. Esto refleja cómo el modelo aprovecha la mayor duración y estabilidad de las señales de estímulos tipo rampa.

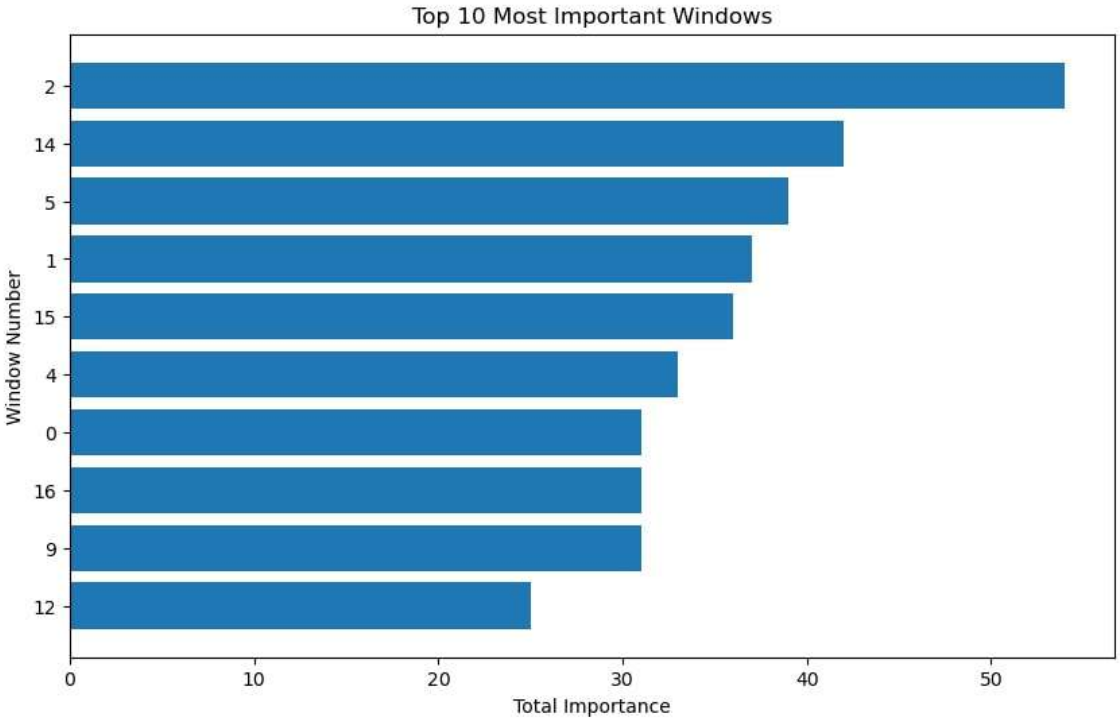

Figura 4.15: Ventanas más importantes para el mejor modelo de Caso Respuesta ante rampa.

Figura 4.16: Ventanas más importantes del mejor modelo del Caso Respuesta ante rampa sobre una señal de reacción pupilar ante un estímulo tipo rampa.

En cuanto a las métricas estadísticas, la figura 4.17 destaca el IQR, kurtosis y skewness como las características más relevantes, indicando que el modelo se centra en aspectos de dispersión, forma y asimetría para identificar patrones en señales prolongadas.

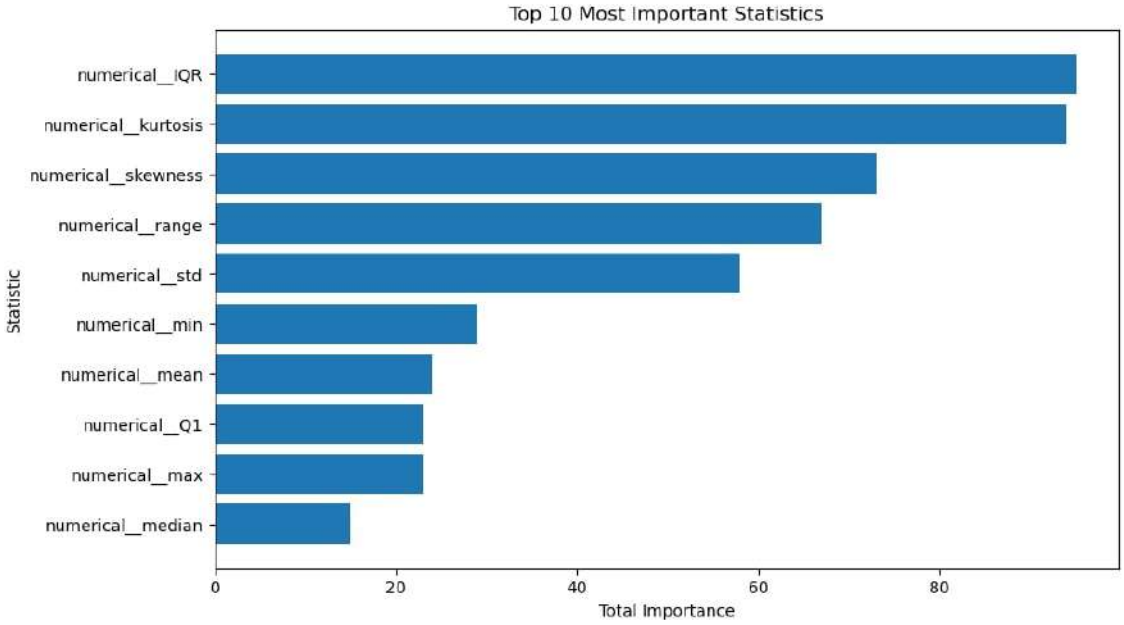

Figura 4.17: Características estadísticas más importantes del mejor modelo de Caso Respuesta ante rampa.

4.1.2.5. Observaciones generales

Al analizar las figuras 4.5, 4.6, 4.9, 4.10, 4.13 y 4.16, se observa que la mayoría de las ventanas destacadas como más importantes se encuentran concentradas en los momentos iniciales de la respuesta pupilar. Estos intervalos coinciden con los cambios significativos en la dimensión de la pupila, que suelen ocurrir inmediatamente después del estímulo lumínico. Este comportamiento resalta la capacidad del modelo para enfocar su atención en los eventos críticos de las señales, que contienen la mayor cantidad de información relevante para la clasificación.

En el Caso Base, los momentos más importantes identificados corresponden a los primeros segundos después del estímulo, cuando la pupila experimenta una contracción rápida antes de estabilizarse. De manera similar, en el Caso Primeros 20 segundos, las ventanas clave se concentran dentro de los primeros segundos de la reacción pupilar, donde el estímulo inicial genera la mayor variación en la señal. Este patrón se mantiene también en el Caso Respuesta ante *flash*, donde la naturaleza breve de los estímulos *flash* hace que los eventos relevantes se concentren en el inicio de la señal, permitiendo al modelo capturar de manera efectiva estos cambios abruptos.

Al combinar todos los intervalos de tiempo (ventanas) más relevantes en un único gráfico, se obtienen las figuras 4.18 para estímulos tipo rampa y 4.19 para estímulos tipo *flash*. En estos gráficos, se observa que la mayoría de las ventanas destacadas como más importantes están concentradas en los momentos iniciales de la respuesta pupilar, coincidiendo con las fases de mayor variación.

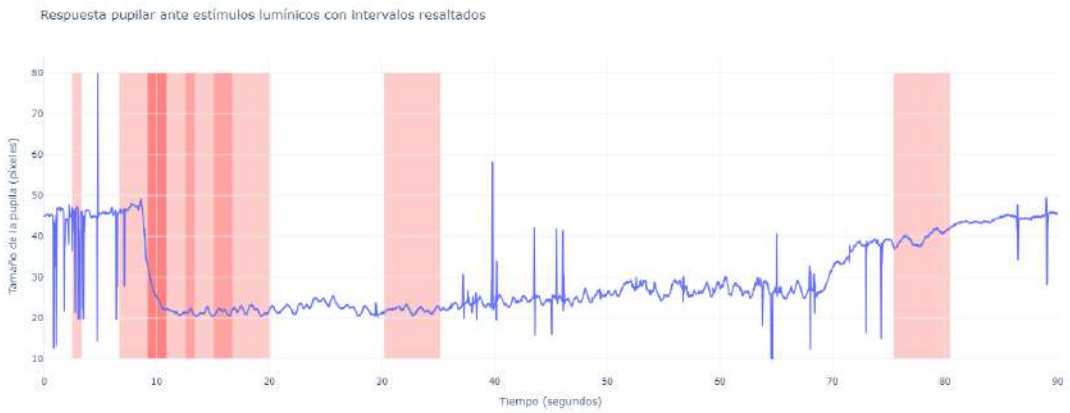

Figura 4.18: Ventanas más importantes de los mejores modelos por caso sobre una señal de reacción pupilar ante un estímulo tipo rampa.

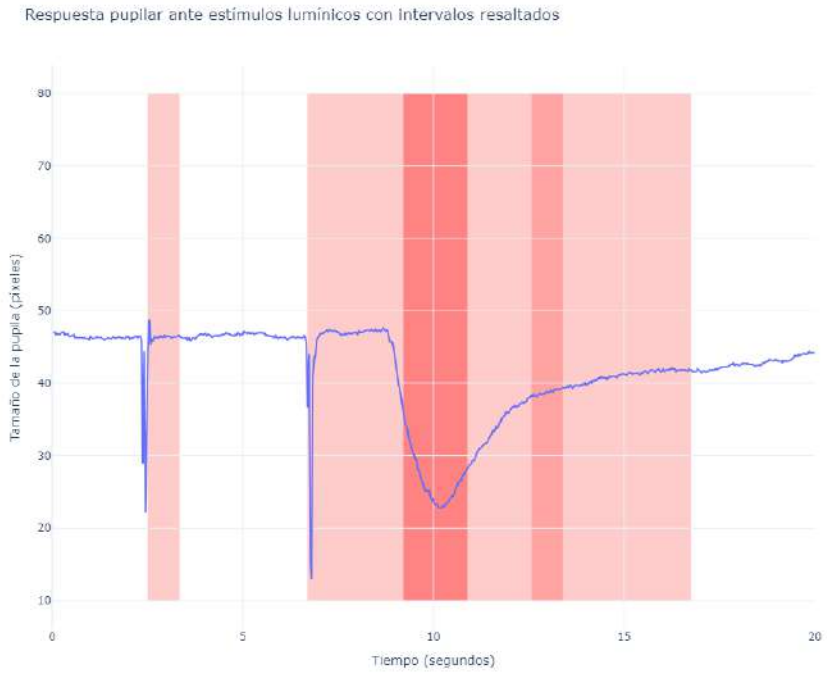

Figura 4.19: Ventanas más importantes de los mejores modelos por caso sobre una señal de reacción pupilar ante un estímulo tipo *flash*.

No obstante, una excepción notable se encuentra en las ventanas asociadas al mejor modelo del caso Respuesta ante rampa, donde las ventanas relevantes abarcan tanto los momentos iniciales como fases posteriores de la señal. Esto es consistente con la naturaleza de los estímulos tipo rampa, los cuales generan respuestas más sostenidas y menos abruptas. En este caso, las ventanas más relevantes se distribuyen tanto en los intervalos iniciales como en los periodos de estabilización y recuperación de la pupila, lo que refleja la capacidad del modelo para adaptarse a las características específicas de este tipo de señal.

Este análisis sugiere que, aunque el modelo demuestra una alta sensibilidad para identificar los momentos iniciales de cambio en todos los casos, en estímulos de larga duración como las rampas lumínicas, también es capaz de identificar patrones relevantes en otras fases de la respuesta pupilar. Esto subraya la importancia de ajustar el tamaño de las ventanas y los enfoques de preprocesamiento según el tipo de estímulo para maximizar el rendimiento del modelo.

Por otra parte, en cuanto a las características estadísticas, este análisis sugiere que las métricas estadísticas como el IQR (Rango Intercuartílico), la *kurtosis* y la *skewness* tienen un papel consistente en la predicción del modelo. El IQR mide la amplitud de la dispersión de los valores centrales de la señal, lo que es clave para identificar cambios significativos que ocurren en los intervalos críticos de la respuesta pupilar. La *kurtosis*, que evalúa la agudeza o prominencia de los picos de la distribución, ayuda al modelo a identificar eventos excepcionales o patrones poco frecuentes en las señales. Por su parte, la *skewness*, que mide la asimetría de la distribución, permite al modelo captar irregularidades o desviaciones en la forma de la señal.

Estos hallazgos resaltan que estas métricas no solo describen propiedades estadísticas de las señales, sino que también capturan características fisiológicamente relevantes, ayudando al modelo a distinguir entre patrones asociados a respuestas normales y aquellas indicativas de glaucoma. Incorporar estas métricas permite aprovechar de manera efectiva la información contenida en las señales pupilares, mejorando la capacidad predictiva del modelo al enfocarse en los aspectos más relevantes para el diagnóstico.

4.2. Análisis General de *Deep Learning*

En esta sección se examina el rendimiento y los resultados obtenidos a partir de los modelos implementados utilizando técnicas de *Deep Learning*. El objetivo principal es interpretar de manera más profunda la efectividad de los datos empleados y comprender cómo estas técnicas contribuyen al análisis y clasificación de las respuestas pupilares. Se presentan las métricas de rendimiento, las visualizaciones relevantes y las observaciones clave derivadas de los resultados obtenidos, proporcionando una visión integral de la utilidad de este enfoque en el contexto del estudio.

4.2.1. Análisis FCN

Al analizar las matrices de confusión correspondientes a los datos de validación, presentadas en la sección 3.5.2, se observa que, en su mayoría (exceptuando el Caso Base), los modelos muestran un mejor rendimiento para clasificar sujetos con glaucoma en comparación con sujetos control. Este comportamiento es opuesto a lo obtenido al aplicar técnicas de *Machine Learning*, donde la clasificación de sujetos control fue generalmente más precisa. Sin embargo, al examinar las matrices de confusión para los datos de prueba, se evidencia un rendimiento cercano al aleatorio. Esto se ve reflejado en las curvas ROC obtenidas, las cuales siguen una trayectoria apenas superior, o incluso inferior, a la diagonal de aleatoriedad, indicando una baja capacidad discriminativa del modelo en este contexto.

En lo que respecta a las curvas de pérdida mostradas en la sección 3.5.2, se puede notar que

las curvas de pérdida de validación tienden a converger hacia las curvas de entrenamiento, lo que sugiere que el modelo está aprendiendo a diferenciar entre las clases. No obstante, estas curvas presentan un nivel considerable de ruido, lo cual puede atribuirse a la alta variabilidad en los datos y a la limitada cantidad disponible para el entrenamiento.

De manera similar, las curvas de error exhiben una separación constante entre las curvas de entrenamiento y validación, lo que podría interpretarse como una estabilidad relativa en el aprendizaje. Sin embargo, estas curvas también presentan una fluctuación significativa, probablemente causada por la misma variabilidad inherente a los datos. En general, aunque el modelo FCN muestra indicios de aprendizaje, el ruido y las limitaciones en los datos de entrenamiento afectan negativamente su rendimiento, particularmente en los datos de prueba.

4.2.2. Análisis GRU

Al analizar las matrices de confusión correspondientes a los datos de validación presentadas en la sección 3.5.3, se observa que el modelo GRU muestra un rendimiento generalmente superior en comparación con los resultados obtenidos en los datos de prueba. Sin embargo, las matrices de confusión para los datos de prueba revelan una distribución cercana a lo aleatorio, indicando que el modelo no logra generalizar adecuadamente. Esto también se refleja en las curvas ROC, que en algunos casos presentan trayectorias apenas por encima de la diagonal de aleatoriedad e incluso por debajo de esta en otros, evidenciando una limitada capacidad discriminativa del modelo.

En cuanto a las curvas de pérdida, estas muestran un comportamiento significativamente más estable y suave en comparación con las curvas observadas en la implementación de FCN. En la mayoría de los casos, las curvas de pérdida para el conjunto de validación tienden a converger hacia las curvas de pérdida del conjunto de entrenamiento durante las primeras épocas. Sin embargo, una vez que dejan de converger, las pérdidas permanecen en valores altos, lo que sugiere la presencia de sobreajuste. Este fenómeno indica que el modelo logra aprender los patrones presentes en los datos de entrenamiento, pero no es capaz de extrapolarlos eficazmente a los datos de validación.

Por otro lado, las curvas de error reflejan comportamientos contrastantes entre los conjuntos de datos. Para los datos de entrenamiento, el error desciende de manera constante a lo largo de las épocas, lo que indica un progreso en el aprendizaje del modelo. Sin embargo, en el caso de los datos de validación, el error permanece casi constante con reducciones mínimas en la mayoría de los casos, lo que refuerza la hipótesis de que el modelo no logra generalizar de manera efectiva. Este comportamiento subraya la necesidad de explorar técnicas adicionales, como regularización o aumento de datos, para mitigar el sobreajuste y mejorar el rendimiento en los datos de prueba.

4.2.3. Observaciones generales

Al analizar la tabla 3.7, se observa que en la mayoría de los casos los modelos de *Deep Learning* presentan un mejor rendimiento en los datos de validación que en los datos de prueba. Esta tendencia es aún más evidente al observar la figura 4.20, que muestra la distribución de precisiones para ambos modelos de *Deep Learning* (FCN y GRU) en los conjuntos de validación y prueba. En esta figura se destaca la diferencia notable entre las distribuciones,

donde las precisiones en los datos de prueba suelen ser considerablemente menores.

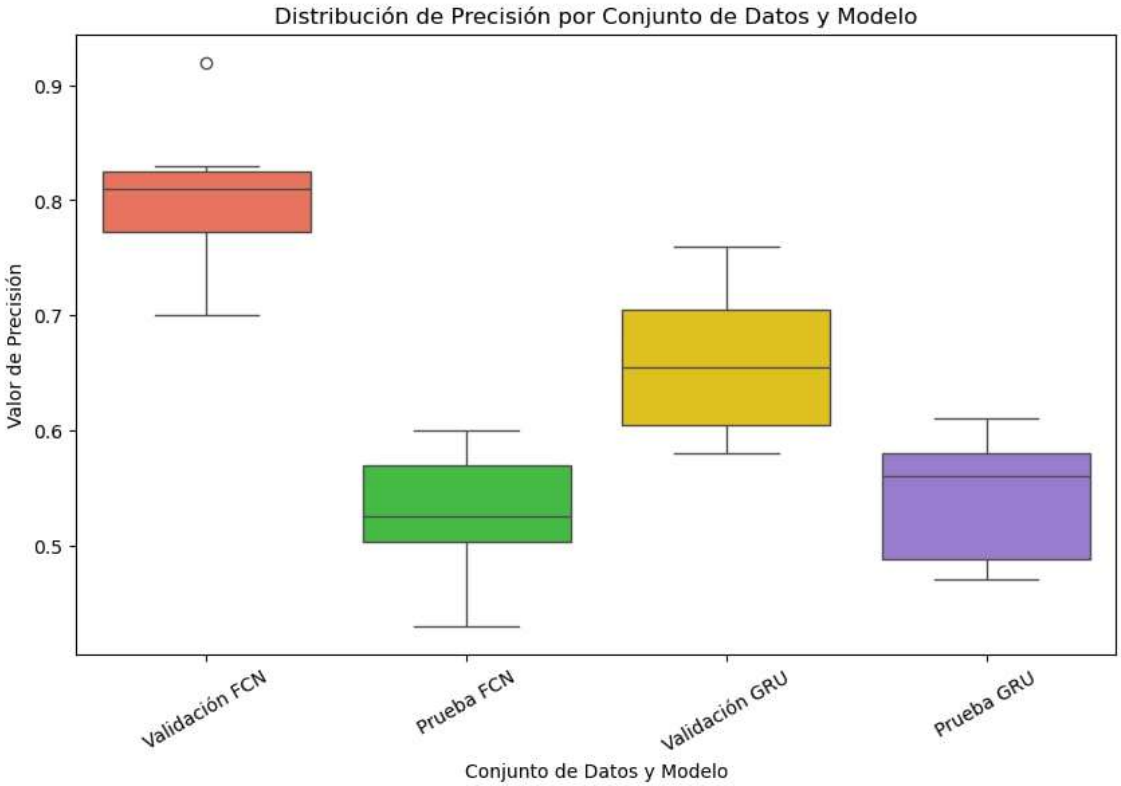

Figura 4.20: Distribución de precisión por tipo conjunto de datos y modelo de *Deep Learning*, basado en las precisiones obtenidas en la tabla 3.7.

Lo anterior, junto con lo discutido en las secciones 4.2.1 y 4.2.2, permite concluir que los modelos de *Deep Learning* están aprendiendo patrones específicos presentes en los datos de entrenamiento y validación. Sin embargo, debido a la limitada cantidad de datos, su alta variabilidad y la falta de generalización, estos modelos no logran mantener un rendimiento adecuado al evaluarse con el conjunto de prueba, compuesto por datos previamente no vistos.

En este contexto, y dado el análisis detallado, se concluye que para este caso particular, donde la base de datos es relativamente pequeña y presenta alta variabilidad, resulta más efectivo utilizar modelos menos complejos basados en técnicas de *Machine Learning*. Como se demostró en los análisis previos, estos modelos tienen una mayor efectividad al clasificar datos de prueba, a pesar de requerir un preprocesamiento más elaborado y una extracción de características previa.

En general, los modelos de *Deep Learning* suelen superar a los modelos de *Machine Learning* cuando se cuenta con un volumen de datos significativo que permita capturar patrones generales y reducir la variabilidad. Sin embargo, en escenarios con datos limitados, como este caso, los modelos menos complejos muestran un desempeño más robusto, lo que resalta la importancia de adaptar la complejidad del modelo al tamaño y calidad de los datos disponibles.

Capítulo 5

Conclusiones y Trabajo Futuro

5.1. Conclusiones

En este trabajo se desarrolló exitosamente una herramienta basada en técnicas de *Machine Learning* para el diagnóstico del glaucoma, utilizando datos obtenidos a partir de un *setup* experimental de pupilometría cromática. Los resultados logrados muestran una alta tasa de precisión relativa, y los modelos implementados entregan orientaciones claras y resultados prometedores, posicionando esta metodología como una solución viable y con potencial en el ámbito del diagnóstico automatizado.

Para alcanzar este objetivo, se optimizó de manera rigurosa la preparación de los datos experimentales, generando una base de datos ordenada y consistente. Se implementaron diversas técnicas de procesamiento, tales como saturación de señales, *padding* y filtrado, garantizando una representación adecuada de las señales. Adicionalmente, se extrajeron características relevantes de las señales, y se experimentó con técnicas de aumento de datos y selección de características, evaluando su impacto en el rendimiento de los modelos.

En el ámbito del modelado, se implementaron tanto modelos basados en *Machine Learning* como en *Deep Learning*, explorando diferentes configuraciones y optimizando su precisión y robustez. Los resultados evidenciaron que los modelos de *Machine Learning* mostraron un desempeño más consistente y robusto, favorecidos por el preprocesamiento exhaustivo realizado y por la simplicidad intrínseca de estos modelos. Las características extraídas lograron describir de manera precisa el comportamiento de las señales, lo que permitió a los modelos capturar patrones significativos.

El análisis de precisión permitió identificar al mejor modelo implementado, el cual corresponde a un modelo de *Machine Learning* que utiliza únicamente estímulos de tipo rampa, aplicando filtrado y ventanas de 150 muestras de largo. Sin embargo, se observó que, en promedio, los estímulos de tipo *flash* generan mejores resultados generales. Además, se confirmó que los algoritmos asignan mayor importancia a las respuestas pupilares iniciales provocadas por los estímulos lumínicos. Este hallazgo refuerza la capacidad de los modelos para evaluar con precisión las características más relevantes de estos momentos iniciales, que son clave desde una perspectiva médica.

Por otro lado, los modelos de *Deep Learning* no lograron un rendimiento óptimo, debido

a la limitada cantidad de datos, su alta variabilidad y la baja generalización de estos. Este resultado subraya que no siempre es preferible utilizar modelos más complejos, y que modelos más sencillos pueden ofrecer mejores resultados si están acompañados de un preprocesamiento sólido de los datos.

Finalmente, este trabajo representa un primer paso exploratorio hacia el desarrollo de una herramienta confiable para el diagnóstico y tamizaje del glaucoma. Se concluye que es posible implementar herramientas diagnósticas efectivas basadas en pupilometría cromática, abriendo un abanico de posibilidades para su continuidad, perfeccionamiento y eventual aplicación clínica.

5.2. Trabajo Futuro

Los resultados obtenidos en este trabajo representan un avance significativo hacia el desarrollo de una herramienta confiable para el diagnóstico del glaucoma a partir de datos de pupilometría cromática. Sin embargo, el análisis realizado también ha puesto de manifiesto desafíos importantes, como la necesidad de mejorar la generalización de los modelos y de trabajar con conjuntos de datos más amplios y diversos. Estos hallazgos no solo consolidan las bases para abordar el diagnóstico automatizado del glaucoma, sino que también abren nuevas oportunidades para optimizar las metodologías empleadas, explorar enfoques innovadores y superar las limitaciones identificadas, aspectos que serán fundamentales en los próximos pasos de esta investigación. Entre las líneas de trabajo futuro que surgen de este análisis, destacan:

1. **Extracción de Información:**

    - **Automatización del procesamiento:** Diseñar un programa que permita coordinar los estímulos lumínicos con la información extraída de *PupilLabs*. Esto facilitaría un procesamiento más preciso, rápido y automatizado de los datos, reduciendo la intervención manual y minimizando errores.
    - **Homogeneidad en frecuencias de muestreo:** Verificar que las frecuencias de muestreo de los datos extraídos de *PupilLabs* y las cámaras utilizadas sean siempre consistentes. Esto garantizaría un preprocesamiento uniforme para todos los datos, evitando pérdida de información y asegurando la comparabilidad de los resultados.
    - **Incremento en la recolección de datos:** Realizar múltiples tomas de muestras por paciente, repitiendo los experimentos más de una vez para aumentar la cantidad y la calidad de los datos disponibles. Esto permitiría una mejor representación estadística de cada caso y reduciría la variabilidad inherente a los experimentos.

2. **Preparación de Datos y Extracción de Características:**

    - **Métodos avanzados de remuestreo:** Explorar nuevas estrategias de remuestreo de señales que minimicen la pérdida de información o, idealmente, evitar la necesidad de remuestreo. Esto es especialmente relevante para preservar las características críticas de las señales originales.

- **Extracción de características especializadas:** Investigar métodos específicos para extraer información de señales no periódicas, independientemente de la longitud de las muestras. Esto permitiría una integración más eficiente de respuestas ante estímulos tipo rampa y *flash*, optimizando el análisis sin restricciones de longitud.
- **Combinación de estímulos:** Diseñar estrategias que permitan combinar de manera efectiva los datos provenientes de estímulos lumínicos de diferentes tipos, facilitando un análisis conjunto que aproveche las fortalezas de cada estímulo.

3. **Implementación de Modelos:**

   - **Exploración de modelos adicionales:** Probar una variedad más amplia de modelos predictivos para identificar configuraciones más robustas. Esto incluye modelos tradicionales, técnicas más avanzadas de *deep learning* y enfoques híbridos.
   - **Combinación de modelos:** Experimentar con más técnicas de *Ensemble Learning*, que combinan las predicciones de múltiples modelos para crear un sistema más robusto y preciso. Esto podría ser particularmente útil si ciertos modelos son más efectivos con características específicas y otros con diferentes aspectos de los datos.
   - **Clasificación jerárquica:** Si se dispone de suficientes datos, considerar la implementación de un sistema de clasificación jerárquico. Por ejemplo, desarrollar un modelo inicial que determine si un paciente tiene glaucoma y, en caso afirmativo, implementar un modelo secundario que clasifique la etapa de la enfermedad (temprana, moderada o avanzada).

4. **Métricas de Evaluación:**

   - **Ampliación de métricas:** Considerar métricas de evaluación adicionales al *accuracy* (precisión) es fundamental, dado el contexto médico de este diagnóstico. En este caso, el costo asociado a clasificaciones erróneas no se mide en términos económicos, sino en términos del impacto en la salud de las personas. Clasificar incorrectamente a un paciente con glaucoma como sano (falso negativo) es significativamente más crítico que diagnosticar a una persona sana como enferma (falso positivo). Por ello, se debe priorizar la maximización del *recall*, que mide la proporción de pacientes con glaucoma correctamente diagnosticados. Optimizar el *recall* aseguraría que la mayoría de los casos de glaucoma sean detectados, reduciendo el riesgo de no identificar a pacientes que requieren tratamiento urgente.

Estas líneas de trabajo futuro no solo consolidarán los avances logrados en este estudio, sino que también abrirán nuevas posibilidades para mejorar la efectividad y la aplicabilidad clínica de los modelos desarrollados.

Bibliografía

[1] Bourne, R., "The optic nerve head in glaucoma", Community Eye Health, vol. 19, no. 59, pp. 44–45, 2006.

[2] Guglielmi, P., Carradori, S., Campestre, C., y Poce, G., "Novel therapies for glaucoma: a patent review (2013–2019)", Expert Opinion on Therapeutic Patents, vol. 29, pp. 769–780, October 2019, doi:10.1080/13543776.2019.1653279. Epub 2019 Aug 11.

[3] Tham, Y.-C., Li, X., Wong, T., Quigley, H., Aung, T., y Cheng, C., "Global prevalence of glaucoma and projections of glaucoma burden through 2040: a systematic review and meta-analysis", Ophthalmology, vol. 121, pp. 2081–2090, November 2014, doi:10.1016/j.ophtha.2014.05.013. Epub 2014 Jun 26.

[4] Plaza-Rosales, I., "La pupilometría cromática: Un nuevo enfoque para el diagnóstico temprano del glaucoma crónico de ángulo abierto". Presentación en Jornadas SOCHIOF 2023, 2023.

[5] Broadway, D., "Visual field testing for glaucoma - a practical guide", Community Eye Health, vol. 25, no. 79-80, pp. 66–70, 2012.

[6] Arévalo-López, C., Gleitze, S., Madariaga, S., y Plaza-Rosales, I., "Pupillary response to chromatic light stimuli as a possible biomarker at the early stage of glaucoma: a review", International Ophthalmology, vol. 43, pp. 343–356, January 2023, doi:10.1007/s10792-022-02381-8. Epub 2022 Jul 4.

[7] Negrete, F. M. y Rebolleda, G., "Automated evaluation of the pupil", Archivos de la Sociedad Española de Oftalmología, vol. 88, pp. 125–126, April 2013, doi:10.1016/j.oftal.2013.02.014. English, Spanish.

[8] Berson, D., Dunn, F., y Takao, M., "Phototransduction by retinal ganglion cells that set the circadian clock", Science, vol. 295, pp. 1070–1073, February 8 2002, doi:10.1126/science.1067262.

[9] Amazon Web Services, "¿cuál es la diferencia entre la ia y el machine learning?". https://aws.amazon.com/es/compare/the-difference-between-artificial-intelligence-and-machine-learning/#:~:text=La%20IA%20es%20mejor%20para,finalidad%20de%20resolver%20problemas%20espec%C3%ADficos.&text=La%20IA%20puede%20usar%20una,%2C%20la%20visi%C3%B3n%20artificial%2C%20etc, s.f. Accedido: agosto 2024.

[10] Bishop, C., Pattern Recognition and Machine Learning (Information Science and Statistics). Springer, octubre 2007.

[11] Murphy, K. P., Machine Learning: A Probabilistic Perspective. Cambridge, MA: MIT Press, 2012.

[12] Hastie, T., Tibshirani, R., y Friedman, J., The Elements of Statistical Learning. New

York, NY: Springer, 2009.

[13] Domingos, P., "A few useful things to know about machine learning", Commun. ACM, vol. 55, p. 78–87, oct 2012, doi:10.1145/2347736.2347755.

[14] Jolliffe, I. T. y Cadima, J., "Principal component analysis: a review and recent developments", Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, vol. 374, no. 2065, p. 20150202, 2016, doi:10.1098/rsta.2015.0202.

[15] Reddy, K. y Shah, M., "Macro-class selection for hierarchical k-nn classification of inertial sensor data", PECCS 2012 - Proceedings of the 2nd International Conference on Pervasive Embedded Computing and Communication Systems, 05 2012.

[16] Singh, A. K. y Krishnan, S., "Ecg signal feature extraction trends in methods and applications", BioMed Engineering OnLine, vol. 22, no. 22, 2023, doi:10.1186/s12938-023-01075-1. Published March 8, 2023, Received October 26, 2022, Accepted January 27, 2023.

[17] IBM, "What is a decision tree?", 2024, https://www.ibm.com/topics/decision-trees. Accessed: Sep. 24, 2024.

[18] Devore, J. L., Probability and Statistics for Engineering and the Sciences. Boston, MA: Cengage Learning, 8th ed., 2011.

[19] Chen, T. y Guestrin, C., "Xgboost: A scalable tree boosting system", en Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, (New York, NY, USA), p. 785–794, Association for Computing Machinery, 2016, doi:10.1145/2939672.2939785.

[20] Friedman, J. H., "Greedy function approximation: A gradient boosting machine", The Annals of Statistics, vol. 29, no. 5, pp. 1189–1232, 2001, http://www.jstor.org/stable/2699986 (visitado el 2024-09-24).

[21] MDS7202 Repository, "15_Boosting [Notebook]". *GitHub*. Available: https://github.com/MDS7202/MDS7202/blob/main/clases/2024-01/15_Boosting.ipynb, 2024. Accessed: Nov. 8, 2024.

[22] Haykin, S., Neural Networks: A Comprehensive Foundation, vol. 3259 de Spie Proceedings Series. Upper Saddle River, NJ: Prentice Hall, 2 ed., 1999.

[23] Rumelhart, D. E., Hinton, G. E., y Williams, R. J., "Learning representations by back-propagating errors", Nature, vol. 323, pp. 533–536, October 1986, doi:10.1038/323533a0. Received 01 May 1986, Accepted 31 July 1986, Issue Date 09 October 1986.

[24] Goodfellow, I., Deep learning. MIT press, 2016.

[25] Nielsen, M. A., Neural Networks and Deep Learning. Determination Press, 2015, http://neuralnetworksanddeeplearning.com/.

[26] LeCun, Y., Bengio, Y., y Hinton, G., "Deep learning", Nature, vol. 521, pp. 436–444, May 2015, doi:10.1038/nature14539. Received 25 February 2015, Accepted 01 May 2015, Published 27 May 2015, Issue Date 28 May 2015.

[27] Goldberg, Y., Neural Network Methods for Natural Language Processing. Morgan & Claypool Publishers, 2017.

[28] Zhang, G., "Time series forecasting using a hybrid arima and neural network model", Neurocomputing, vol. 50, pp. 159–175, 2003, doi:https://doi.org/10.1016/S0925-231

2(01)00702-0.

[29] Krizhevsky, A., Sutskever, I., y Hinton, G. E., "Imagenet classification with deep convolutional neural networks", en Advances in Neural Information Processing Systems (Pereira, F., Burges, C., Bottou, L., y Weinberger, K., eds.), vol. 25, Curran Associates, Inc., 2012, https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[30] Young, T., Hazarika, D., Poria, S., y Cambria, E., "Recent trends in deep learning based natural language processing [review article]", IEEE Computational Intelligence Magazine, vol. 13, no. 3, pp. 55–75, 2018, doi:10.1109/MCI.2018.2840738.

[31] Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., y Kingsbury, B., "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups", IEEE Signal Processing Magazine, vol. 29, no. 6, pp. 82–97, 2012, doi:10.1109/MSP.2012.2205597.

[32] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., y Hassabis, D., "Mastering the game of go with deep neural networks and tree search", Nature, vol. 529, pp. 484–489, January 2016, doi:10.1038/nature16961. Received 11 November 2015, Accepted 05 January 2016, Published 27 January 2016, Issue Date 28 January 2016.

[33] Hochreiter, S. y Schmidhuber, J., "Long short-term memory", Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997, doi:10.1162/neco.1997.9.8.1735.

[34] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., y Bengio, Y., "Generative adversarial nets", en Advances in Neural Information Processing Systems (NIPS), pp. 2672–2680, 2014, https://arxiv.org/abs/1406.2661.

[35] Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., van der Laak, J. A. W. M., van Ginneken, B., y Sánchez, C. I., "A survey on deep learning in medical image analysis", Medical Image Analysis, vol. 42, pp. 60–88, December 2017, doi:10.1016/j.media.2017.07.005. Epub 2017 Jul 26, PMID: 28778026.

[36] Bojarski, M., Testa, D. D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., y Zieba, K., "End to end learning for self-driving cars", 2016, https://arxiv.org/abs/1604.07316.

[37] Heaton, J. B., Polson, N. G., y Witte, J. H., "Deep learning in finance", 2018, https://arxiv.org/abs/1602.06561.

[38] Elgammal, A., Liu, B., Elhoseiny, M., y Mazzone, M., "Can: Creative adversarial networks, generating "art" by learning about styles and deviating from style norms", 2017, https://arxiv.org/abs/1706.07068.

[39] CauchyTuring, "Ucr time series classification deep learning baseline". https://github.com/cauchyturing/UCR_Time_Series_Classification_Deep_Learning_Baseline/blob/master/README.md, 2024. Accessed: 28-Nov-2024.

[40] Zhou, W., Hao, K., Jiang, C., Chen, L., Tang, X.-S., y Cai, X., "A new cross clustering algorithm for improving performance of supervised learning", IEEE Access, vol. 7, pp. 56713–56723, 01 2019, doi:10.1109/ACCESS.2019.2909926.

[41] Sutskever, I., Vinyals, O., y Le, Q. V., "Sequence to sequence learning with neural

networks", 2014, https://arxiv.org/abs/1409.3215.

[42] Graves, A., Supervised Sequence Labelling with Recurrent Neural Networks. Berlin, Heidelberg: Springer, 2012.

[43] Werbos, P., "Backpropagation through time: what it does and how to do it", Proceedings of the IEEE, vol. 78, no. 10, pp. 1550–1560, 1990, doi:10.1109/5.58337.

[44] Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., y Bengio, Y., "Learning phrase representations using rnn encoder-decoder for statistical machine translation", 2014, https://arxiv.org/abs/1406.1078.

[45] Graves, A., rahman Mohamed, A., y Hinton, G., "Speech recognition with deep recurrent neural networks", 2013, https://arxiv.org/abs/1303.5778.

[46] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., *et al.*, "Language models are unsupervised multitask learners", OpenAI blog, vol. 1, no. 8, p. 9, 2019.

[47] Lipton, Z. C., Kale, D. C., Elkan, C., y Wetzel, R., "Learning to diagnose with lstm recurrent neural networks", 2017, https://arxiv.org/abs/1511.03677.

[48] Fawcett, T., "Introduction to roc analysis", Pattern Recognition Letters, vol. 27, pp. 861–874, 06 2006, doi:10.1016/j.patrec.2005.10.010.

[49] Yurek, I., "Roc curve and auc: Evaluating model performance". Medium, December 17 2019, https://medium.com/@ilyurek/roc-curve-and-auc-evaluating-model-performance-c2178008b02. [Online]. Available: https://medium.com/@ilyurek/roc-curve-and-auc-evaluating-model-performance-c2178008b02. [Accessed: Jul. 7, 2024].

[50] Saito, T. y Rehmsmeier, M., "The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets", PLoS ONE, vol. 10, no. 3, p. e0118432, 2015, doi:10.1371/journal.pone.0118432.

[51] Buckland, M. y Gey, F., "The relationship between recall and precision", Journal of the American Society for Information Science, vol. 45, no. 1, pp. 12–19, 1994, doi:https://doi.org/10.1002/(SICI)1097-4571(199401)45:1<12::AID-ASI2>3.0.CO;2-L.

[52] Kurbiel, T., "Gaining an intuitive understanding of precision and recall". Towards Data Science, February 19 2019, https://towardsdatascience.com/gaining-an-intuitive-understanding-of-precision-and-recall-3b9df37804a7. [Online]. Available: https://towardsdatascience.com/gaining-an-intuitive-understanding-of-precision-and-recall-3b9df37804a7. [Accessed: Jul. 7, 2024].

[53] James, G., Witten, D., Hastie, T., y Tibshirani, R., An Introduction to Statistical Learning: with Applications in R. Springer, 2013, https://faculty.marshall.usc.edu/gareth-james/ISL/.

[54] Thudumu, S., Branch, P., Jin, J., y Singh, J., "A comprehensive survey of anomaly detection techniques for high dimensional big data", Journal of Big Data, vol. 7, p. 42, 2020, doi:10.1186/s40537-020-00320-x. Received 21 February 2020, Accepted 21 June 2020, Published 02 July 2020.

[55] Guyon, I. y Elisseeff, A., "An introduction of variable and feature selection", J. Machine Learning Research Special Issue on Variable and Feature Selection, vol. 3, pp. 1157 – 1182, 01 2003, doi:10.1162/153244303322753616.

[56] Kuhn, M. y Johnson, K., Applied Predictive Modeling. New York, NY: Springer, 2013.

[57] MDS7202 Repository, "19_Interpretabilidad_I [Notebook]". *GitHub*. Available: https://github.com/MDS7202/MDS7202/blob/main/clases/2024-01/19_Interpretabilidad_I.ipynb, 2024. Accessed: Nov. 8, 2024.

[58] Doshi-Velez, F. y Kim, B., "Towards a rigorous science of interpretable machine learning", 2017, https://arxiv.org/abs/1702.08608.

[59] Molnar, C., Interpretable Machine Learning: A Guide for Making Black Box Models Explainable. Independently published, 2nd ed., 2022, https://christophm.github.io/interpretable-ml-book/. Creative Commons Licensed, eBook (2nd Edition, 2024-05-26).

[60] Lundberg, S. M. y Lee, S.-I., "A unified approach to interpreting model predictions", en Advances in Neural Information Processing Systems (Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., y Garnett, R., eds.), vol. 30, Curran Associates, Inc., 2017, https://proceedings.neurips.cc/paper_files/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf.

[61] Ribeiro, M. T., Singh, S., y Guestrin, C., "Model-agnostic interpretability of machine learning", 2016, https://arxiv.org/abs/1606.05386.

[62] Adadi, A. y Berrada, M., "Peeking inside the black-box: A survey on explainable artificial intelligence (xai)", IEEE Access, vol. 6, pp. 52138–52160, 2018, doi:10.1109/ACCESS.2018.2870052.

[63] Kavakiotis, I., Tsave, O., Salifoglou, A., Maglaveras, N., Vlahavas, I., y Chouvarda, I., "Machine learning and data mining methods in diabetes research", Computational and Structural Biotechnology Journal, vol. 15, pp. 104–116, 2017, doi:10.1016/j.csbj.2016.12.005.

[64] Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Pedreschi, D., y Giannotti, F., "A survey of methods for explaining black box models", 2018, https://arxiv.org/abs/1802.01933.

[65] Kim, S. J., Cho, K. J., y Oh, S., "Development of machine learning models for diagnosis of glaucoma", PLoS ONE, vol. 12, p. e0177726, May 23 2017, doi:10.1371/journal.pone.0177726. PMID: 28542342, PMCID: PMC5441603.

[66] Shoukat, A., Akbar, S., Hassan, S. A., Iqbal, S., Mehmood, A., y Ilyas, Q. M., "Automatic diagnosis of glaucoma from retinal images using deep learning approach", Diagnostics (Basel), vol. 13, p. 1738, May 14 2023, doi:10.3390/diagnostics13101738.

[67] Thakoor, K. A., Li, X., Tsamis, E., Sajda, P., y Hood, D. C., "Enhancing the accuracy of glaucoma detection from oct probability maps using convolutional neural networks", en 2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), pp. 2036–2040, 2019, doi:10.1109/EMBC.2019.8856899.

[68] Hemelings, R., Elen, B., Barbosa-Breda, J., Lemmens, S., Meire, M., Pourjavan, S., Vandewalle, E., de Veire, S. V., Blaschko, M. B., Boever, P. D., y Stalmans, I., "Accurate prediction of glaucoma from colour fundus images with a convolutional neural network that relies on active and transfer learning", Acta Ophthalmologica, vol. 98, pp. e94–e100, February 2020, doi:10.1111/aos.14193. Epub 2019 Jul 25.

[69] Pinheiro, H. M., Camilo, E. N. R., Paranhos, A., Fonseca, A. U., Laureano, G. T., y da Costa, R. M., "Evaluating machine learning techniques for enhanced glaucoma screening through pupillary light reflex analysis", Array, vol. 23, p. 100359, 2024, doi:

https://doi.org/10.1016/j.array.2024.100359.

Anexos

Anexo A. Resultados *Dummy model*

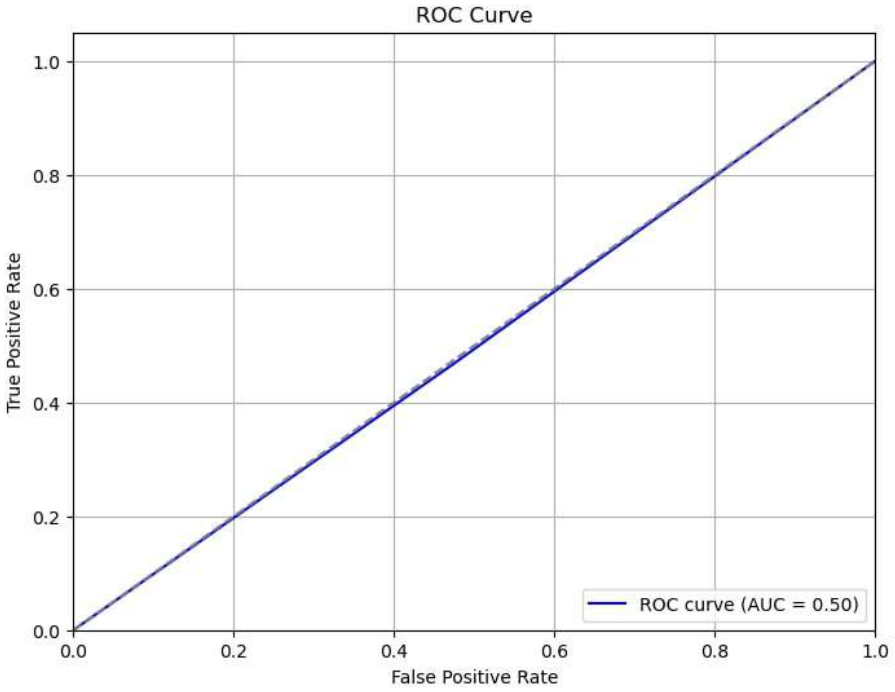

Figura A.1: Curva ROC del modelo *dummy*.

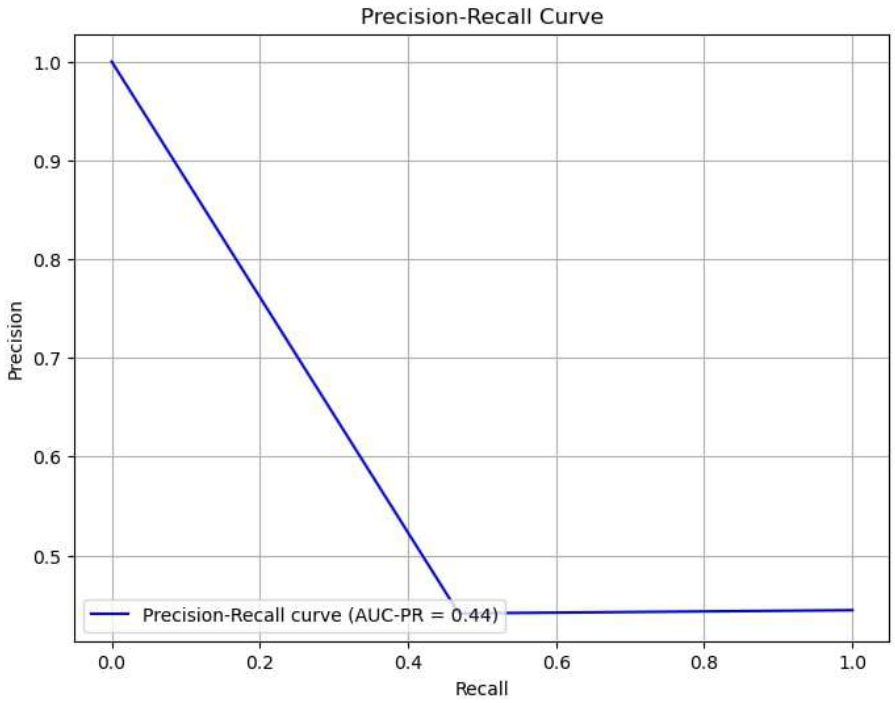

Figura A.2: Curva *precision-recall* del modelo *dummy*.

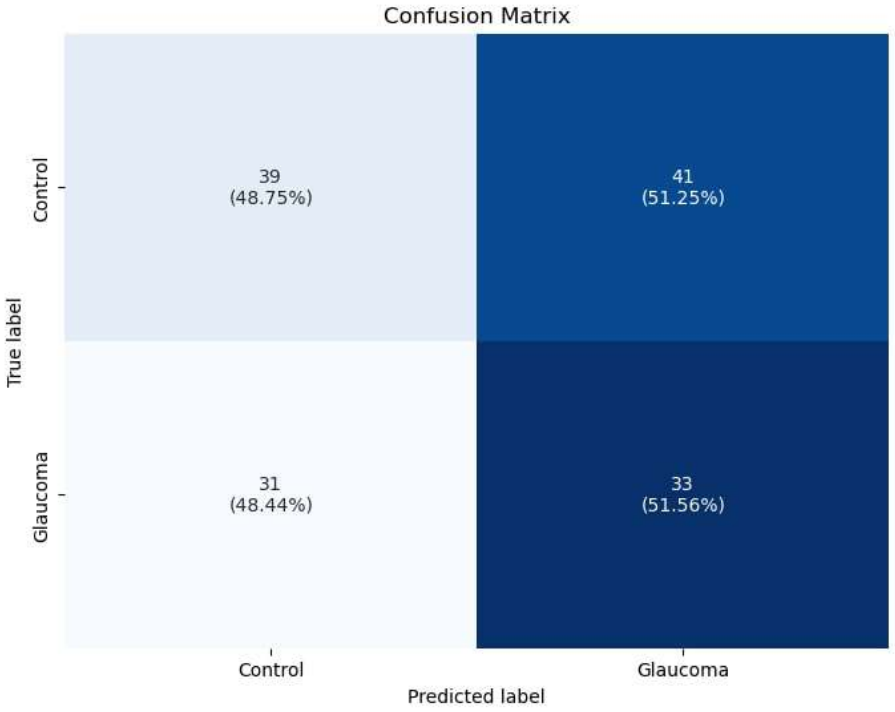

Figura A.3: Matriz de confusión del modelo *dummy*.

Anexo B. Resultados Modelos de Machine Learning

B.1. Resultados caso base

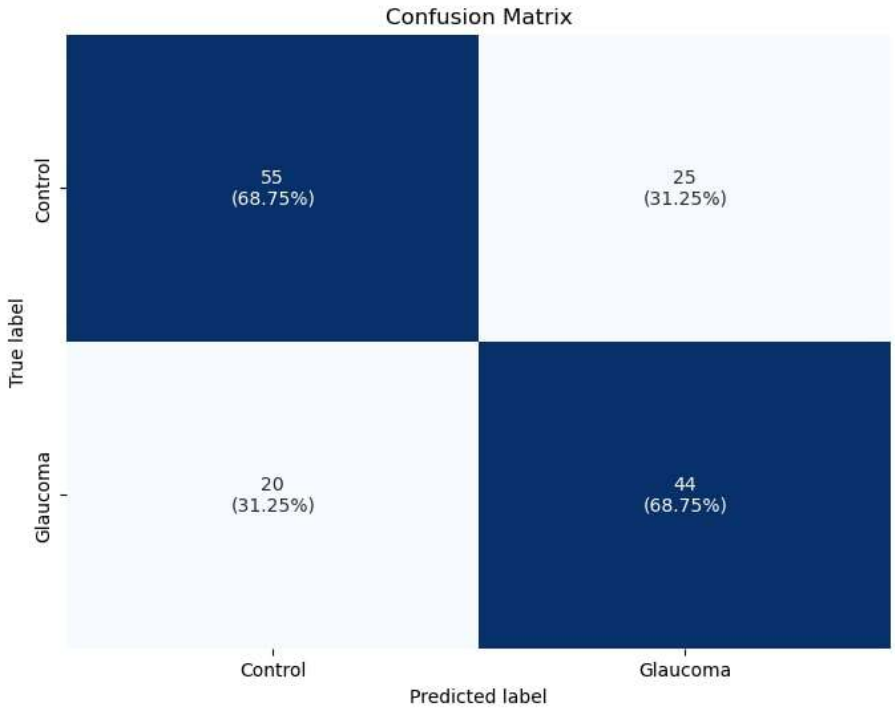

Figura B.1: Matriz de confusión caso base *D.A.*, con ventanas de 50 muestras de largo.

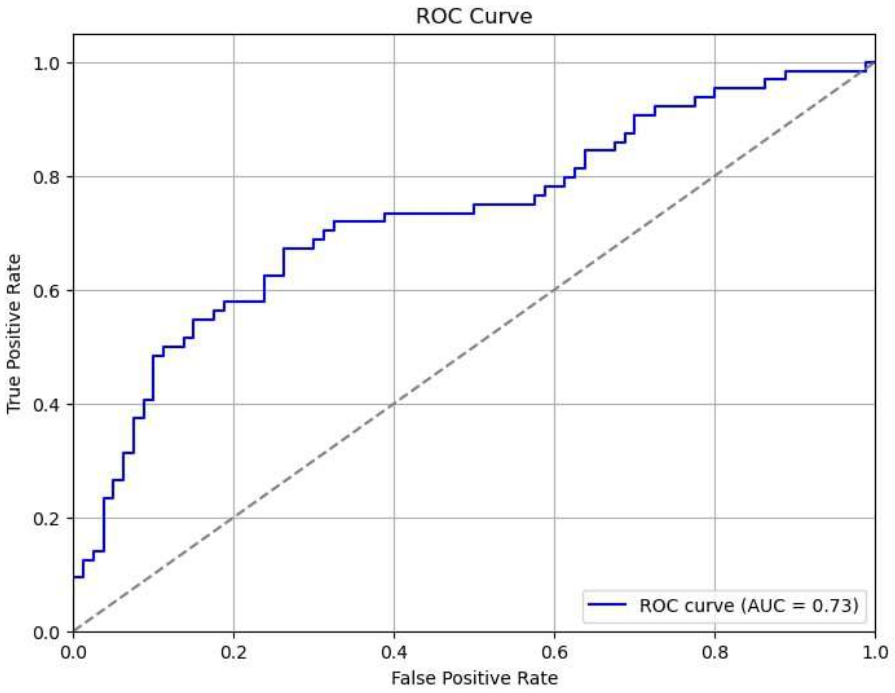

Figura B.2: Curva ROC caso base *D.A.*, con ventanas de 50 muestras de largo.

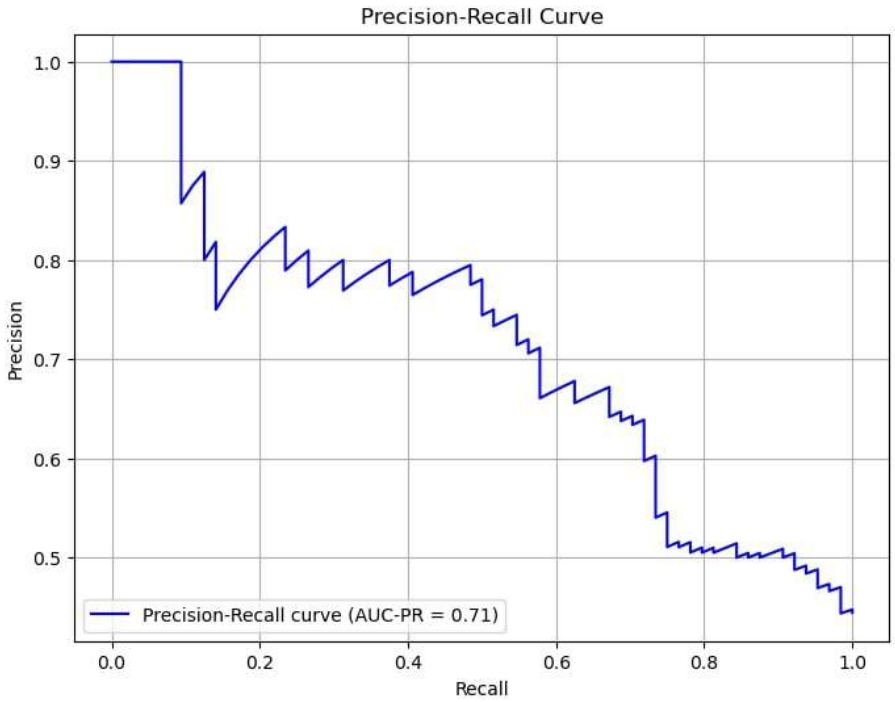

Figura B.3: Curva *precision-recall* caso base *D.A.*, con ventanas de 50 muestras de largo.

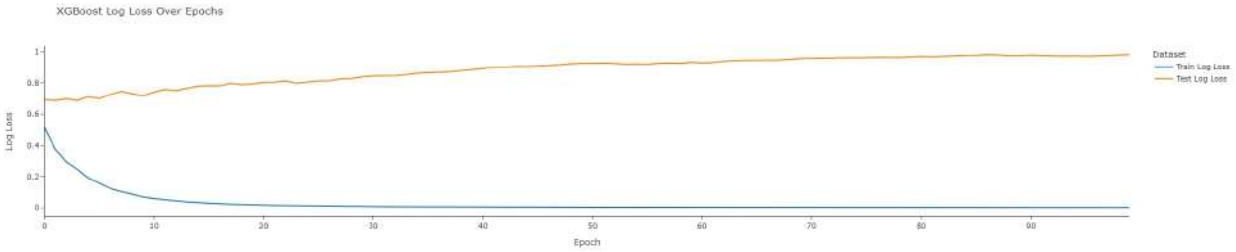

Figura B.4: Curvas de pérdida en el conjunto de entrenamiento y prueba caso base *D.A.*, con ventanas de 50 muestras de largo.

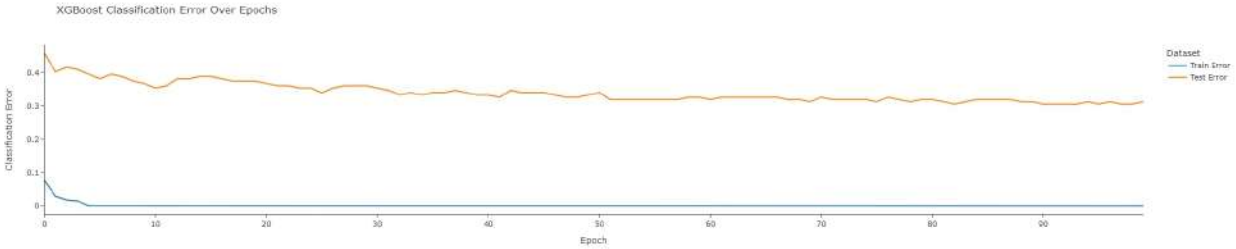

Figura B.5: Curvas de error en el conjunto de entrenamiento y prueba caso base *D.A.*, con ventanas de 50 muestras de largo.

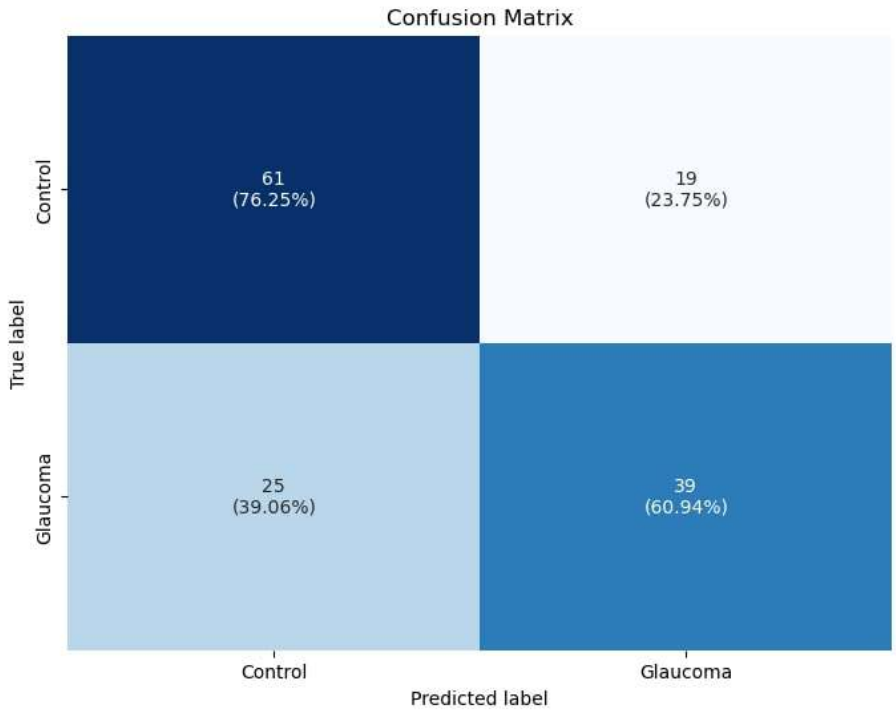

Figura B.6: Matriz de confusión caso base *Filtrado + D.A.*, con ventanas de 100 muestras de largo.

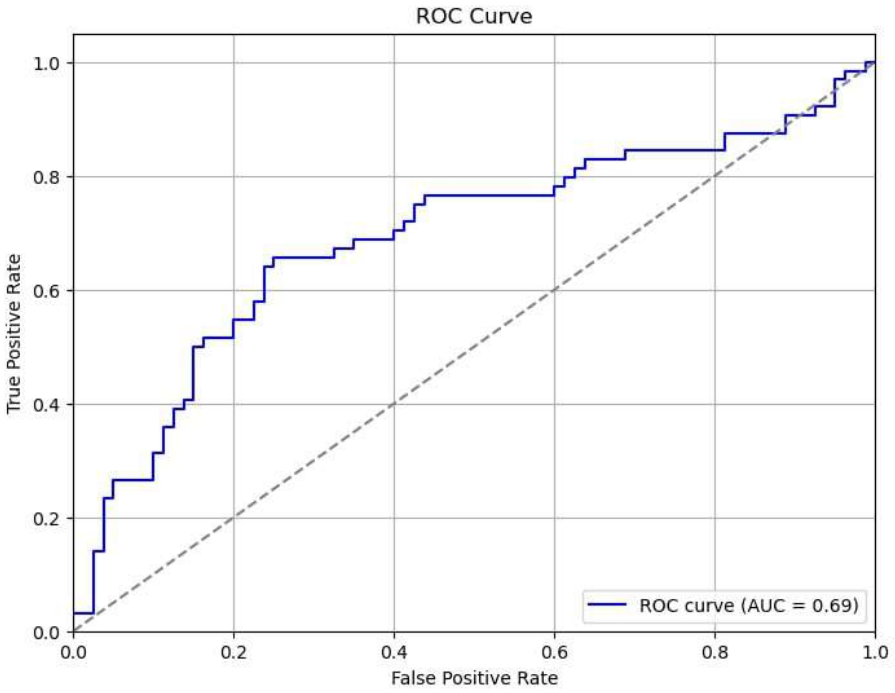

Figura B.7: Curva ROC caso base *Filtrado + D.A.*, con ventanas de 100 muestras de largo.

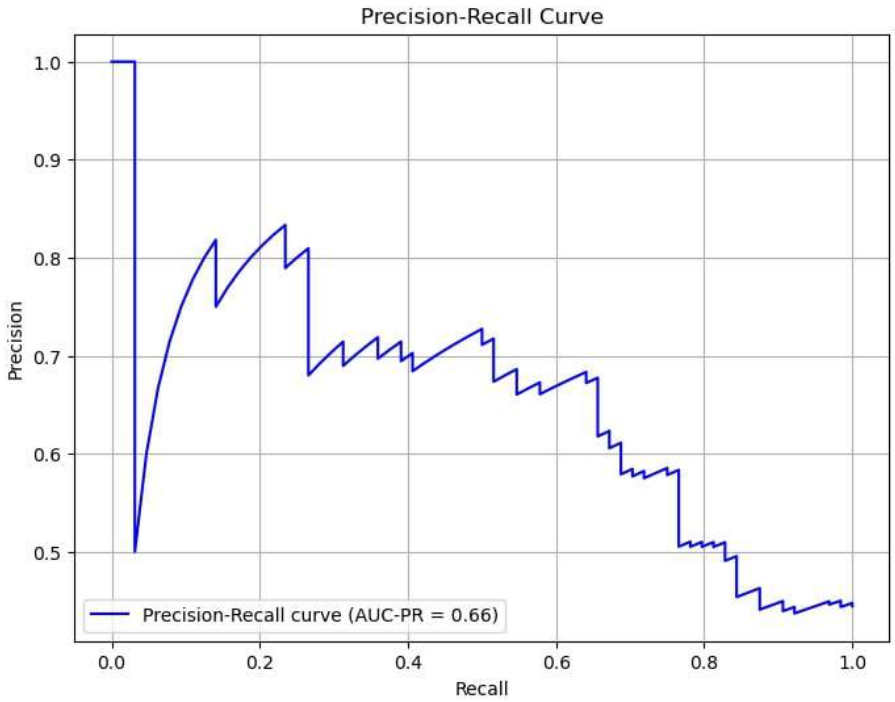

Figura B.8: Curva *precision-recall* caso base *Filtrado + D.A.*, con ventanas de 100 muestras de largo.

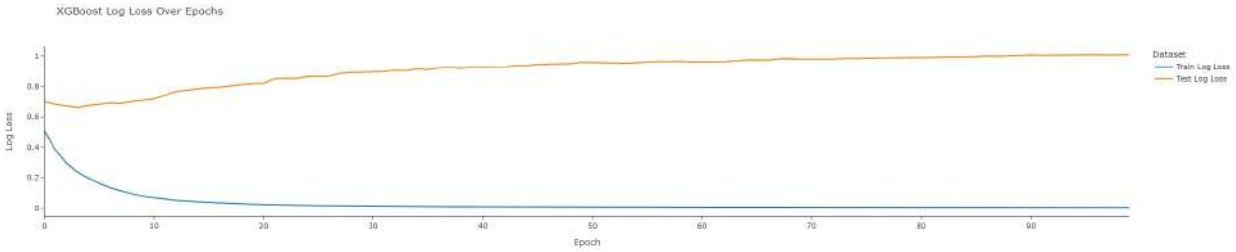

Figura B.9: Curvas de pérdida en el conjunto de entrenamiento y prueba caso base *Filtrado + D.A.*, con ventanas de 100 muestras de largo.

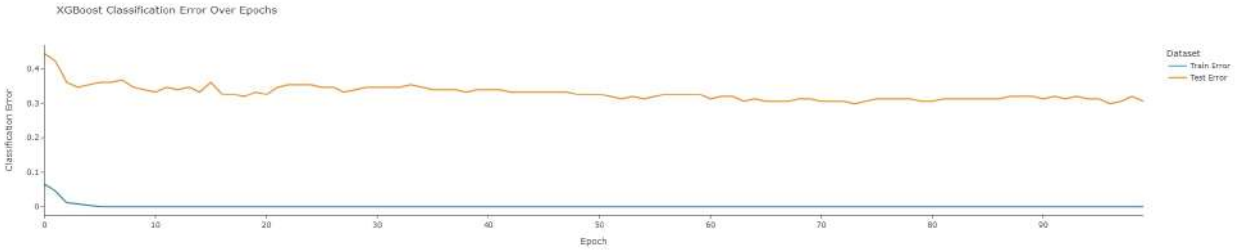

Figura B.10: Curvas de error en el conjunto de entrenamiento y prueba caso base *Filtrado + D.A.*, con ventanas de 100 muestras de largo.

B.2. Resultados primeros 20 segundos

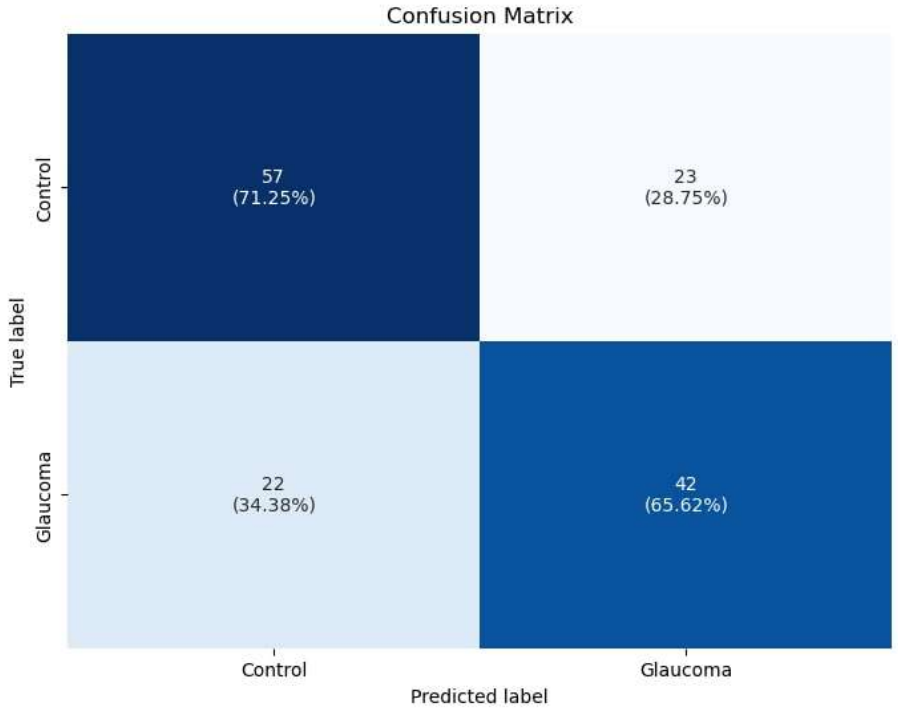

Figura B.11: Matriz de confusión caso selección de primeros 20 segundos *D.A.*, con ventanas de 25 muestras de largo.

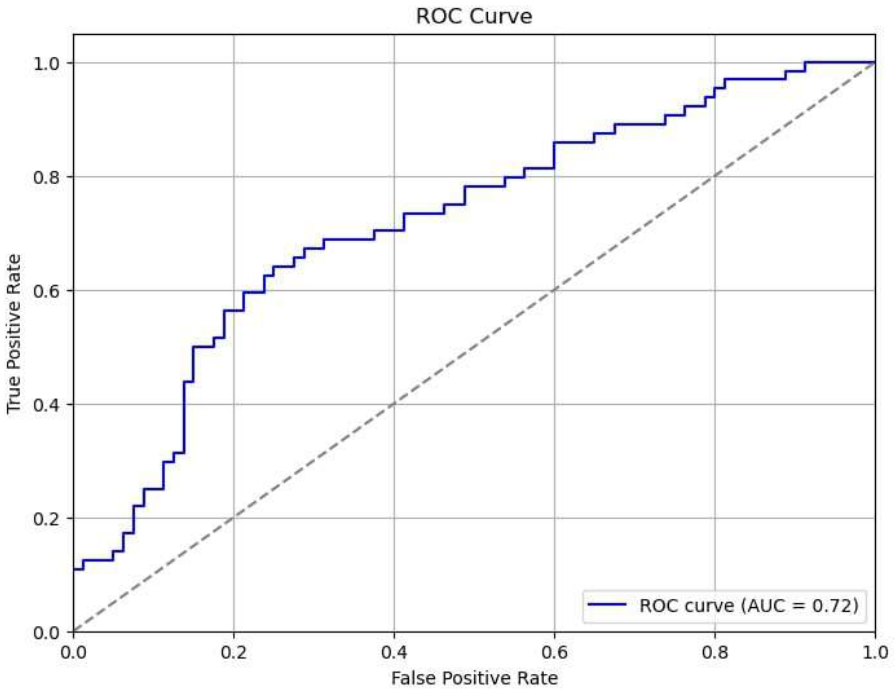

Figura B.12: Curva ROC caso selección de primeros 20 segundos *D.A.*, con ventanas de 25 muestras de largo.

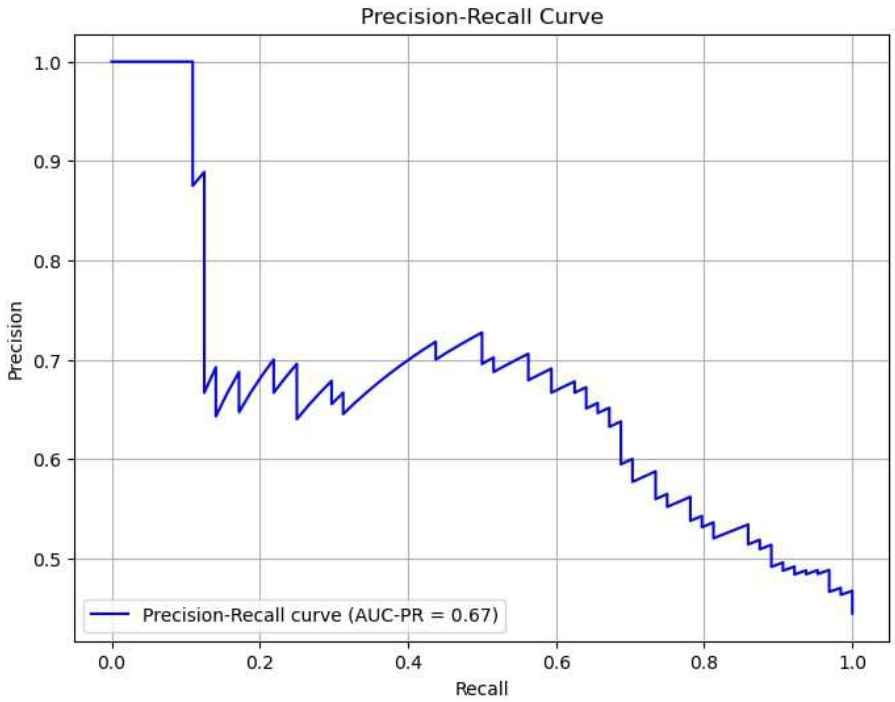

Figura B.13: Curva *precision-recall* caso selección de primeros 20 segundos *D.A.*, con ventanas de 25 muestras de largo.

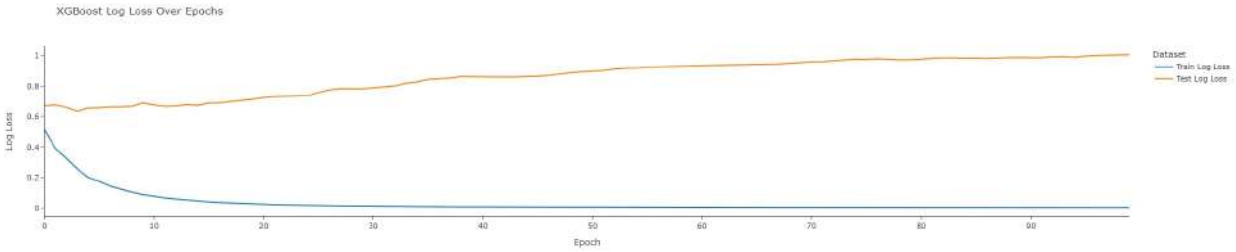

Figura B.14: Curvas de pérdida en el conjunto de entrenamiento y prueba caso selección de primeros 20 segundos *D.A.*, con ventanas de 25 muestras de largo.

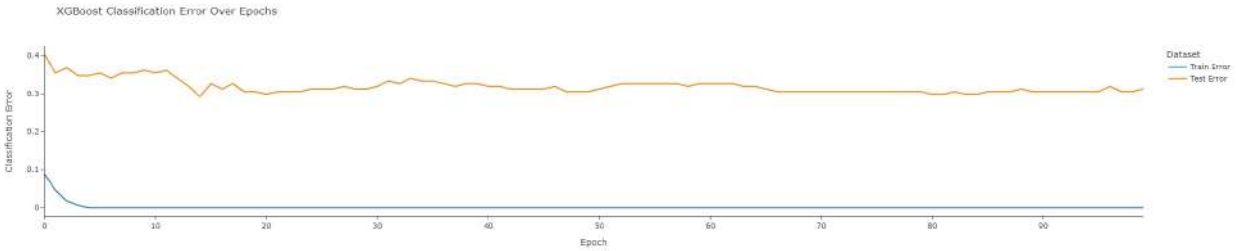

Figura B.15: Curvas de error en el conjunto de entrenamiento y prueba caso selección de primeros 20 segundos *D.A.*, con ventanas de 25 muestras de largo.

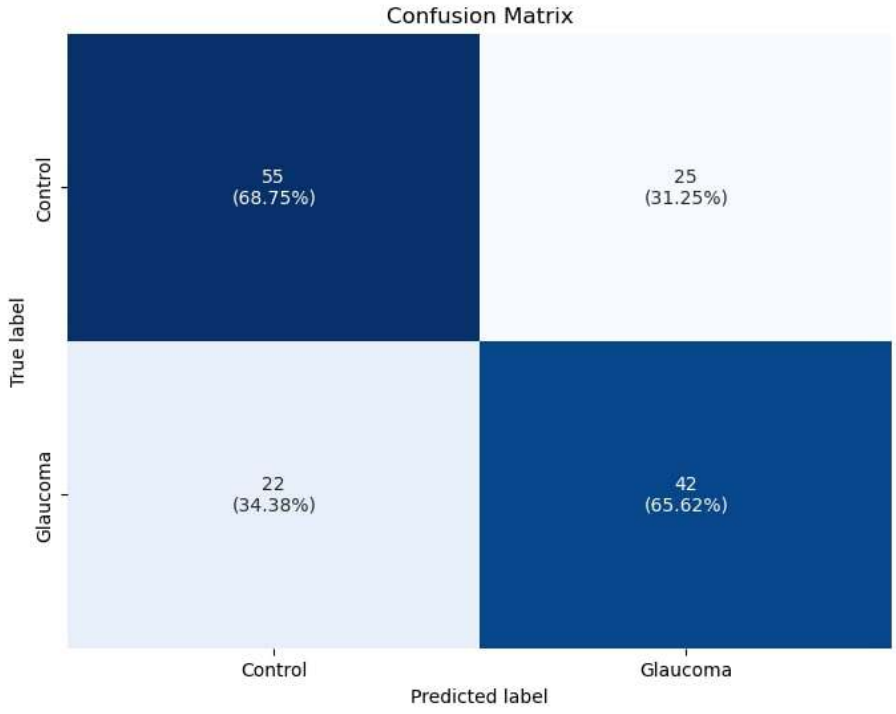

Figura B.16: Matriz de confusión caso selección de primeros 20 segundos *D.A.*, con ventanas de 50 y 100 muestras de largo.

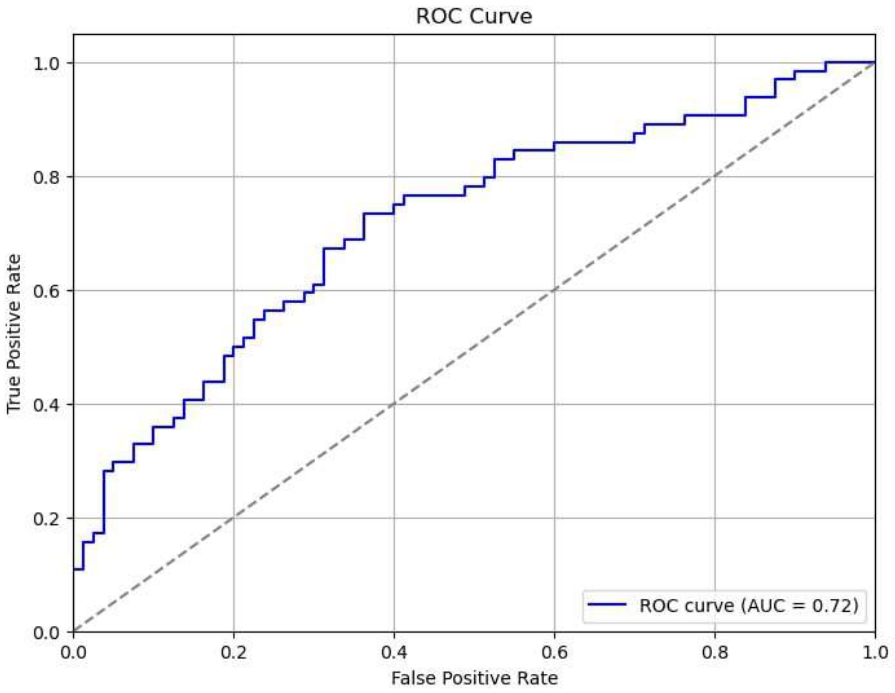

Figura B.17: Curva ROC caso selección de primeros 20 segundos *D.A.*, con ventanas de 50 y 100 muestras de largo.

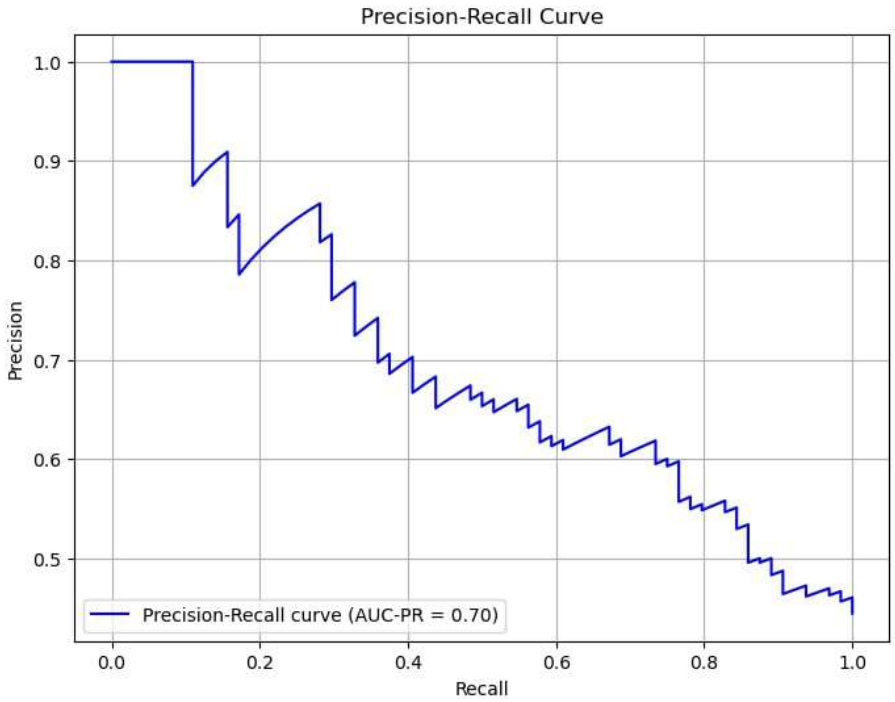

Figura B.18: Curva *precision-recall* caso selección de primeros 20 segundos *D.A.*, con ventanas de 50 y 100 muestras de largo.

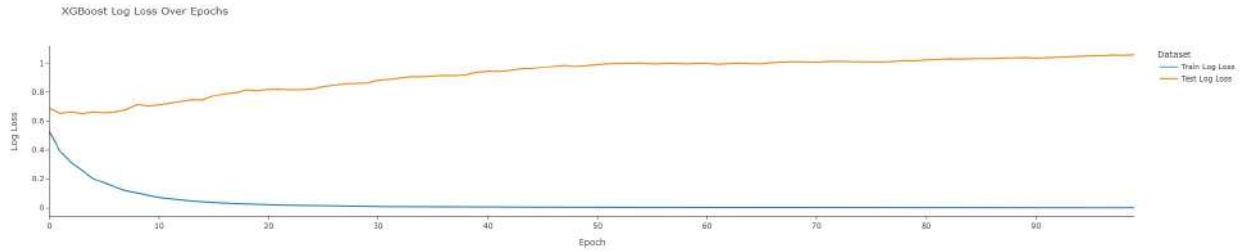

Figura B.19: Curvas de pérdida en el conjunto de entrenamiento y prueba caso selección de primeros 20 segundos *D.A.*, con ventanas de 50 y 100 muestras de largo.

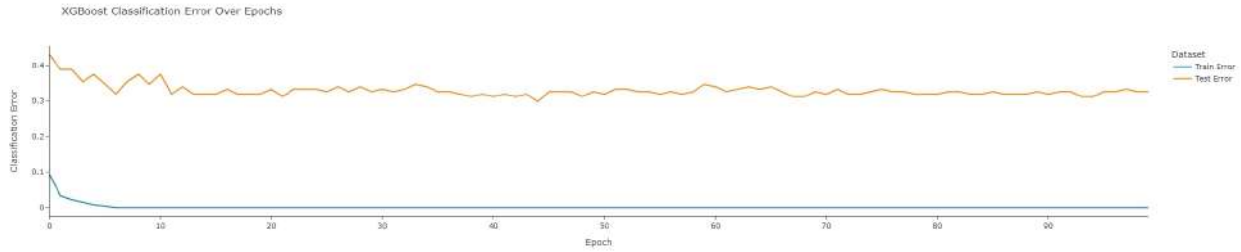

Figura B.20: Curvas de error en el conjunto de entrenamiento y prueba caso selección de primeros 20 segundos *D.A.*, con ventanas de 50 y 100 muestras de largo.

B.3. Resultados ventanas aleatorias

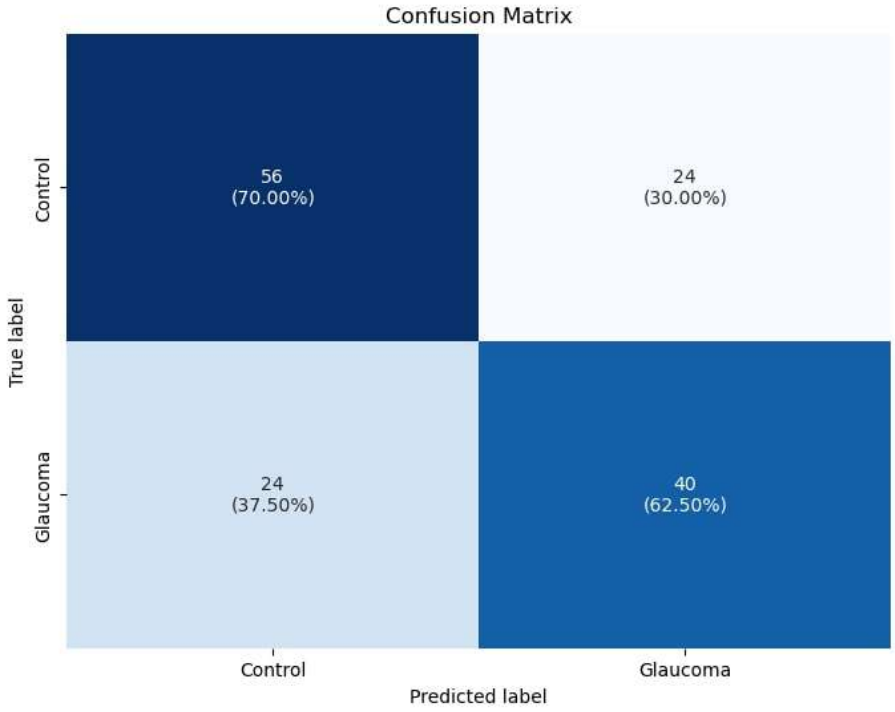

Figura B.21: Matriz de confusión caso selección de ventanas aleatorias *Normal*, con ventanas de 25 muestras de largo.

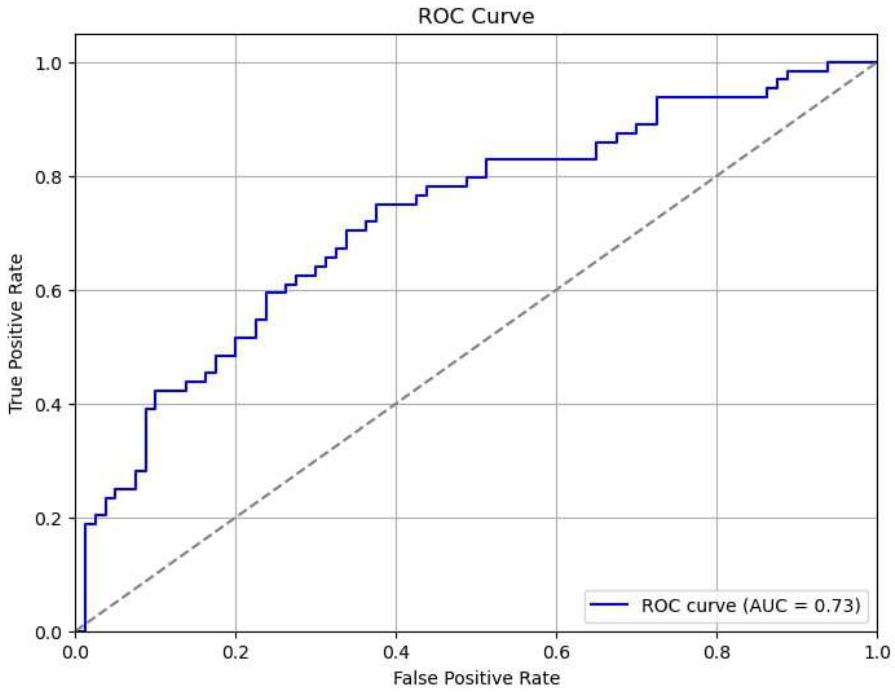

Figura B.22: Curva ROC caso selección de ventanas aleatorias *Normal*, con ventanas de 25 muestras de largo.

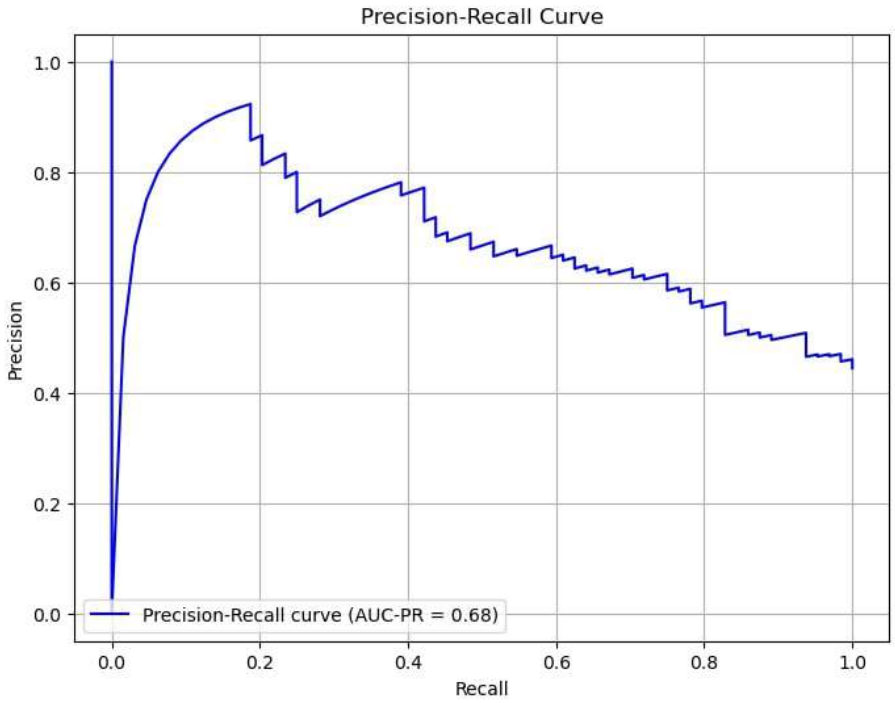

Figura B.23: Curva *precision-recall* caso selección de ventanas aleatorias *Normal*, con ventanas de 25 muestras de largo.

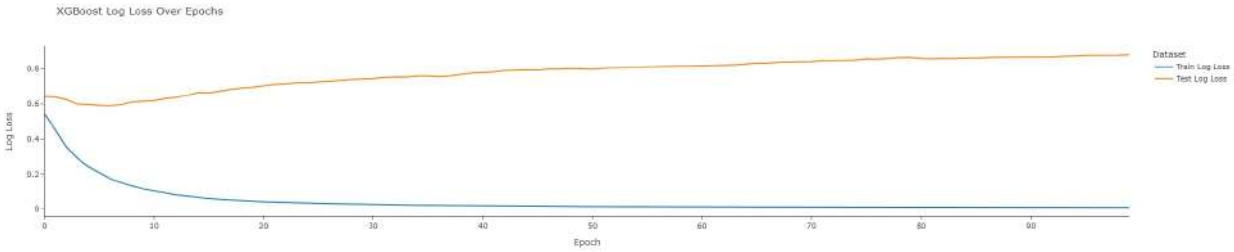

Figura B.24: Curvas de pérdida en el conjunto de entrenamiento y prueba caso selección de ventanas aleatorias *Normal*, con ventanas de 25 muestras de largo.

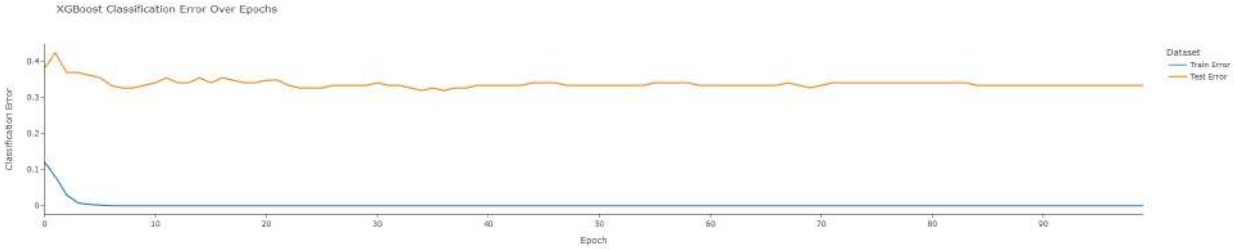

Figura B.25: Curvas de error en el conjunto de entrenamiento y prueba caso selección de ventanas aleatorias *Normal*, con ventanas de 25 muestras de largo.

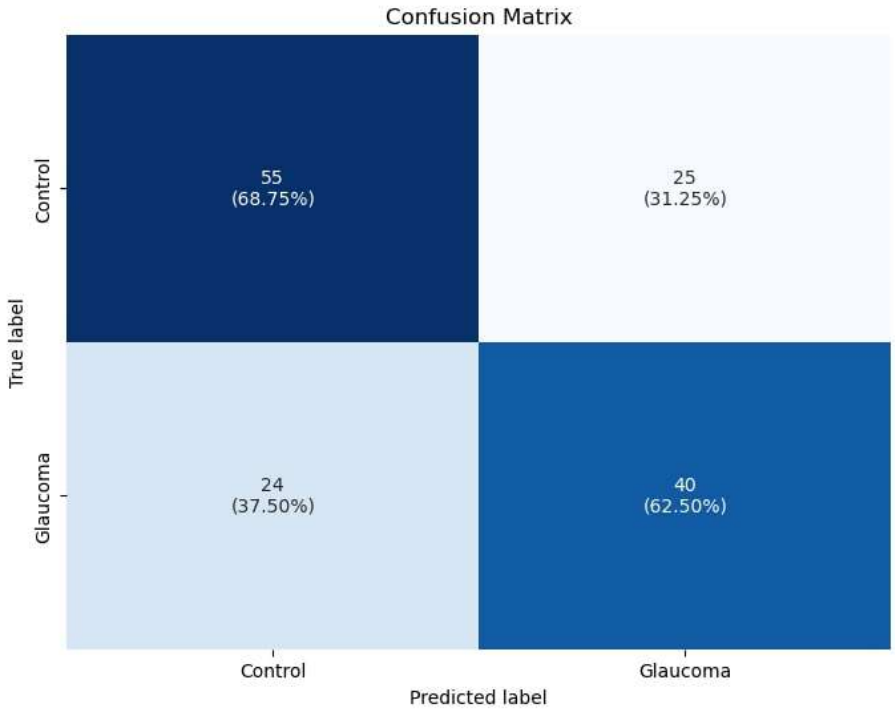

Figura B.26: Matriz de confusión caso selección de ventanas aleatorias *D.A.*, con ventanas de 25 muestras de largo.

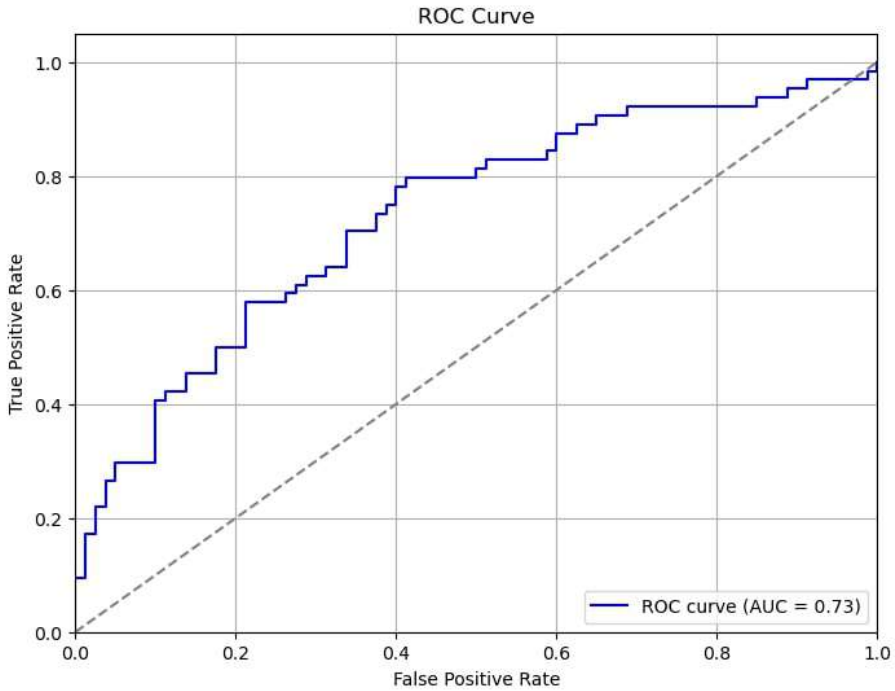

Figura B.27: Curva ROC caso selección de ventanas aleatorias *D.A.*, con ventanas de 25 muestras de largo.

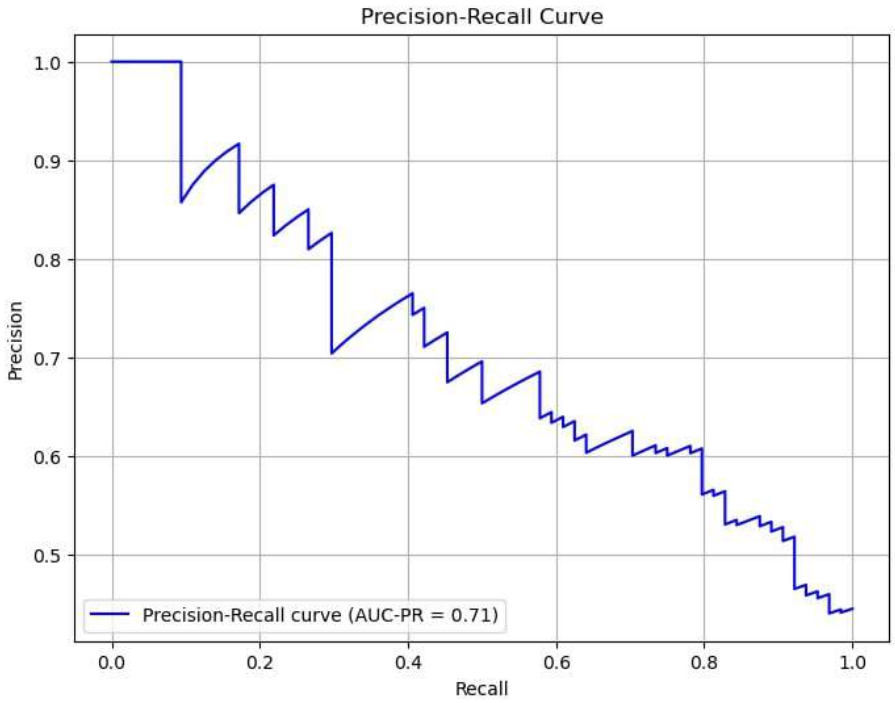

Figura B.28: Curva *precision-recall* caso selección de ventanas aleatorias *D.A.*, con ventanas de 25 muestras de largo.

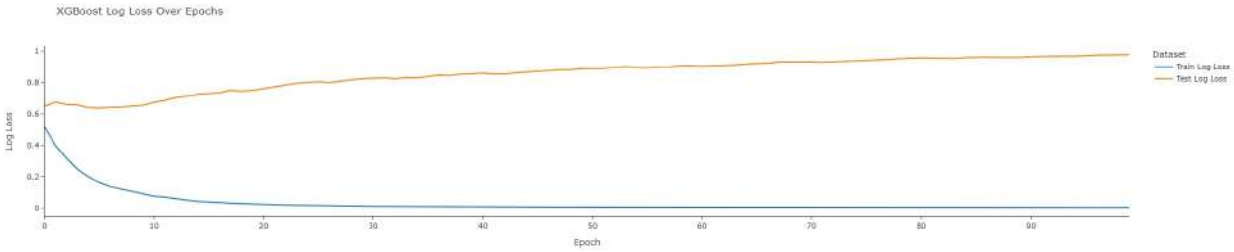

Figura B.29: Curvas de pérdida en el conjunto de entrenamiento y prueba caso selección de ventanas aleatorias *D.A.*, con ventanas de 25 muestras de largo.

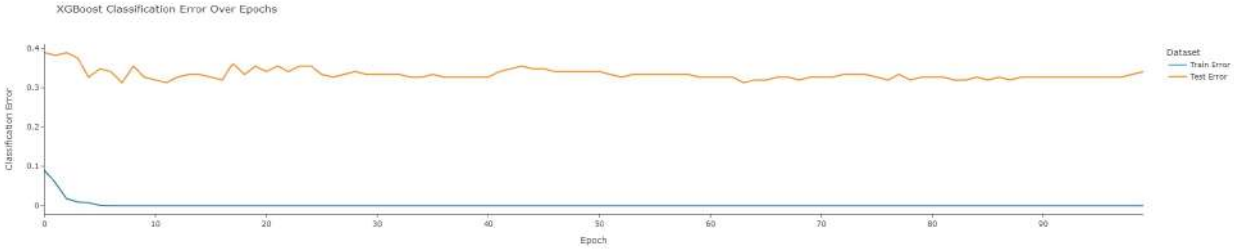

Figura B.30: Curvas de error en el conjunto de entrenamiento y prueba caso selección de ventanas aleatorias *D.A.*, con ventanas de 25 muestras de largo.

B.4. Resultados respuestas ante estímulos tipo *flash*

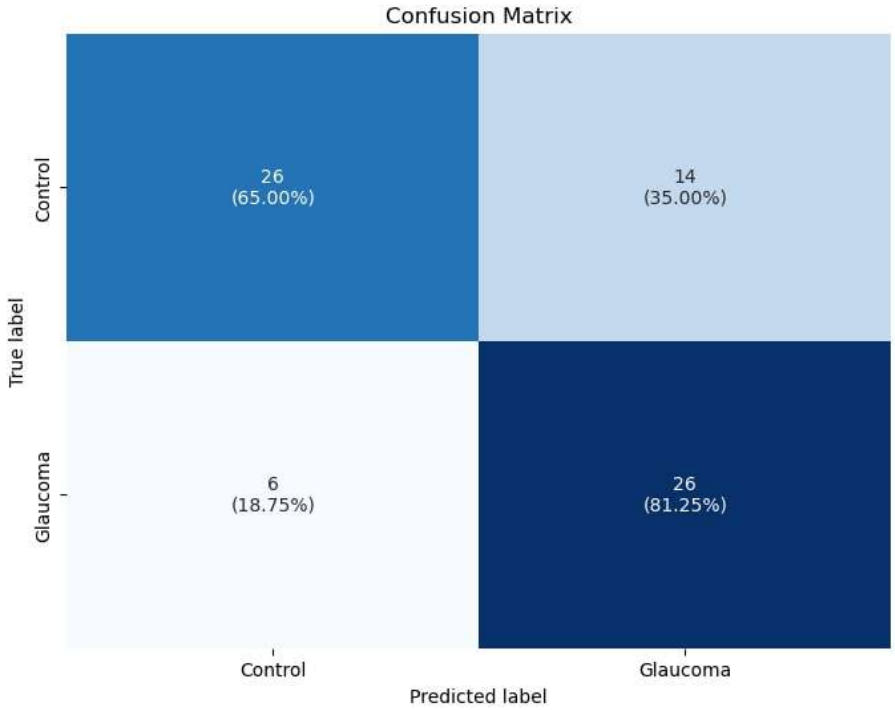

Figura B.31: Matriz de confusión caso selección de respuestas ante estímulos tipo *flash Normal*, con ventanas de 25 muestras de largo.

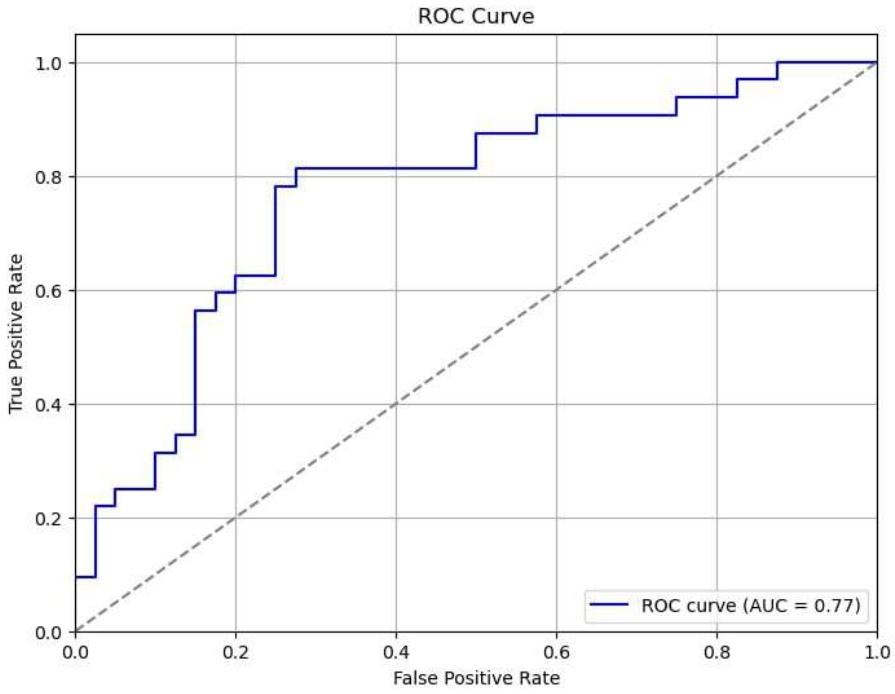

Figura B.32: Curva ROC caso selección de respuestas ante estímulos tipo *flash Normal*, con ventanas de 25 muestras de largo.

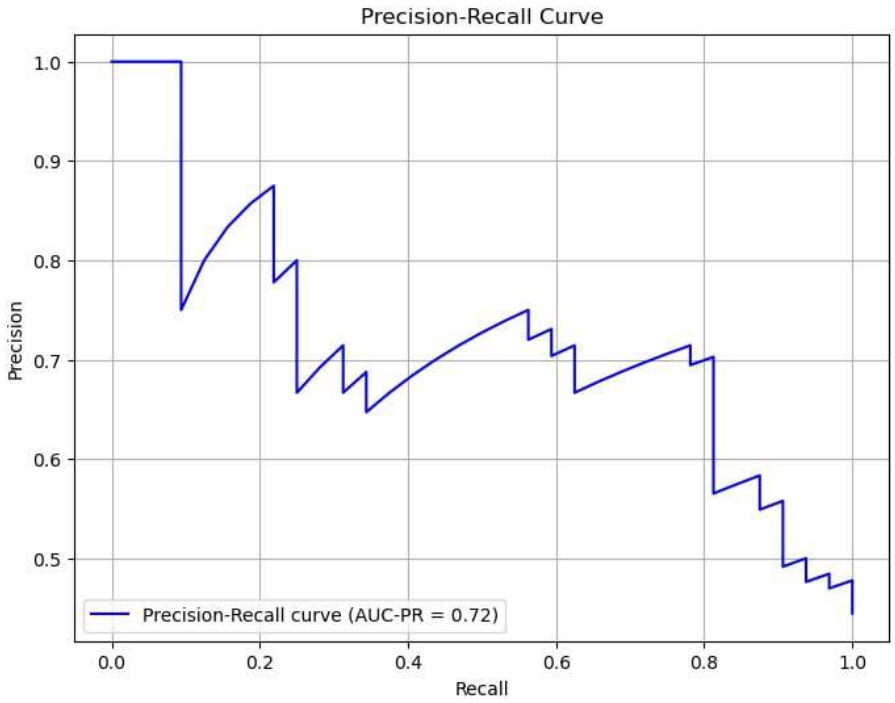

Figura B.33: Curva *precision-recall* caso selección de respuestas ante estímulos tipo *flash Normal*, con ventanas de 25 muestras de largo.

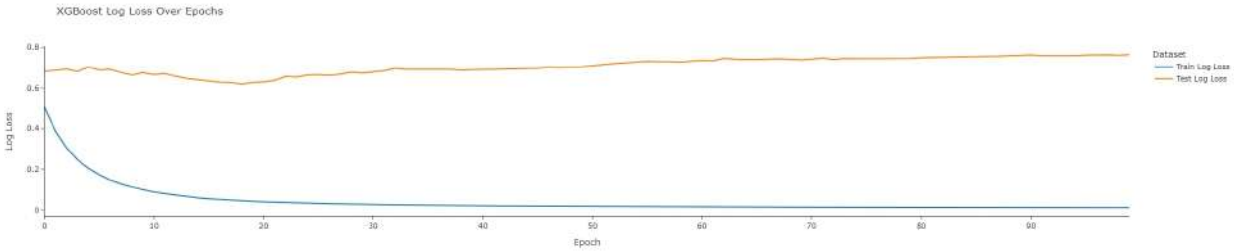

Figura B.34: Curvas de pérdida en el conjunto de entrenamiento y prueba caso selección de respuestas ante estímulos tipo *flash Normal*, con ventanas de 25 muestras de largo.

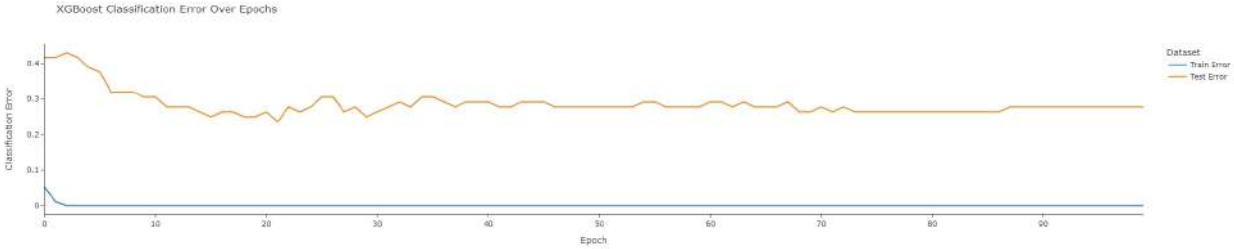

Figura B.35: Curvas de error en el conjunto de entrenamiento y prueba caso selección de respuestas ante estímulos tipo *flash Normal*, con ventanas de 25 muestras de largo.

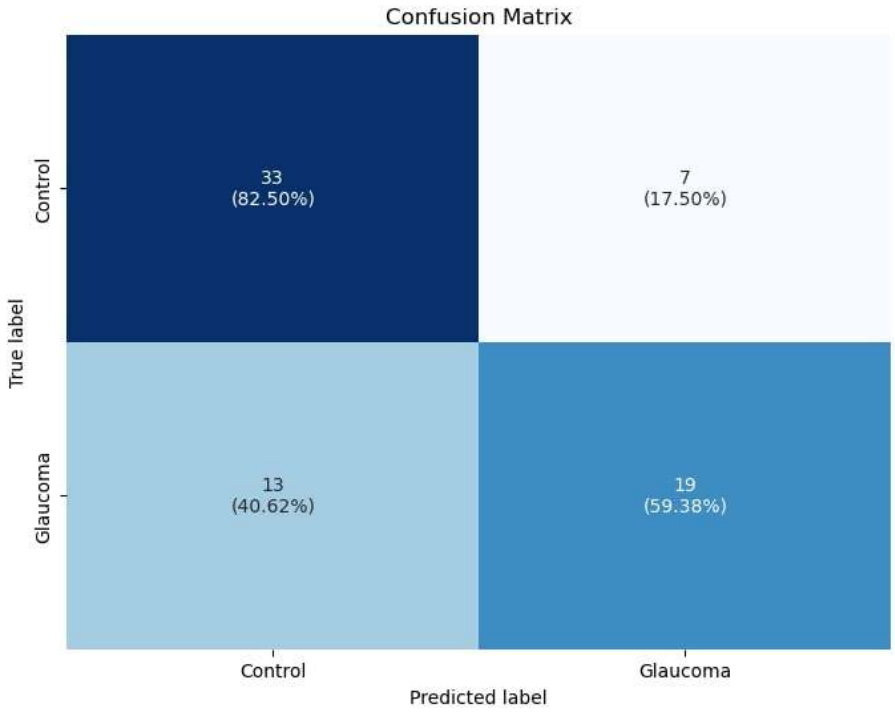

Figura B.36: Matriz de confusión caso selección de respuestas ante estímulos tipo *flash Filtrado + D.A.*, con ventanas de 50 muestras de largo.

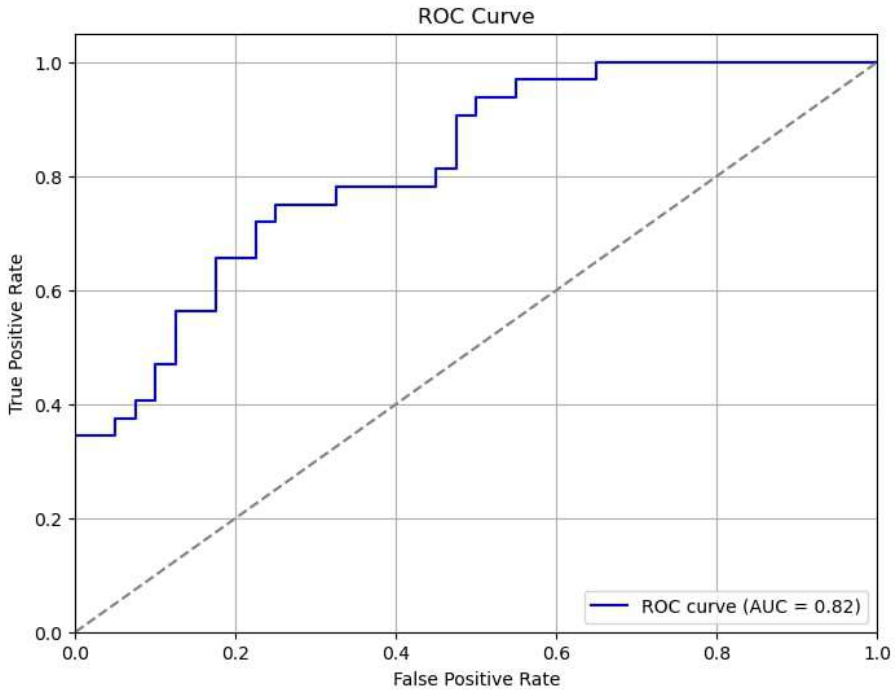

Figura B.37: Curva ROC caso selección de respuestas ante estímulos tipo *flash Filtrado + D.A.*, con ventanas de 50 muestras de largo.

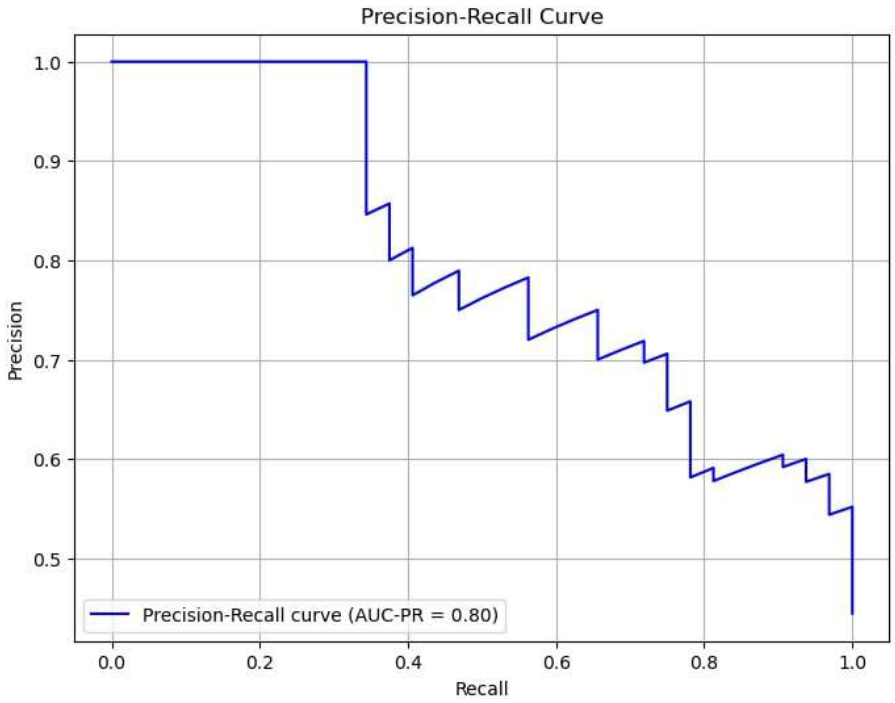

Figura B.38: Curva *precision-recall* caso selección de respuestas ante estímulos tipo *flash Filtrado + D.A.*, con ventanas de 50 muestras de largo.

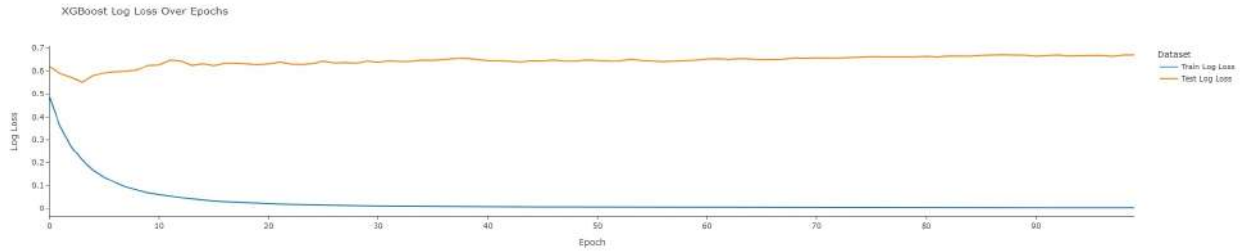

Figura B.39: Curvas de pérdida en el conjunto de entrenamiento y prueba caso selección de respuestas ante estímulos tipo *flash Filtrado + D.A.*, con ventanas de 50 muestras de largo.

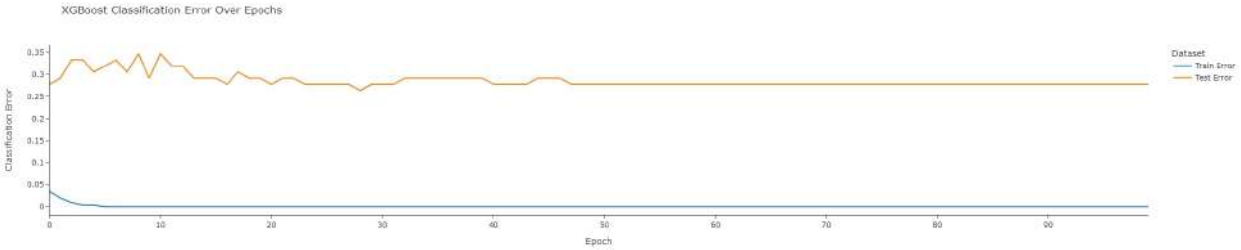

Figura B.40: Curvas de error en el conjunto de entrenamiento y prueba caso selección de respuestas ante estímulos tipo *flash Filtrado + D.A.*, con ventanas de 50 muestras de largo.

B.5. Resultados respuestas ante estímulos tipo rampa

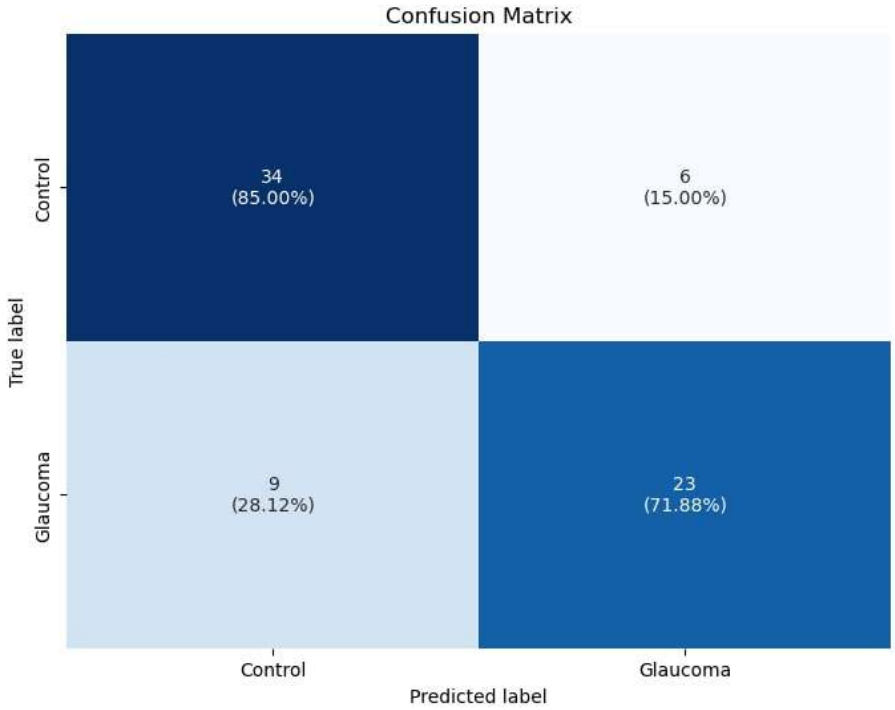

Figura B.41: Matriz de confusión caso selección de respuestas ante estímulos tipo rampa *Filtrado*, con ventanas de 150 muestras de largo.

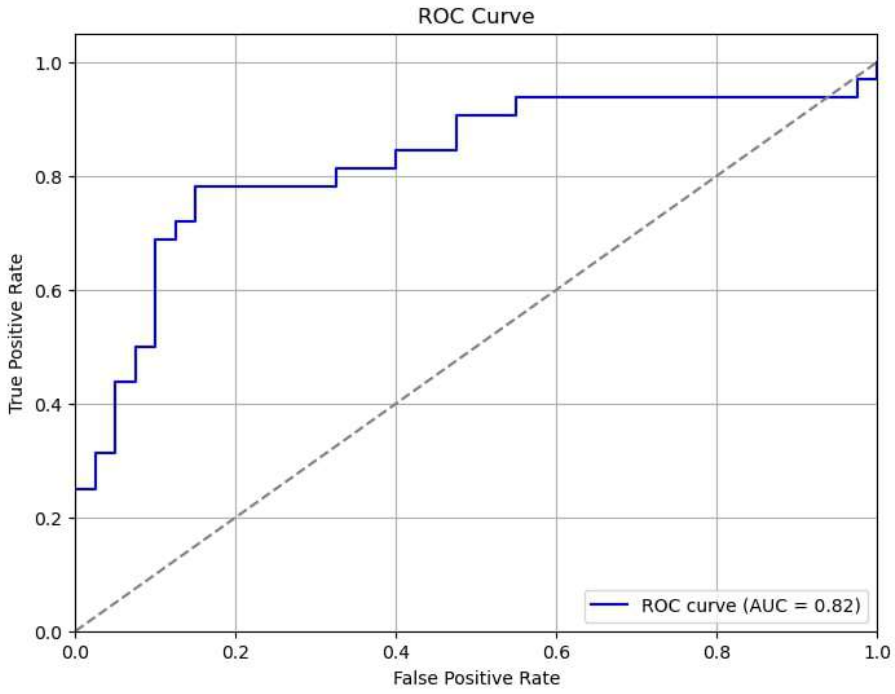

Figura B.42: Curva ROC caso selección de respuestas ante estímulos tipo rampa *Filtrado*, con ventanas de 150 muestras de largo.

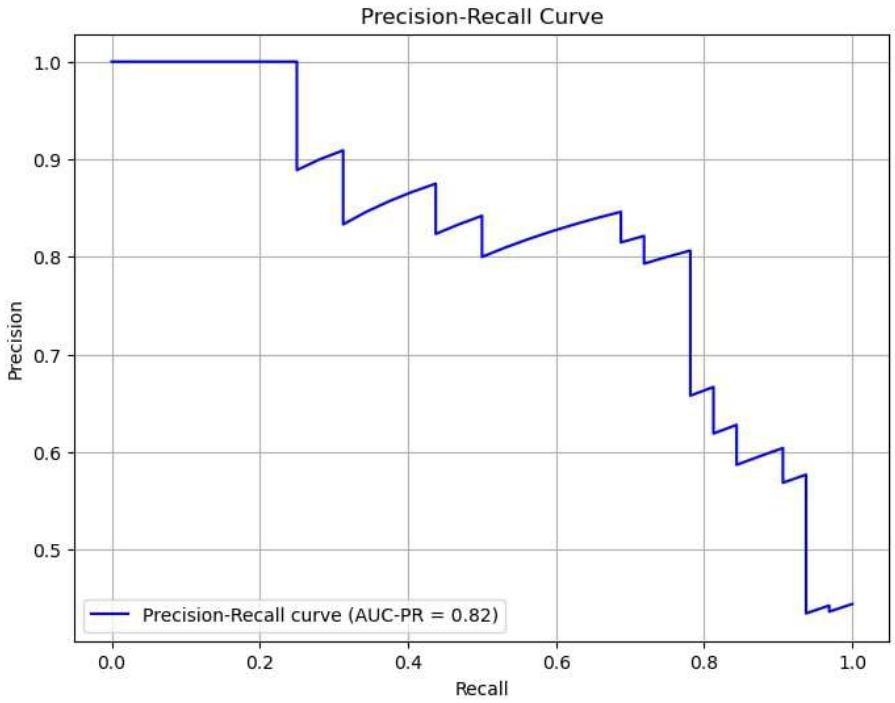

Figura B.43: Curva *precision-recall* caso selección de respuestas ante estímulos tipo rampa *Filtrado*, con ventanas de 150 muestras de largo.

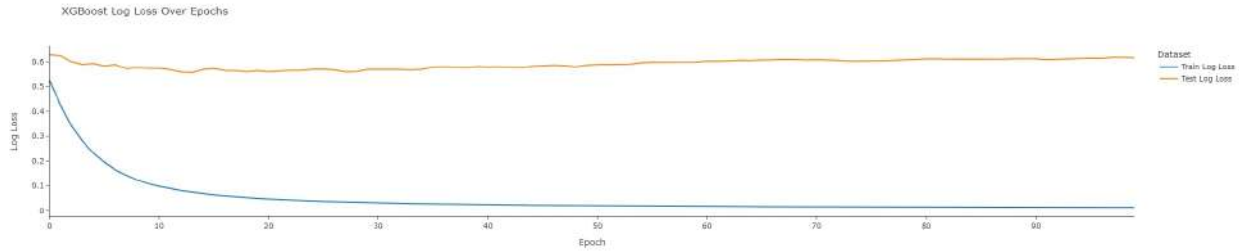

Figura B.44: Curvas de pérdida en el conjunto de entrenamiento y prueba caso selección de respuestas ante estímulos tipo rampa *Filtrado*, con ventanas de 150 muestras de largo.

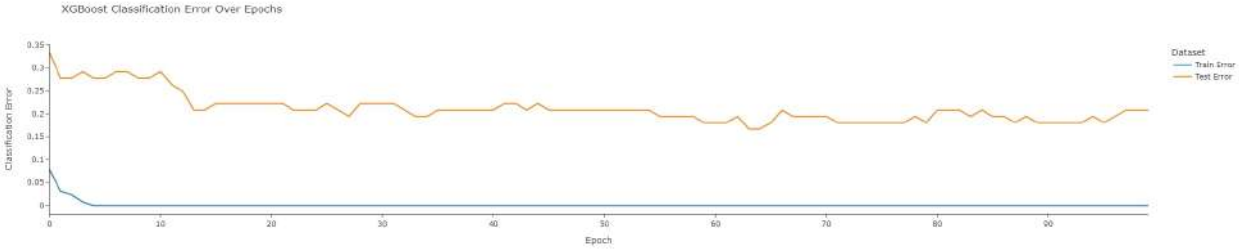

Figura B.45: Curvas de error en el conjunto de entrenamiento y prueba caso selección de respuestas ante estímulos tipo rampa *Filtrado*, con ventanas de 150 muestras de largo.

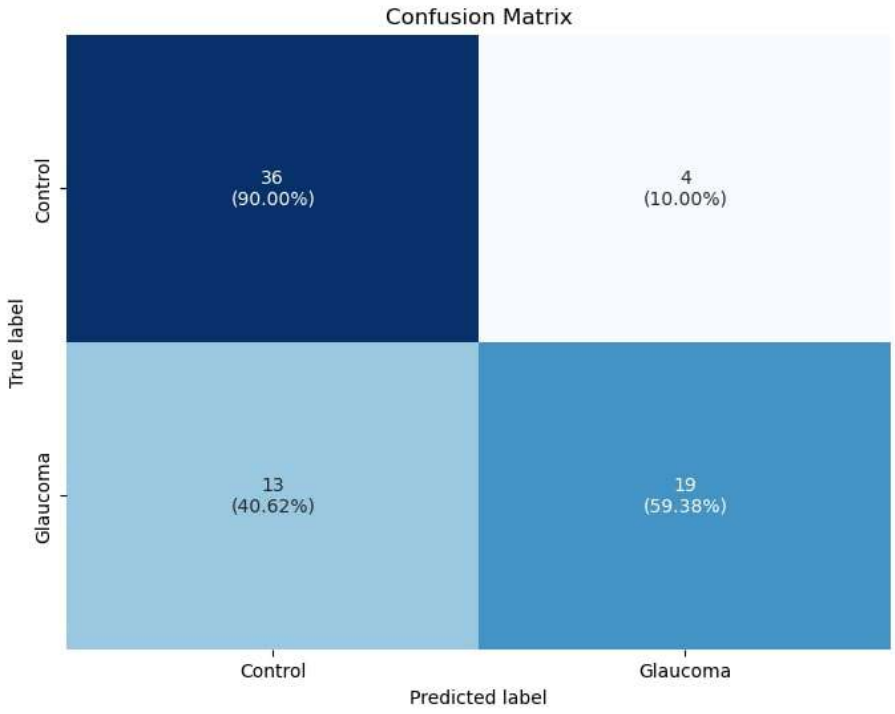

Figura B.46: Matriz de confusión caso selección de respuestas ante estímulos tipo rampa *Filtrado + D.A.*, con ventanas de 100 muestras de largo.

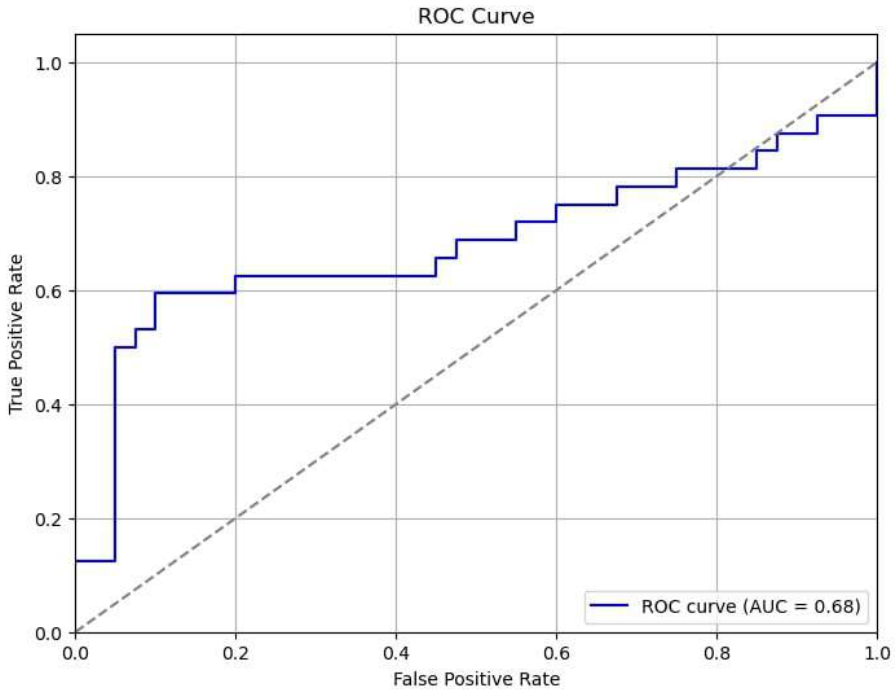

Figura B.47: Curva ROC caso selección de respuestas ante estímulos tipo rampa *Filtrado + D.A.*, con ventanas de 100 muestras de largo.

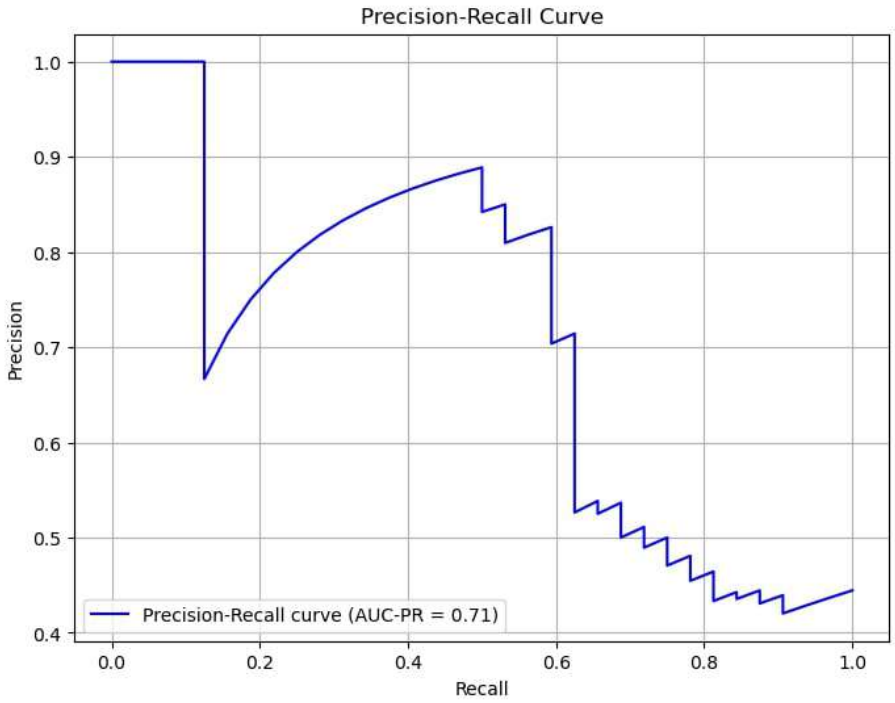

Figura B.48: Curva *precision-recall* caso selección de respuestas ante estímulos tipo rampa *Filtrado + D.A.*, con ventanas de 100 muestras de largo.

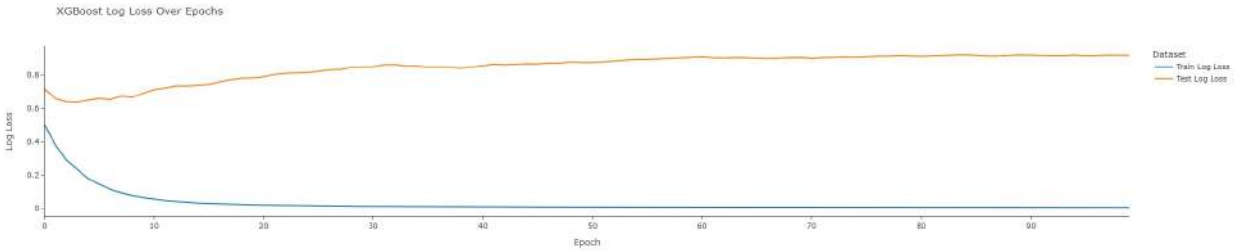

Figura B.49: Curvas de pérdida en el conjunto de entrenamiento y prueba caso selección de respuestas ante estímulos tipo rampa *Filtrado + D.A.*, con ventanas de 100 muestras de largo.

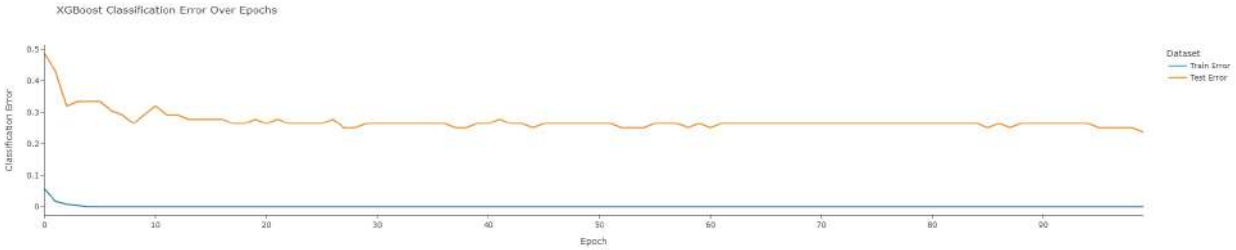

Figura B.50: Curvas de error en el conjunto de entrenamiento y prueba caso selección de respuestas ante estímulos tipo rampa *Filtrado + D.A.*, con ventanas de 100 muestras de largo.

Anexo C. Resultados Modelos de Deep Learning

C.1. Resultados FCN

C.1.1. Resultados caso base

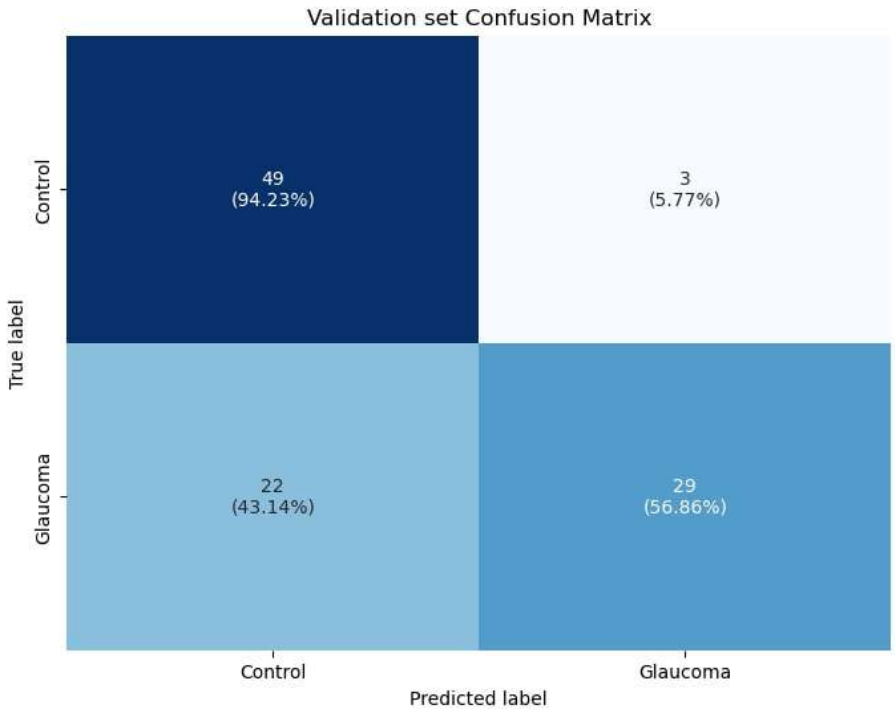

Figura C.1: Matriz de confusión del conjunto de validación, utilizando FCN y el caso base.

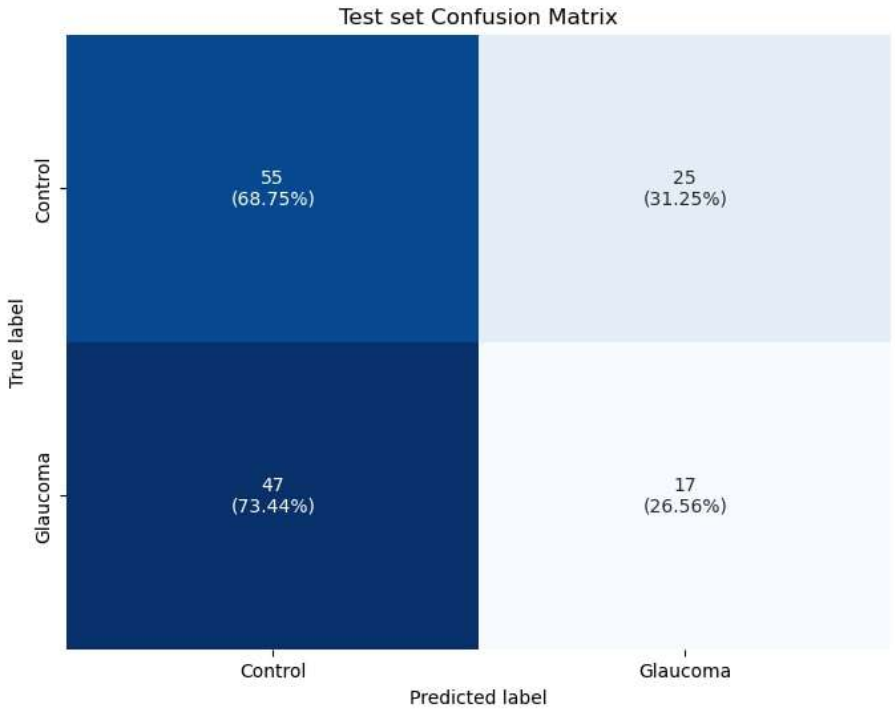

Figura C.2: Matriz de confusión del conjunto de prueba, utilizando FCN y el caso base.

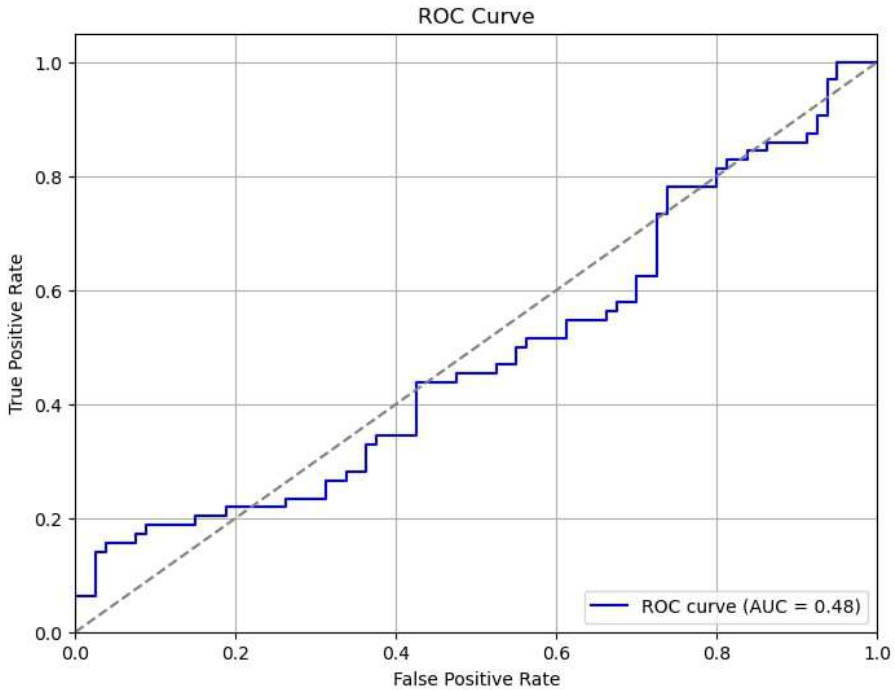

Figura C.3: Curva ROC, utilizando FCN y el caso base.

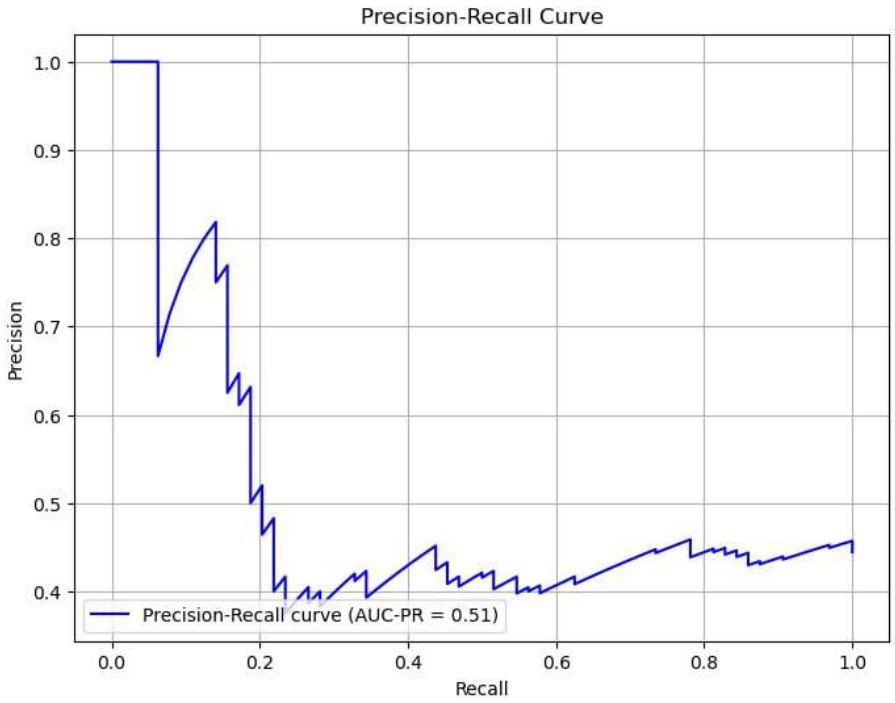

Figura C.4: Curva *precision-recall*, utilizando FCN y el caso base.

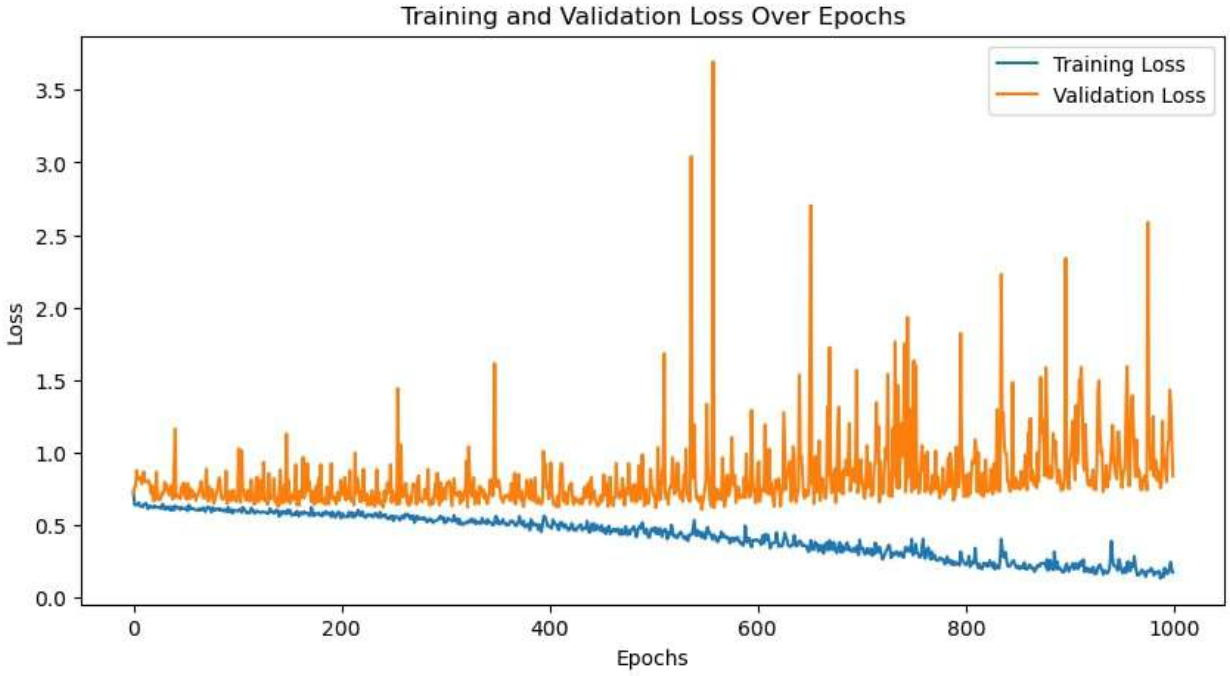

Figura C.5: Curvas de pérdida en el conjunto de entrenamiento y validación, utilizando FCN y el caso base.

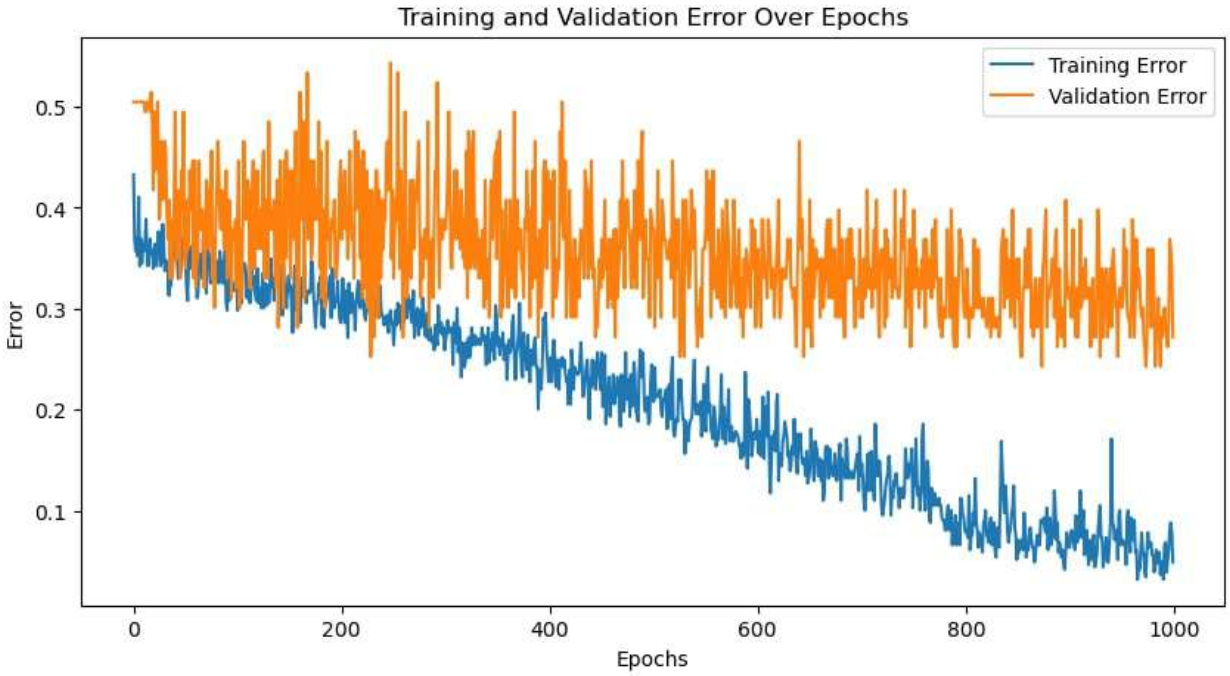

Figura C.6: Curvas de error en el conjunto de entrenamiento y validación, utilizando FCN y el caso base.

C.1.2. Resultados caso filtrado

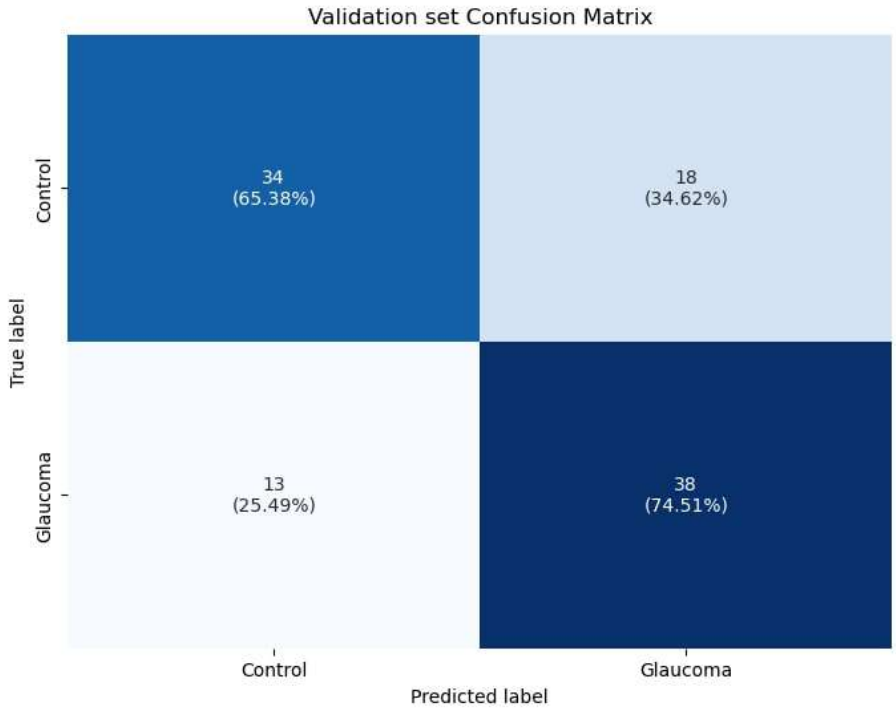

Figura C.7: Matriz de confusión del conjunto de validación, utilizando FCN y el caso filtrado.

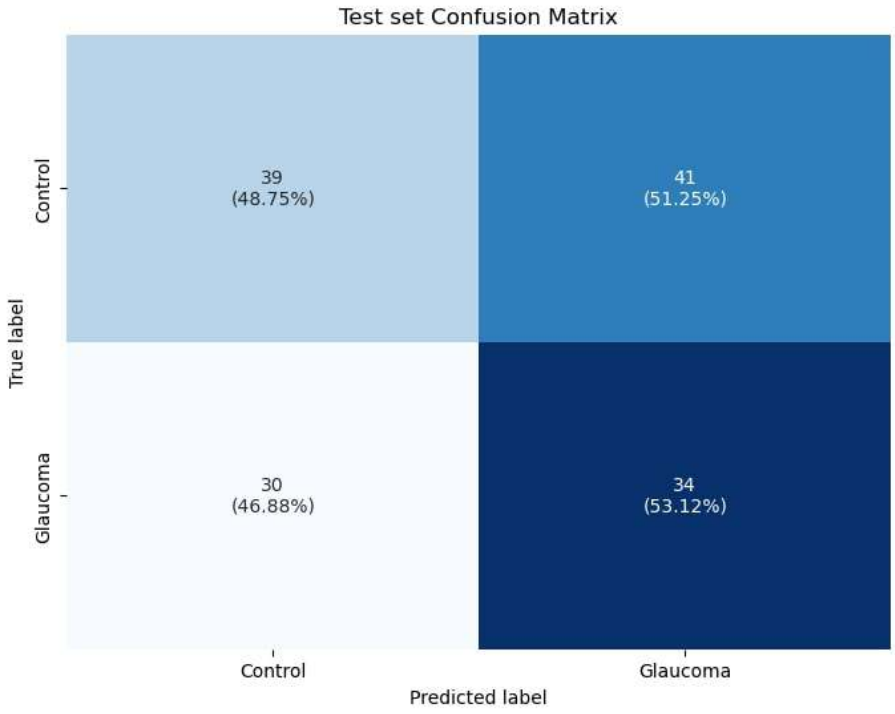

Figura C.8: Matriz de confusión del conjunto de prueba, utilizando FCN y el caso filtrado.

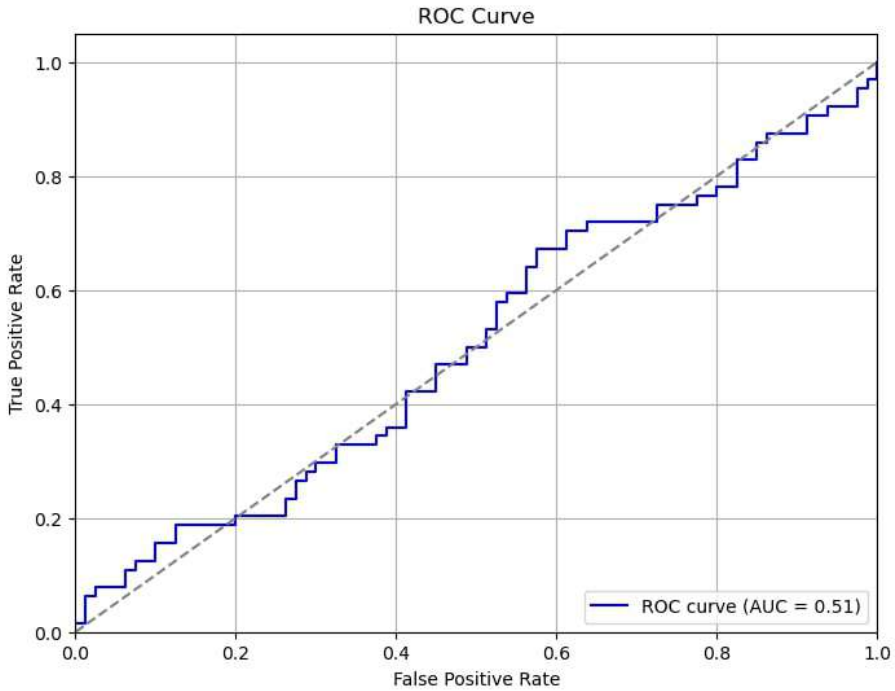

Figura C.9: Curva ROC, utilizando FCN y el caso filtrado.

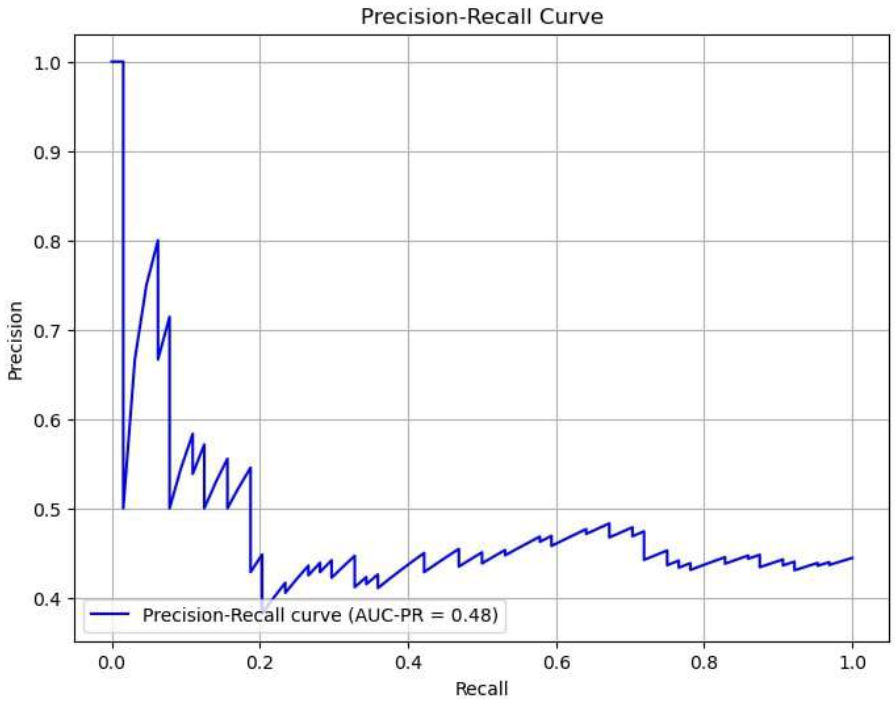

Figura C.10: Curva *precision-recall*, utilizando FCN y el caso filtrado.

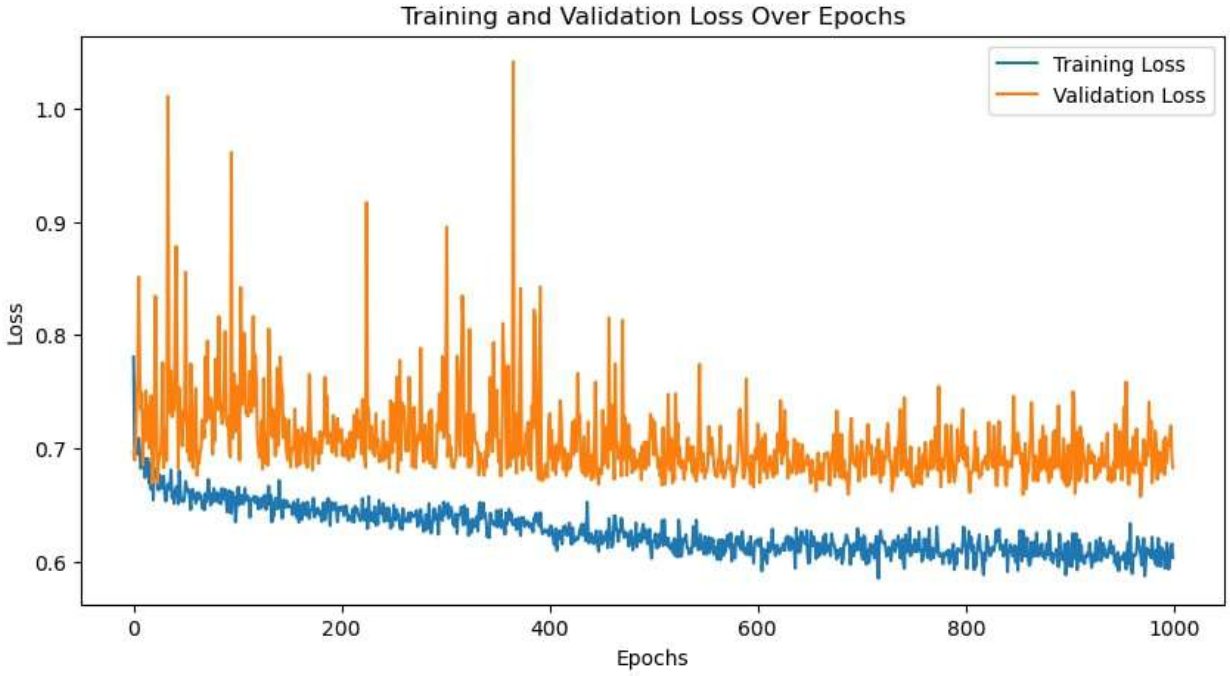

Figura C.11: Curvas de pérdida en el conjunto de entrenamiento y validación, utilizando FCN y el caso filtrado.

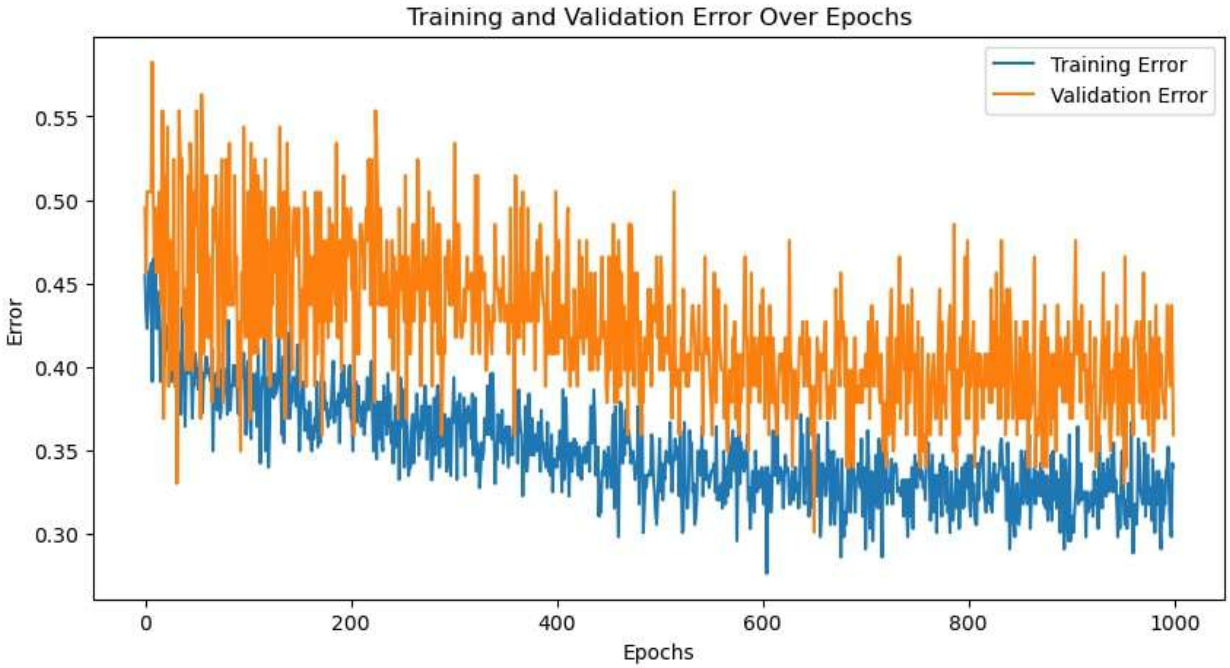

Figura C.12: Curvas de error en el conjunto de entrenamiento y validación, utilizando FCN y el caso filtrado.

C.1.3. Resultados caso rampa

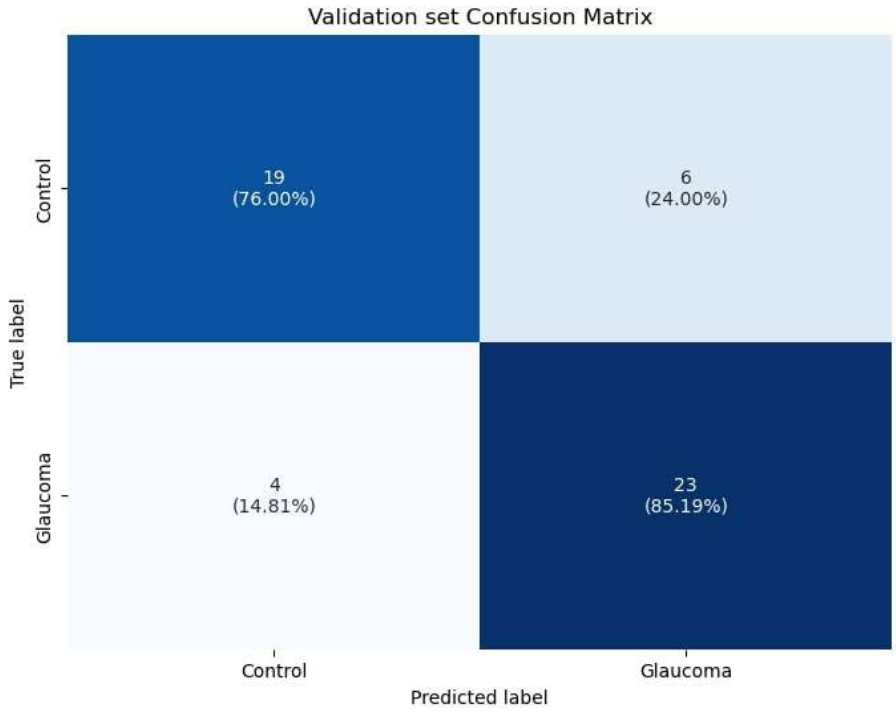

Figura C.13: Matriz de confusión del conjunto de validación, utilizando FCN y el caso rampa.

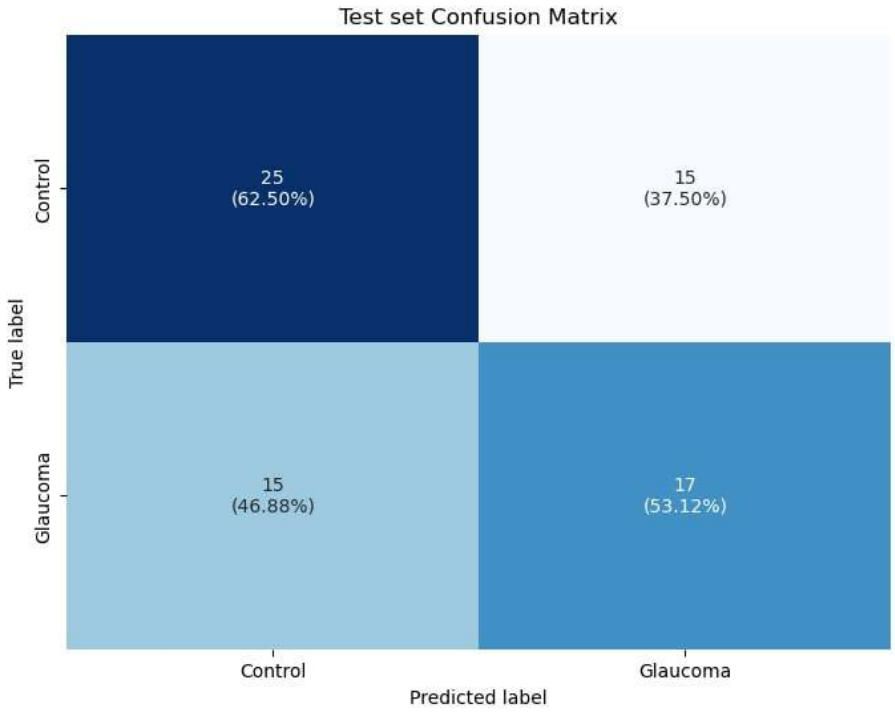

Figura C.14: Matriz de confusión del conjunto de prueba, utilizando FCN y el caso rampa.

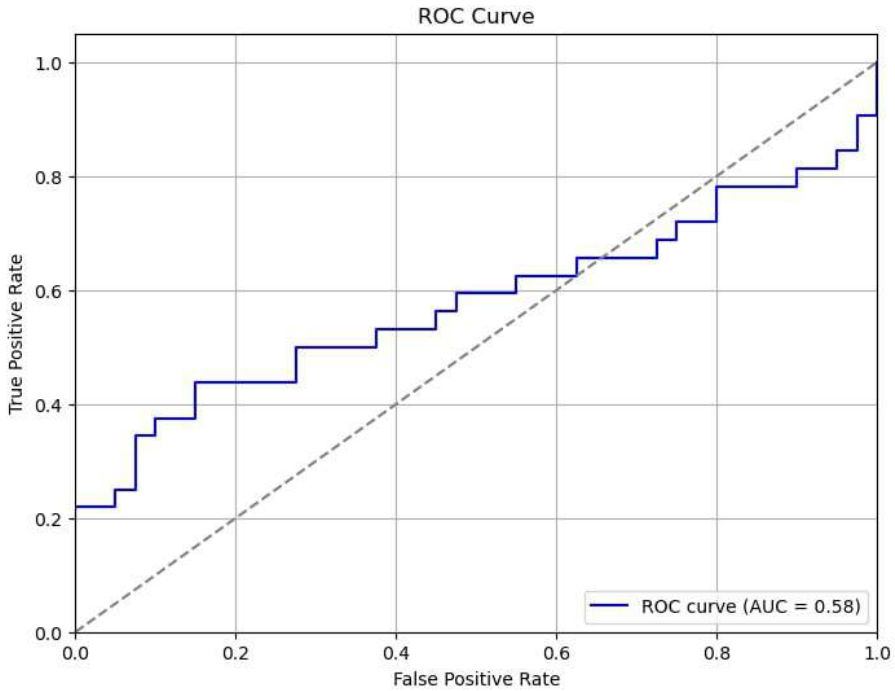

Figura C.15: Curva ROC, utilizando FCN y el caso rampa.

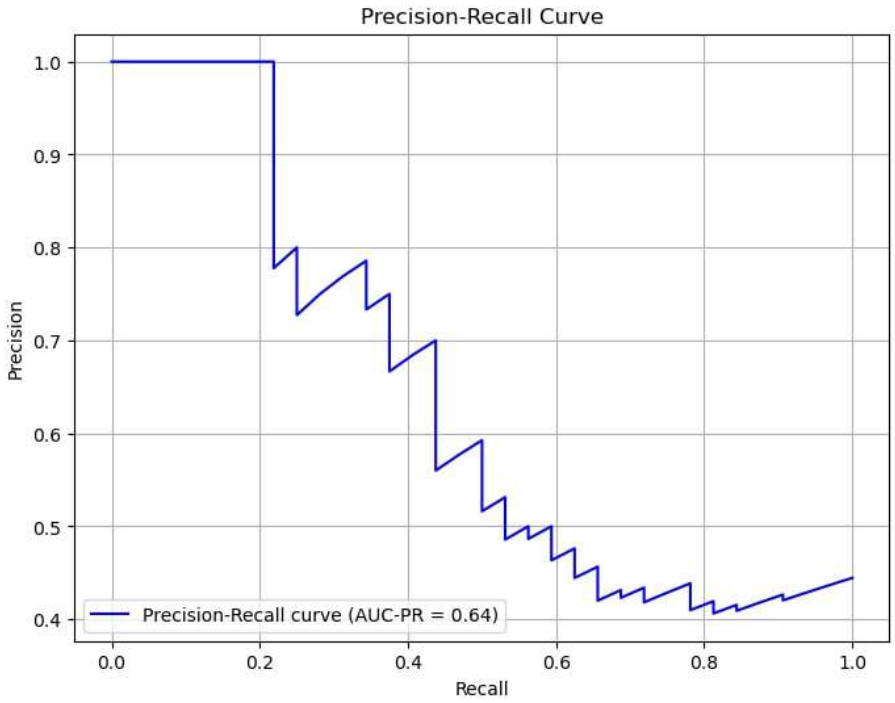

Figura C.16: Curva *precision-recall*, utilizando FCN y el caso rampa.

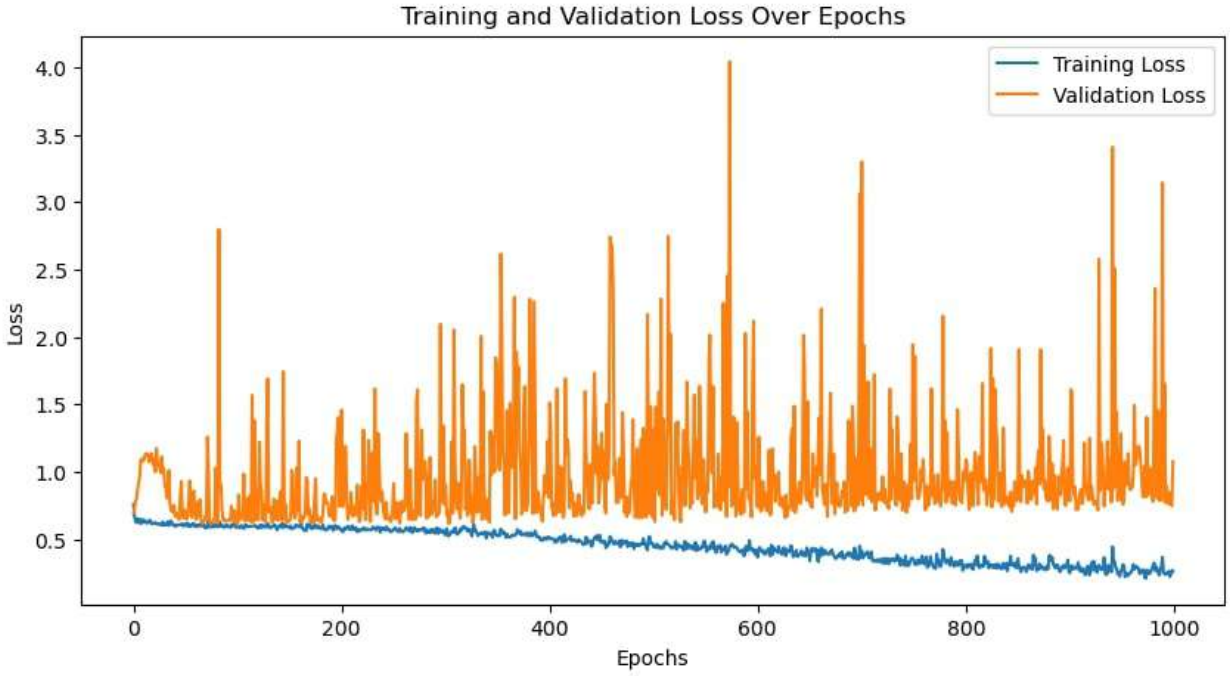

Figura C.17: Curvas de pérdida en el conjunto de entrenamiento y validación, utilizando FCN y el caso rampa.

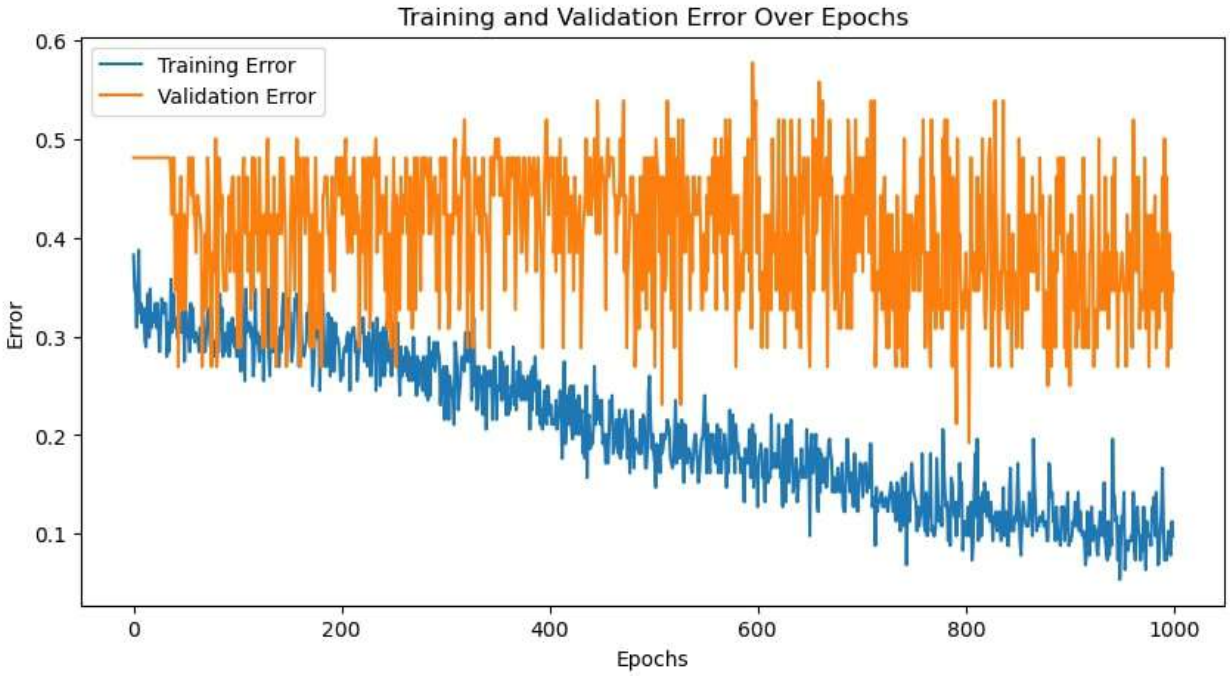

Figura C.18: Curvas de error en el conjunto de entrenamiento y validación, utilizando FCN y el caso rampa.

C.1.4. Resultados caso *flash*

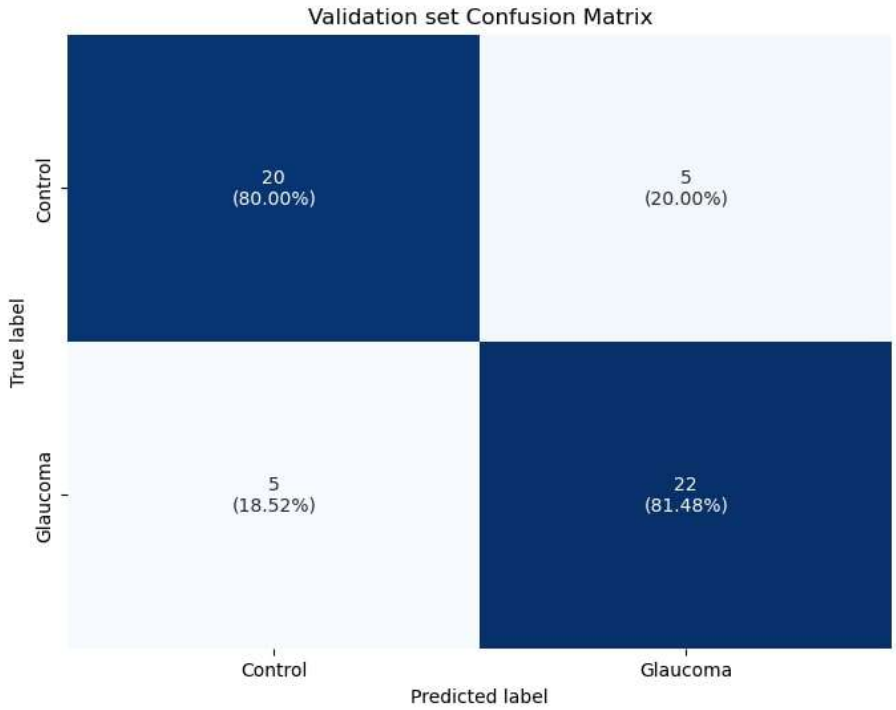

Figura C.19: Matriz de confusión del conjunto de validación, utilizando FCN y el caso *flash*.

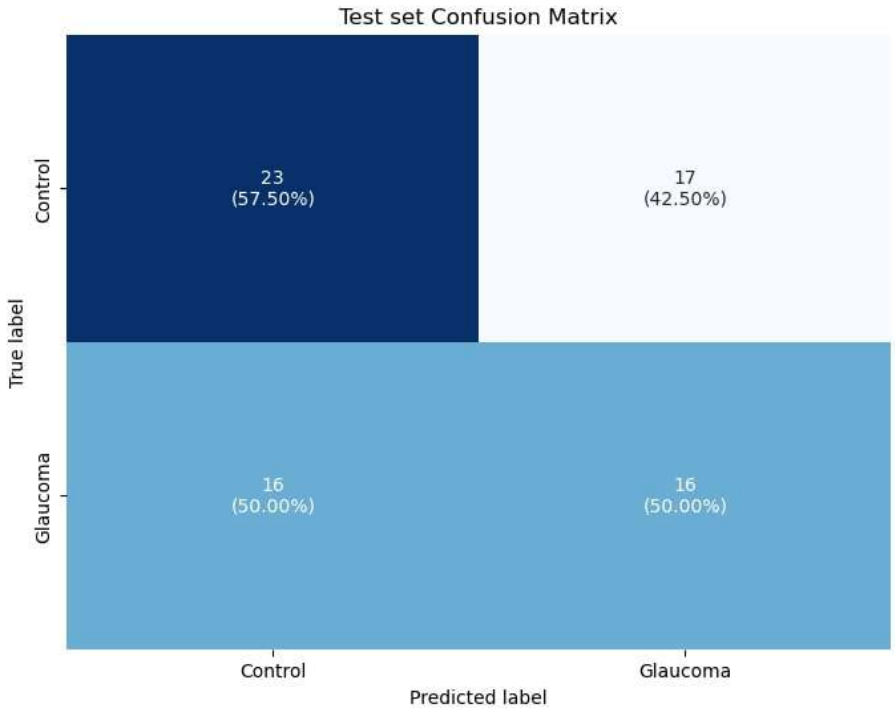

Figura C.20: Matriz de confusión del conjunto de prueba, utilizando FCN y el caso *flash*.

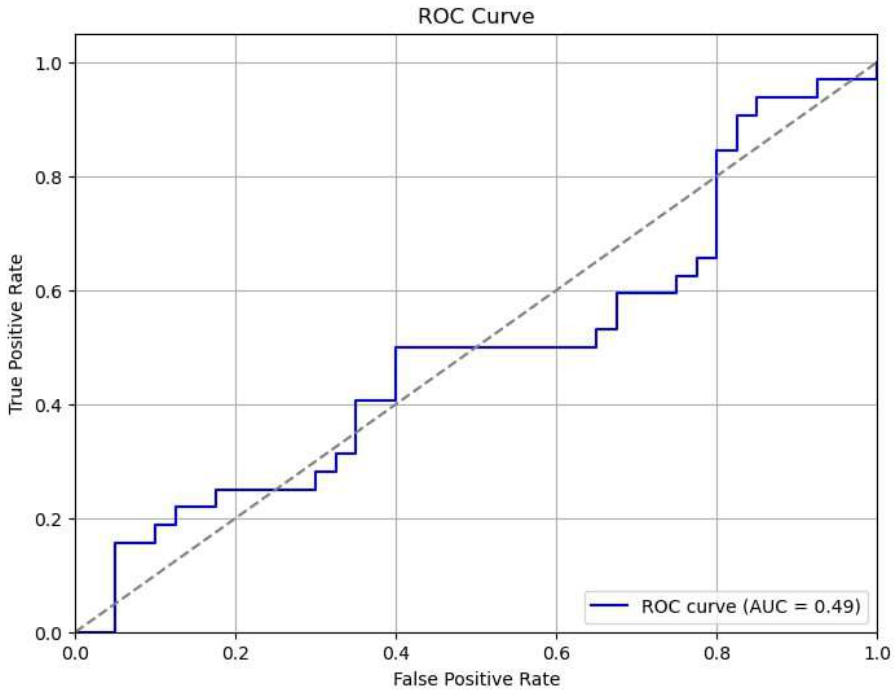

Figura C.21: Curva ROC, utilizando FCN y el caso *flash*.

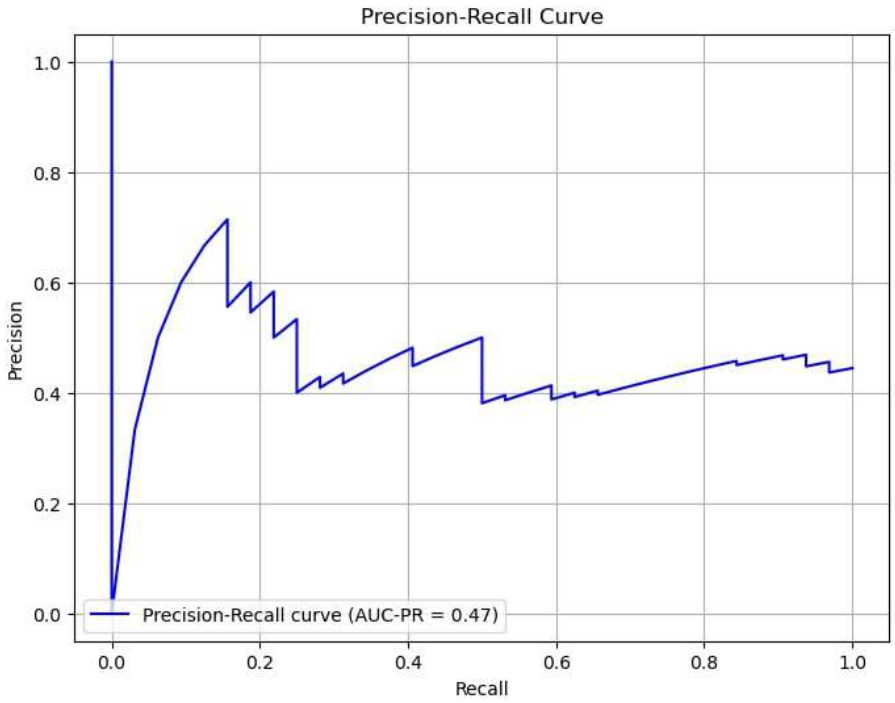

Figura C.22: Curva *precision-recall*, utilizando FCN y el caso *flash*.

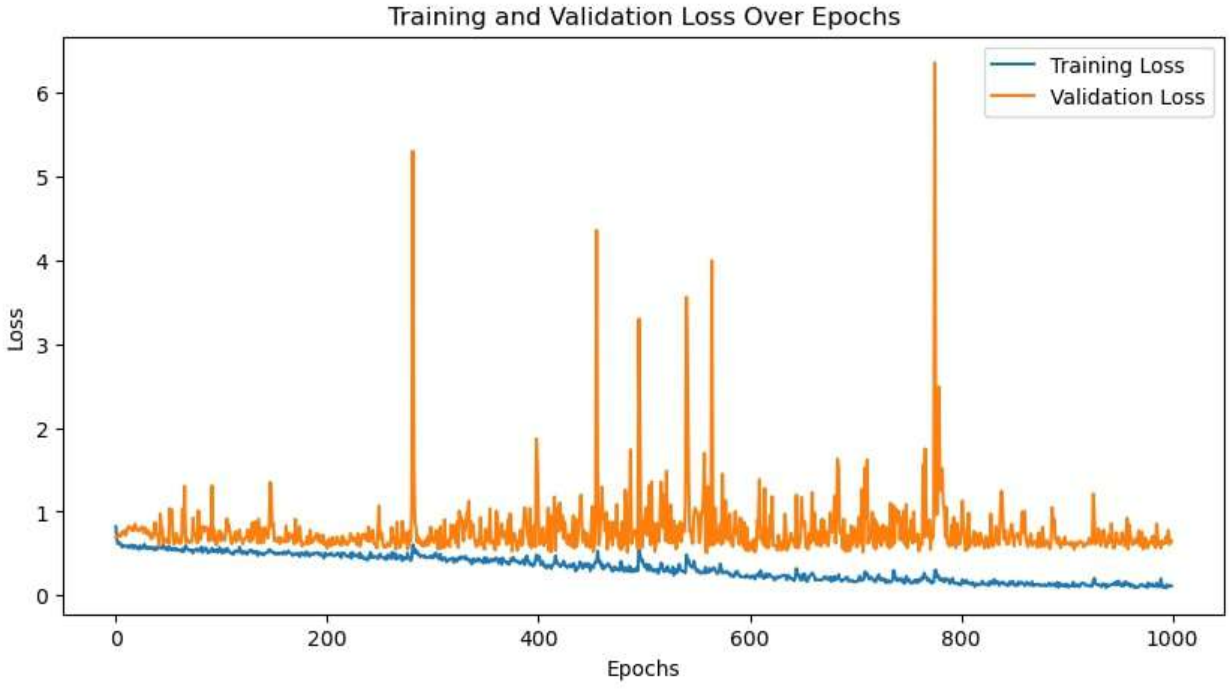

Figura C.23: Curvas de pérdida en el conjunto de entrenamiento y validación, utilizando FCN y el caso *flash*.

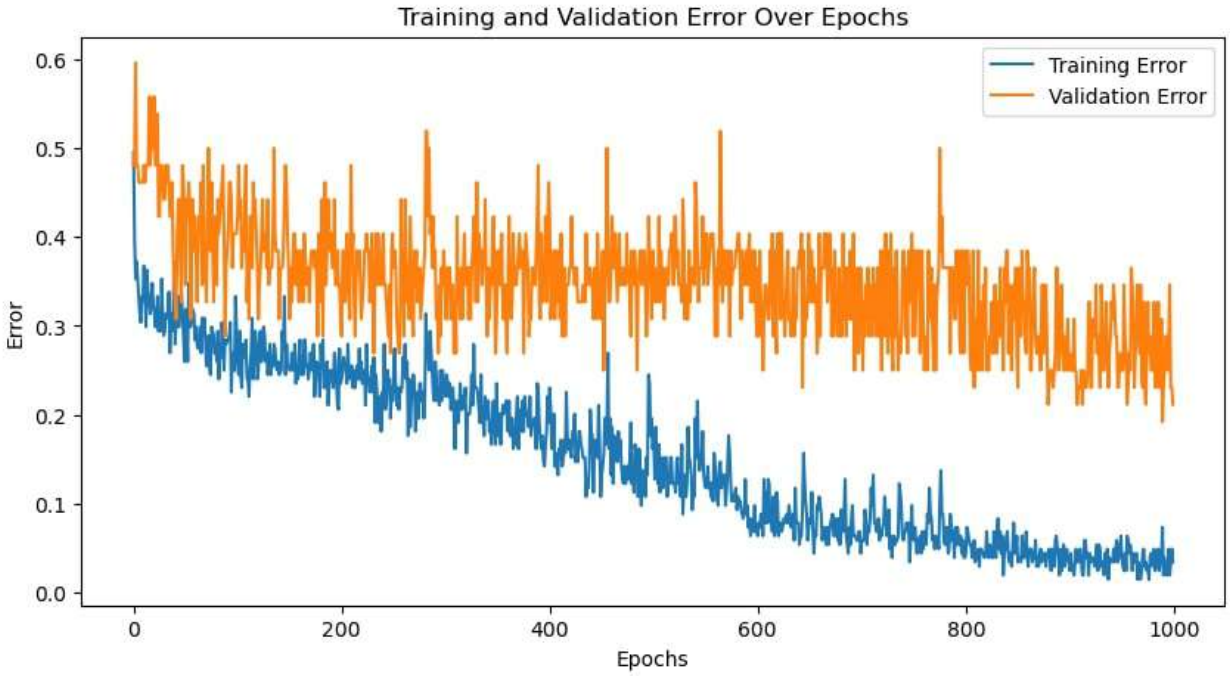

Figura C.24: Curvas de error en el conjunto de entrenamiento y validación, utilizando FCN y el caso *flash*.

C.1.5. Resultado caso rampa filtrado

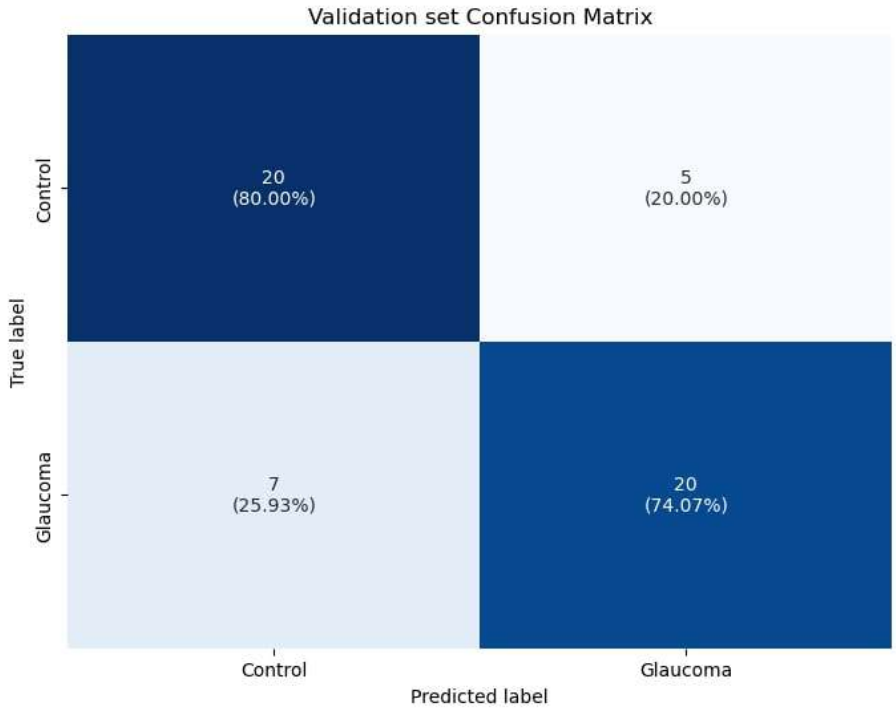

Figura C.25: Matriz de confusión del conjunto de validación, utilizando FCN y el caso rampa filtrado.

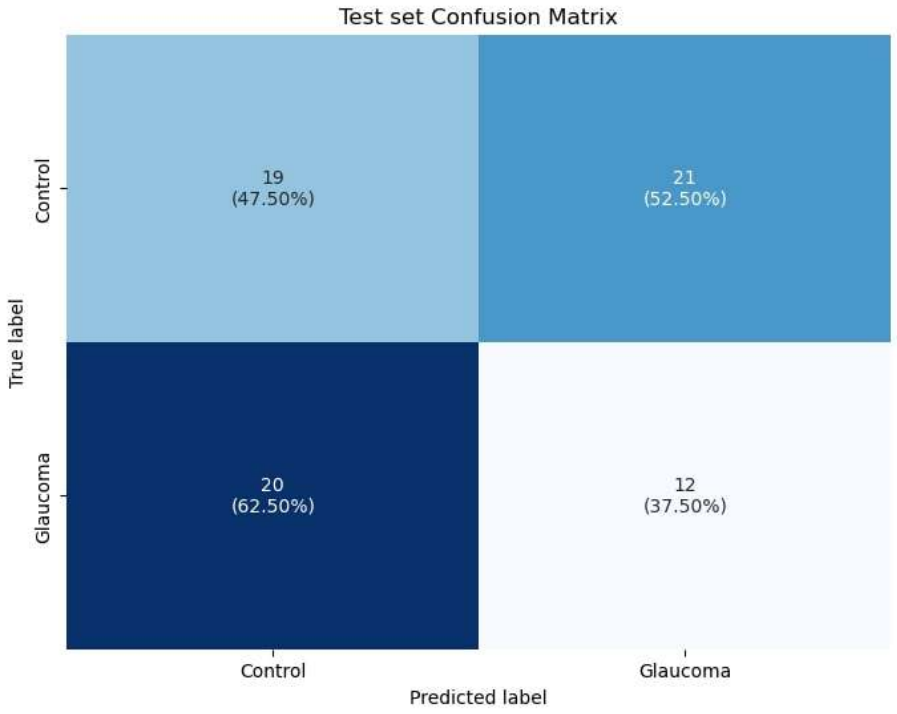

Figura C.26: Matriz de confusión del conjunto de prueba, utilizando FCN y el caso rampa filtrado.

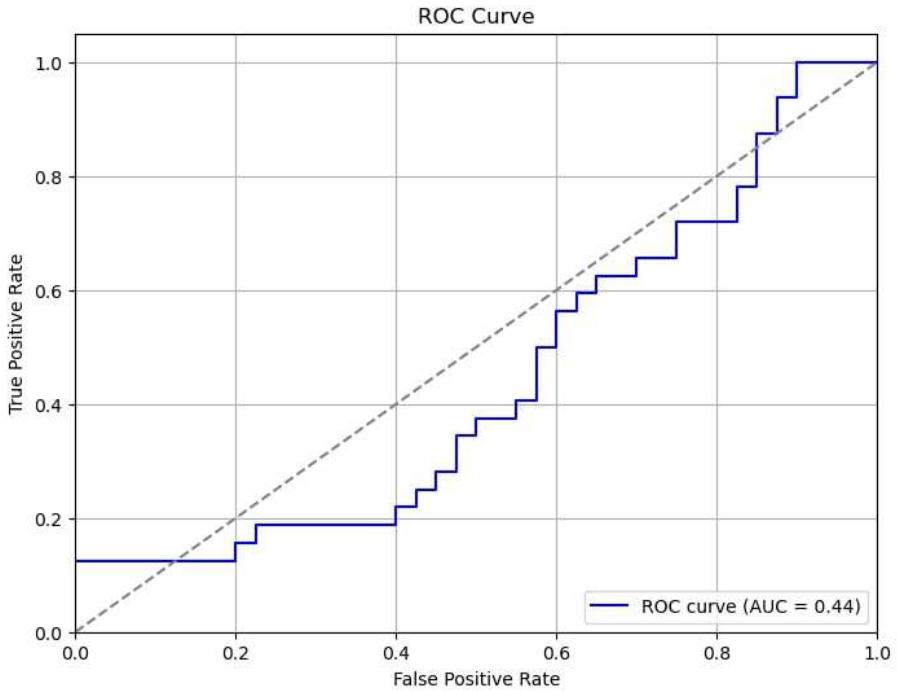

Figura C.27: Curva ROC, utilizando FCN y el caso rampa filtrado.

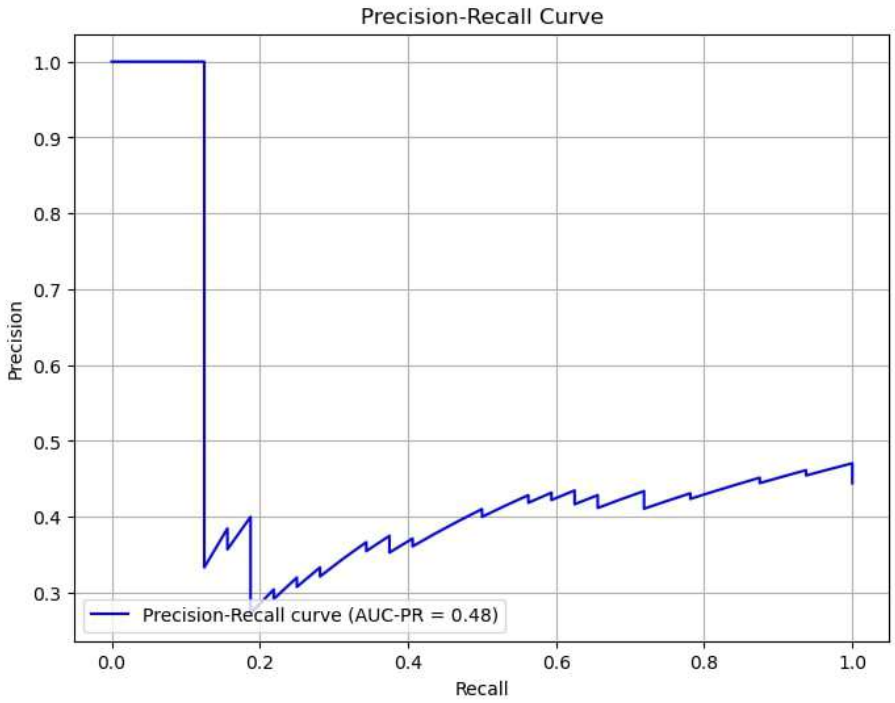

Figura C.28: Curva *precision-recall*, utilizando FCN y el caso rampa filtrado.

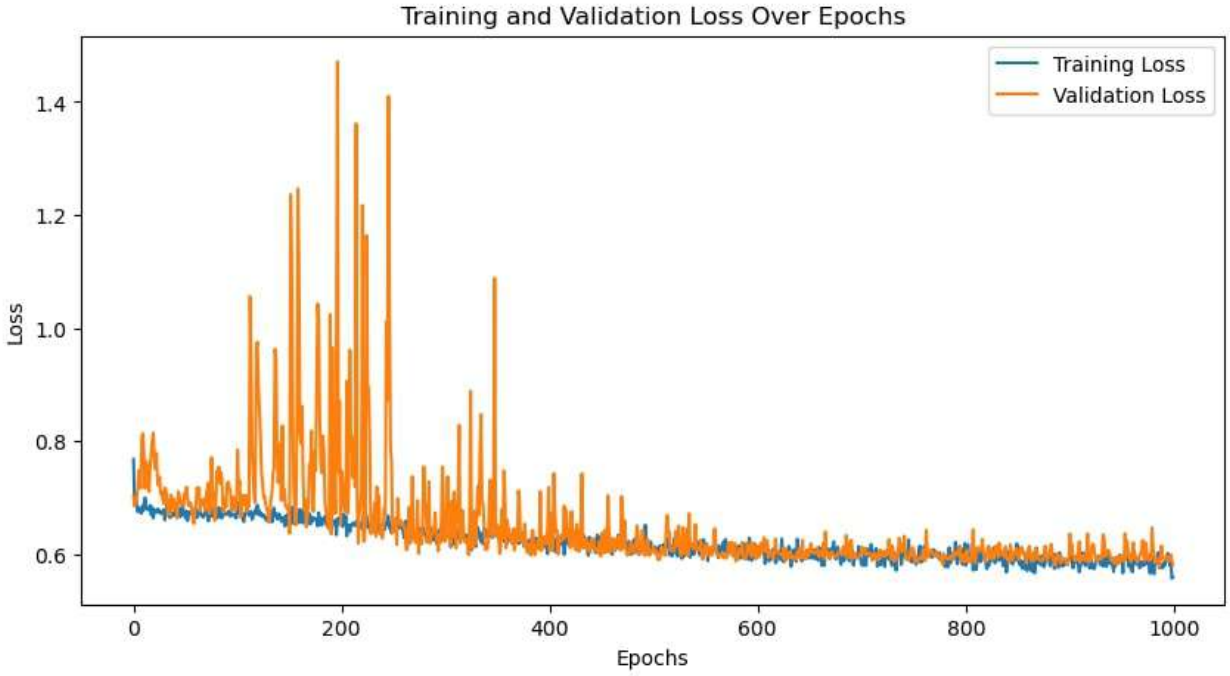

Figura C.29: Curvas de pérdida en el conjunto de entrenamiento y validación, utilizando FCN y el caso rampa filtrado.

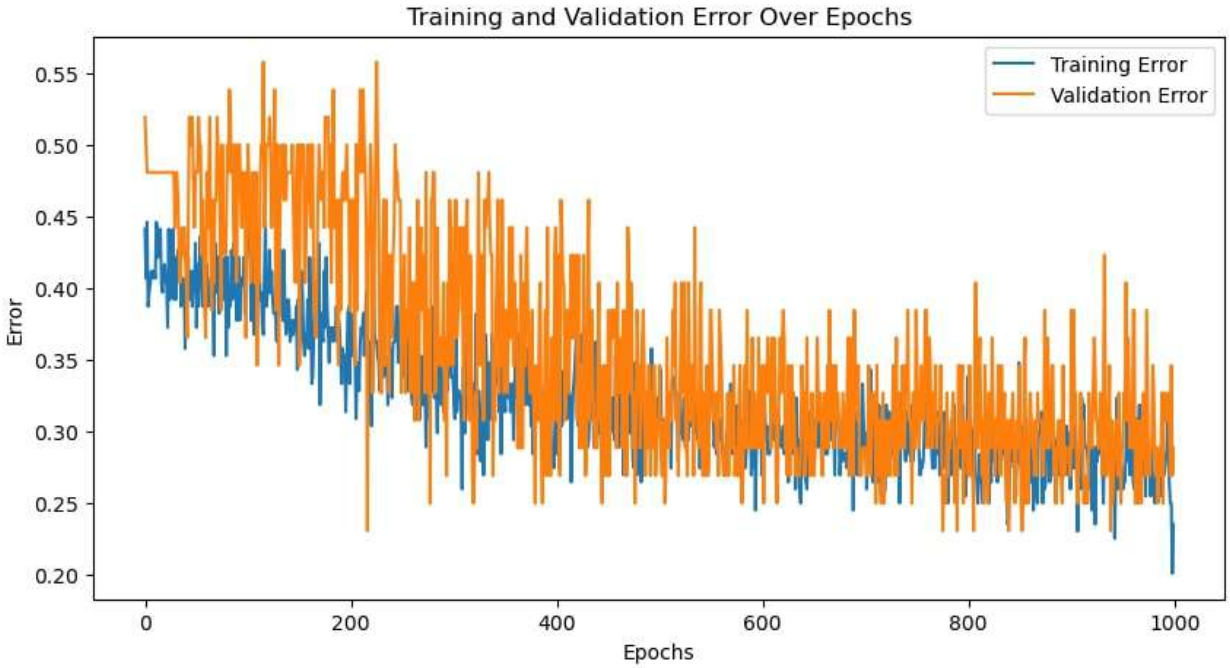

Figura C.30: Curvas de error en el conjunto de entrenamiento y validación, utilizando FCN y el caso rampa filtrado.

C.1.6. Resultados caso *flash* filtrado

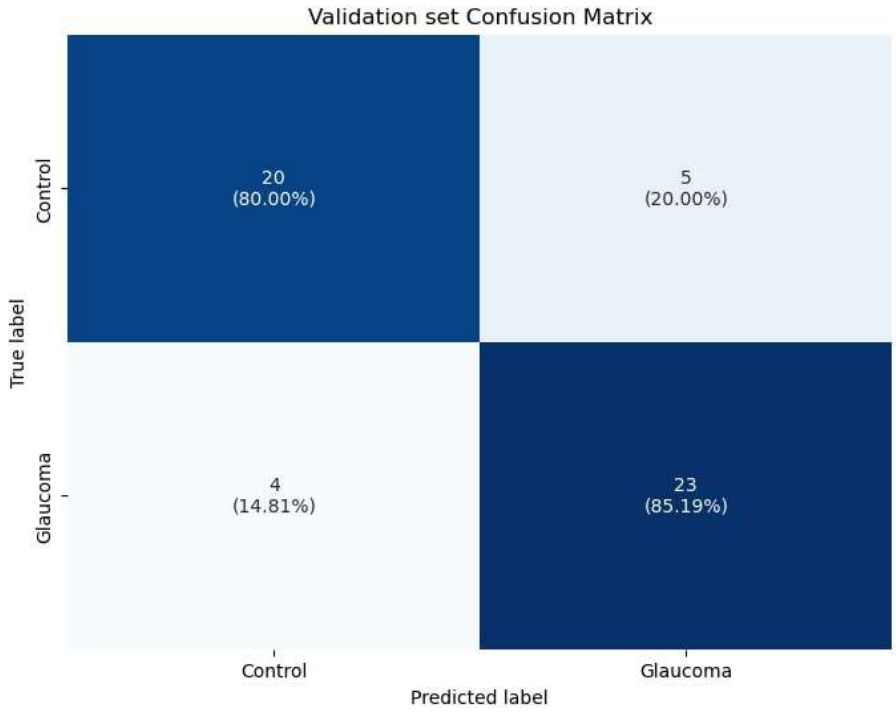

Figura C.31: Matriz de confusión del conjunto de validación, utilizando FCN y el caso *flash* filtrado.

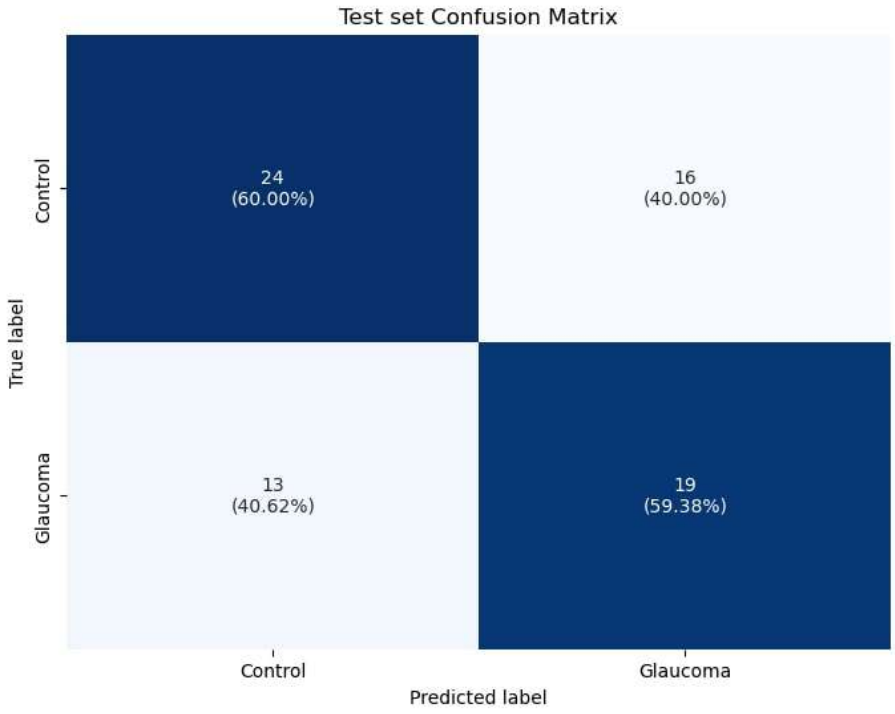

Figura C.32: Matriz de confusión del conjunto de prueba, utilizando FCN y el caso *flash* filtrado.

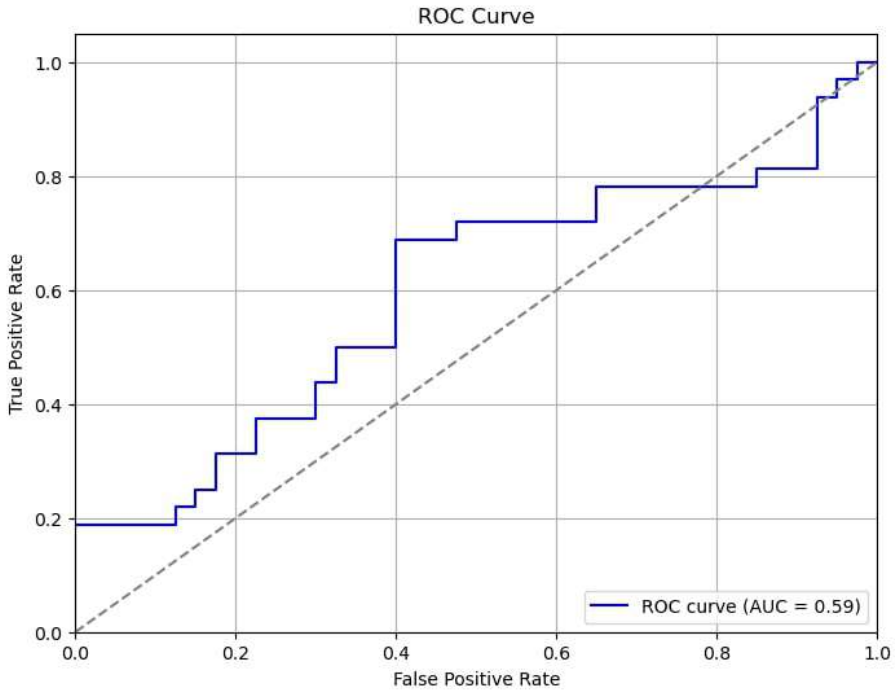

Figura C.33: Curva ROC, utilizando FCN y el caso *flash* filtrado.

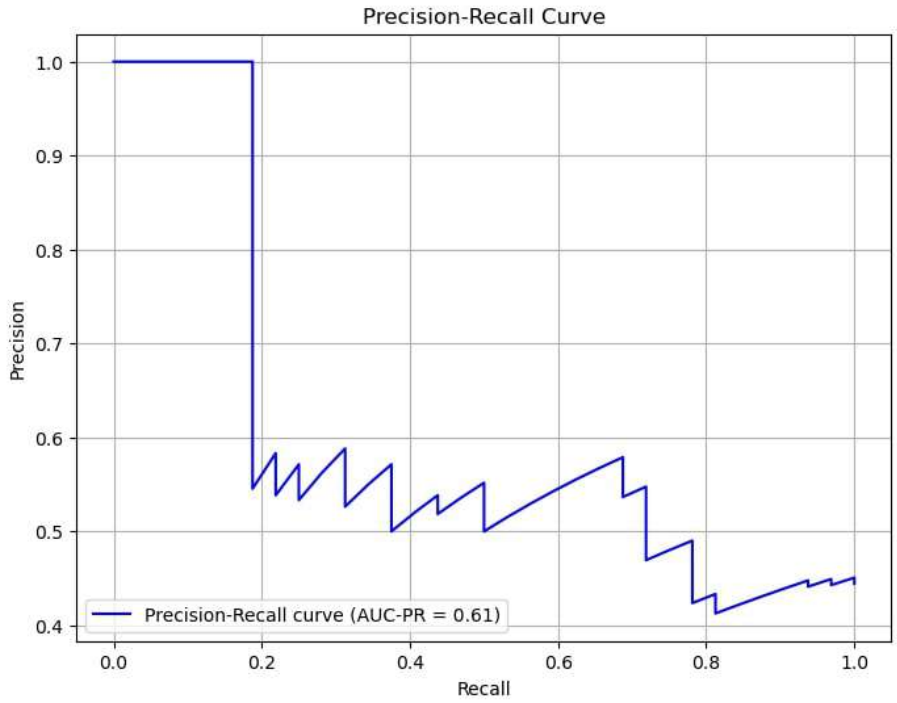

Figura C.34: Curva *precision-recall*, utilizando FCN y el caso *flash* filtrado.

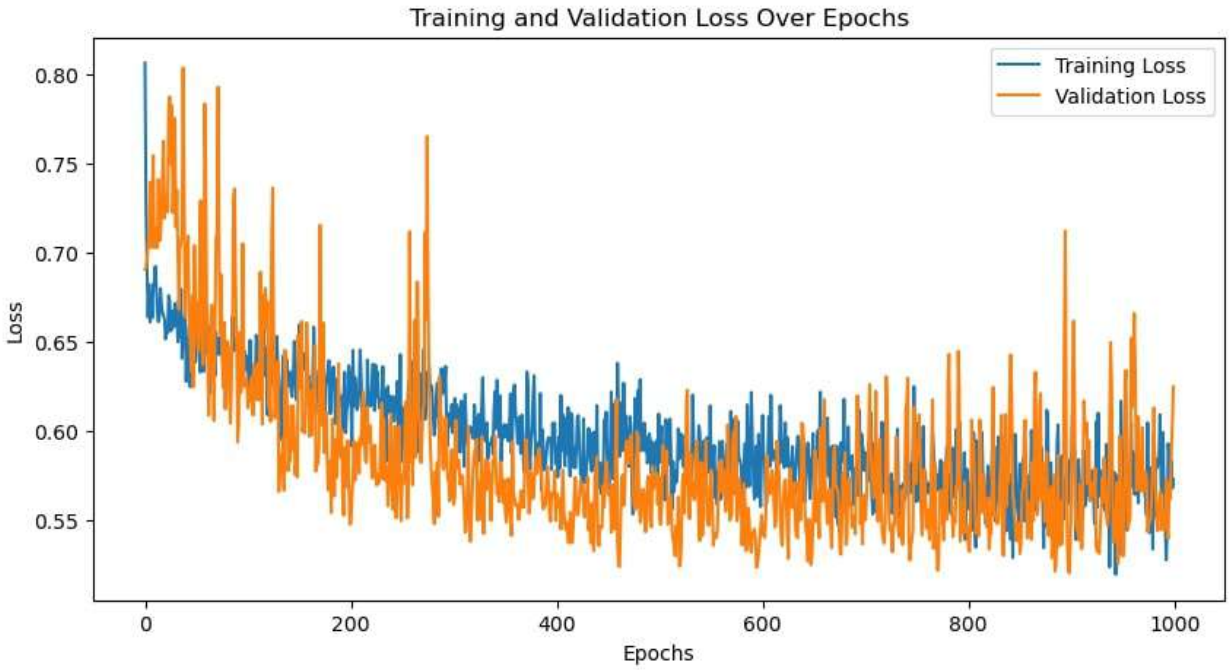

Figura C.35: Curvas de pérdida en el conjunto de entrenamiento y validación, utilizando FCN y el caso *flash* filtrado.

Figura C.36: Curvas de error en el conjunto de entrenamiento y validación, utilizando FCN y el caso *flash* filtrado.

C.2. Resultados GRU

C.2.1. Resultados caso base

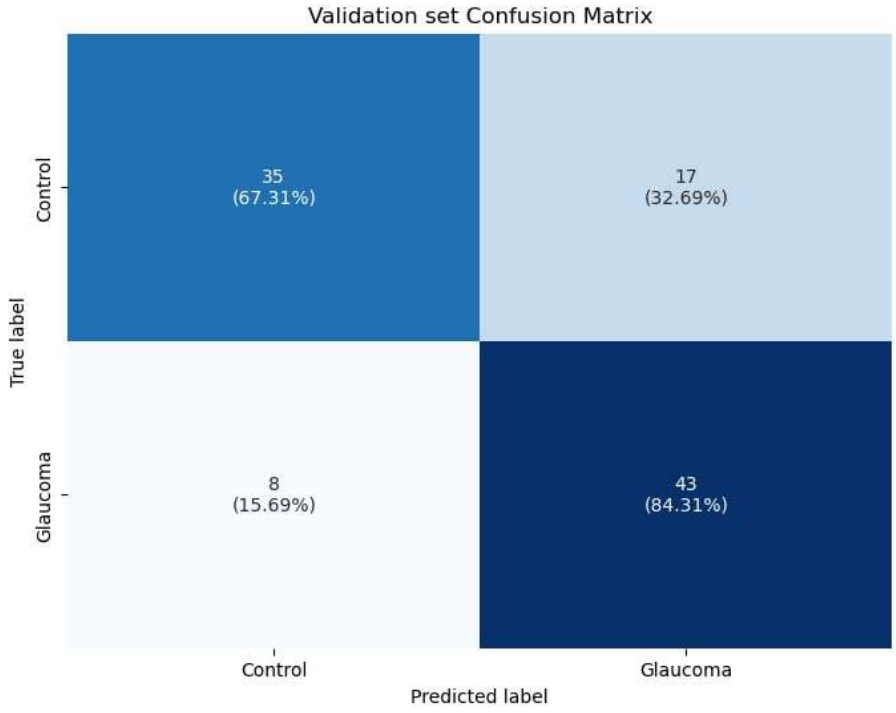

Figura C.37: Matriz de confusión del conjunto de validación, utilizando GRU y el caso base.

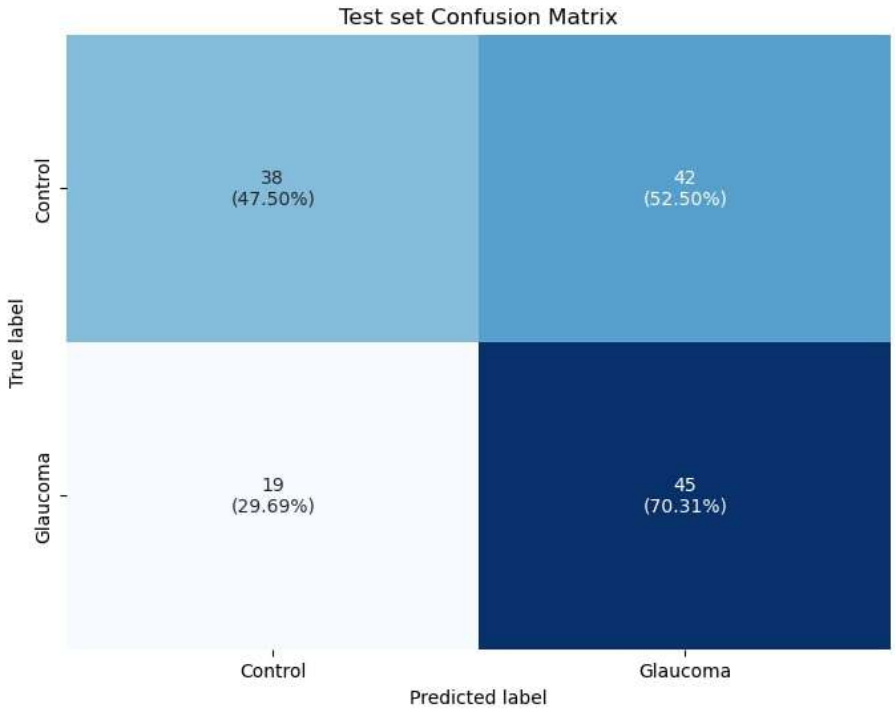

Figura C.38: Matriz de confusión del conjunto de prueba, utilizando GRU y el caso base.

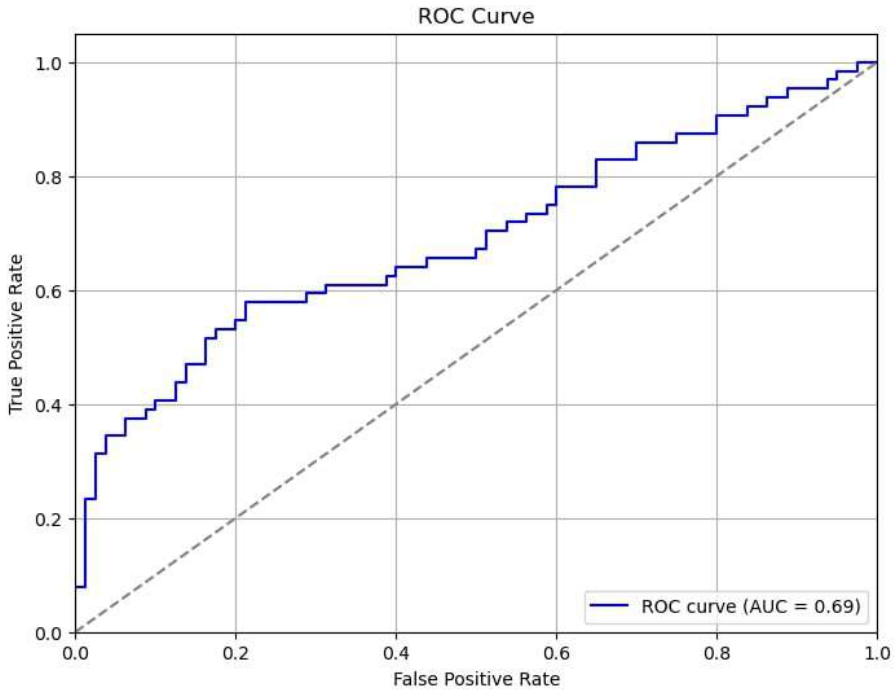

Figura C.39: Curva ROC, utilizando GRU y el caso base.

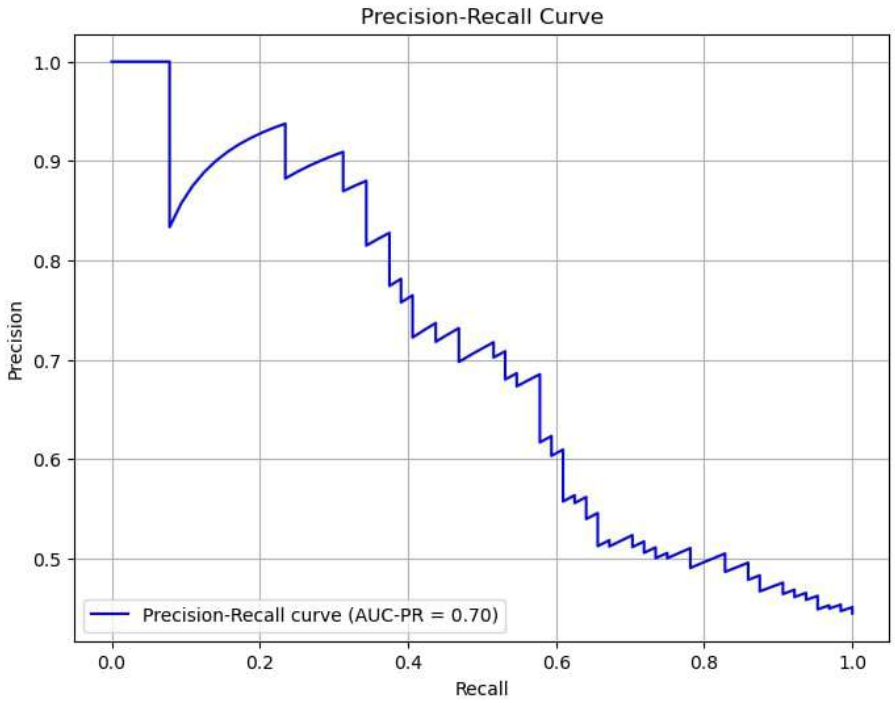

Figura C.40: Curva *precision-recall*, utilizando GRU y el caso base.

Figura C.41: Curvas de pérdida en el conjunto de entrenamiento y validación, utilizando GRU y el caso base.

Figura C.42: Curvas de error en el conjunto de entrenamiento y validación, utilizando GRU y el caso base.

C.2.2. Resultados caso filtrado

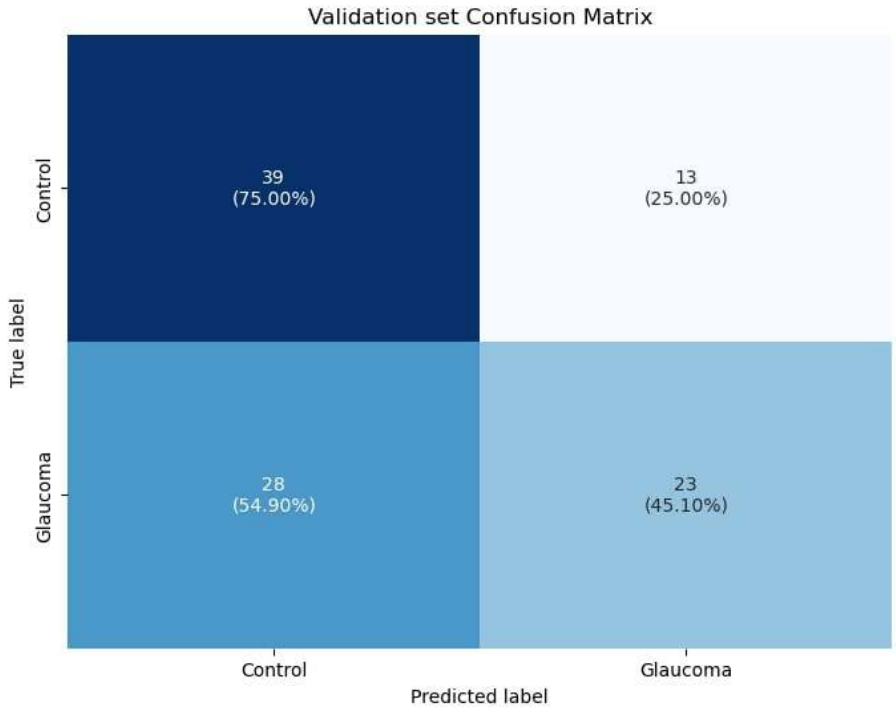

Figura C.43: Matriz de confusión del conjunto de validación, utilizando GRU y el caso filtrado.

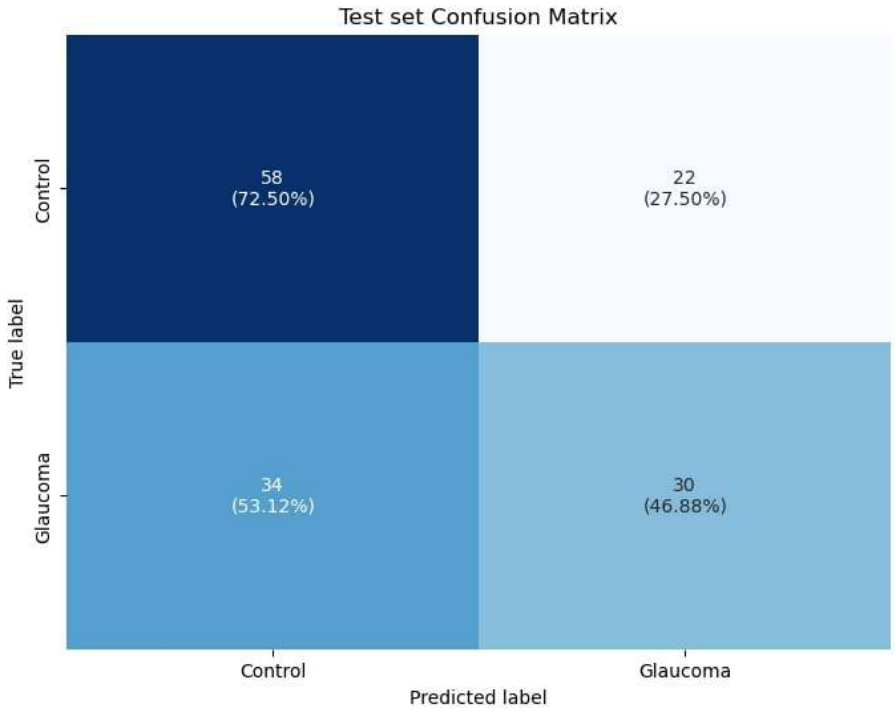

Figura C.44: Matriz de confusión del conjunto de prueba, utilizando GRU y el caso filtrado.

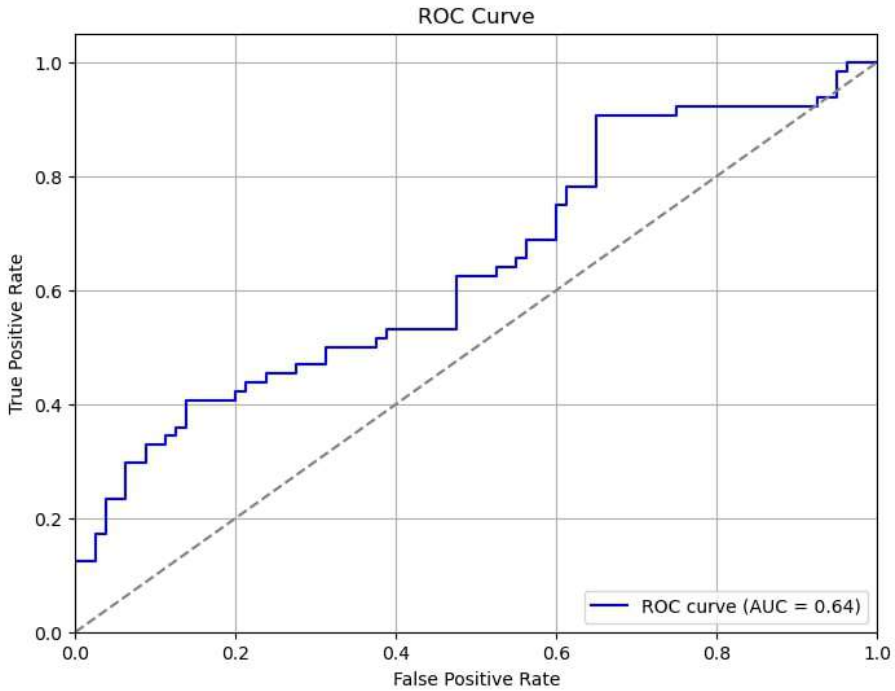

Figura C.45: Curva ROC, utilizando GRU y el caso filtrado.

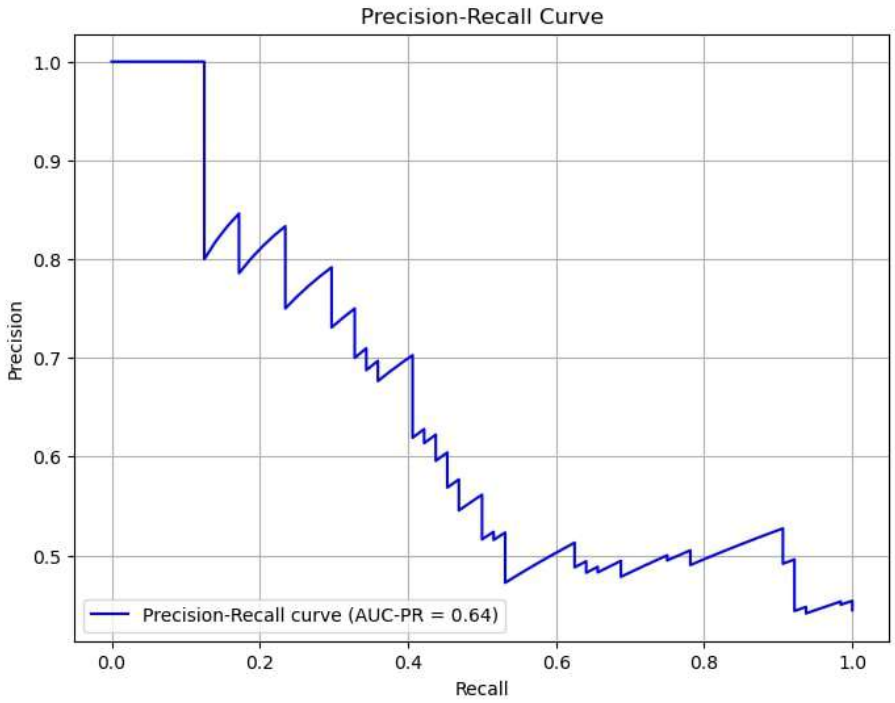

Figura C.46: Curva *precision-recall*, utilizando GRU y el caso filtrado.

Figura C.47: Curvas de pérdida en el conjunto de entrenamiento y validación, utilizando GRU y el caso filtrado.

Figura C.48: Curvas de error en el conjunto de entrenamiento y validación, utilizando GRU y el caso filtrado.

C.2.3. Resultados caso rampa

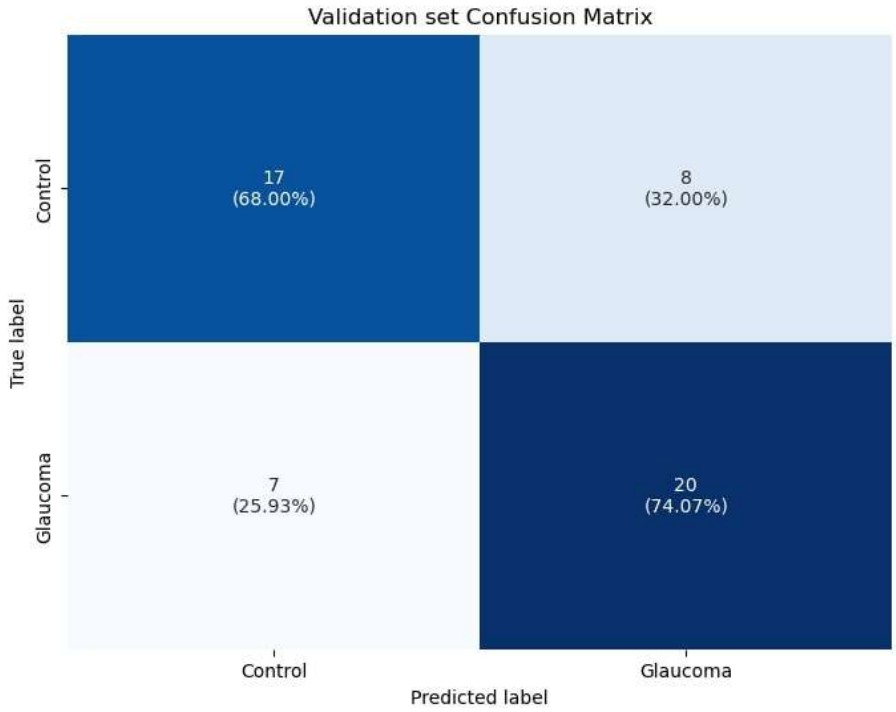

Figura C.49: Matriz de confusión del conjunto de validación, utilizando GRU y el caso rampa.

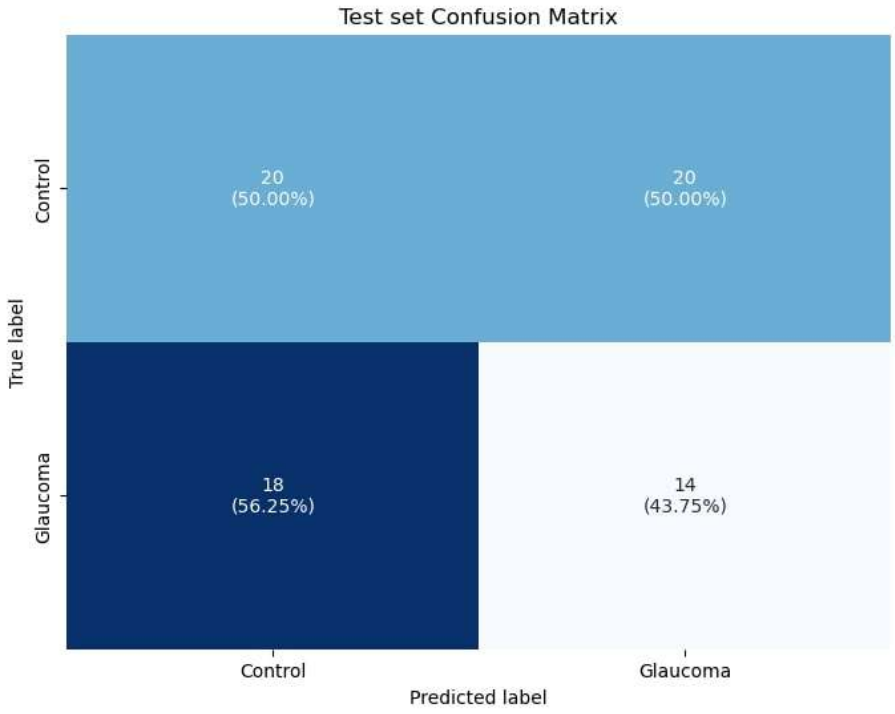

Figura C.50: Matriz de confusión del conjunto de prueba, utilizando GRU y el caso rampa.

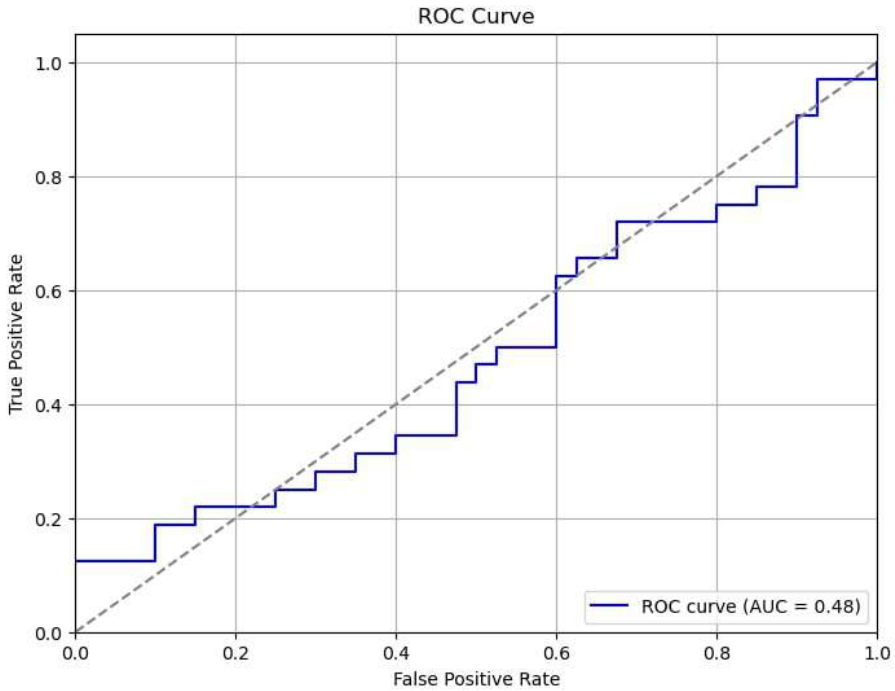

Figura C.51: Curva ROC, utilizando GRU y el caso rampa.

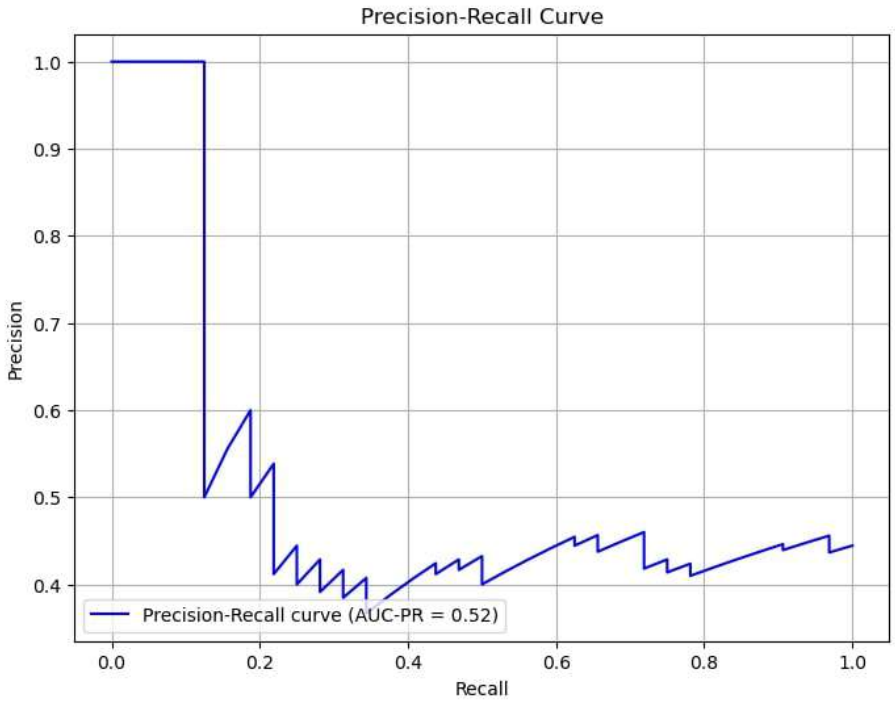

Figura C.52: Curva *precision-recall*, utilizando GRU y el caso rampa.

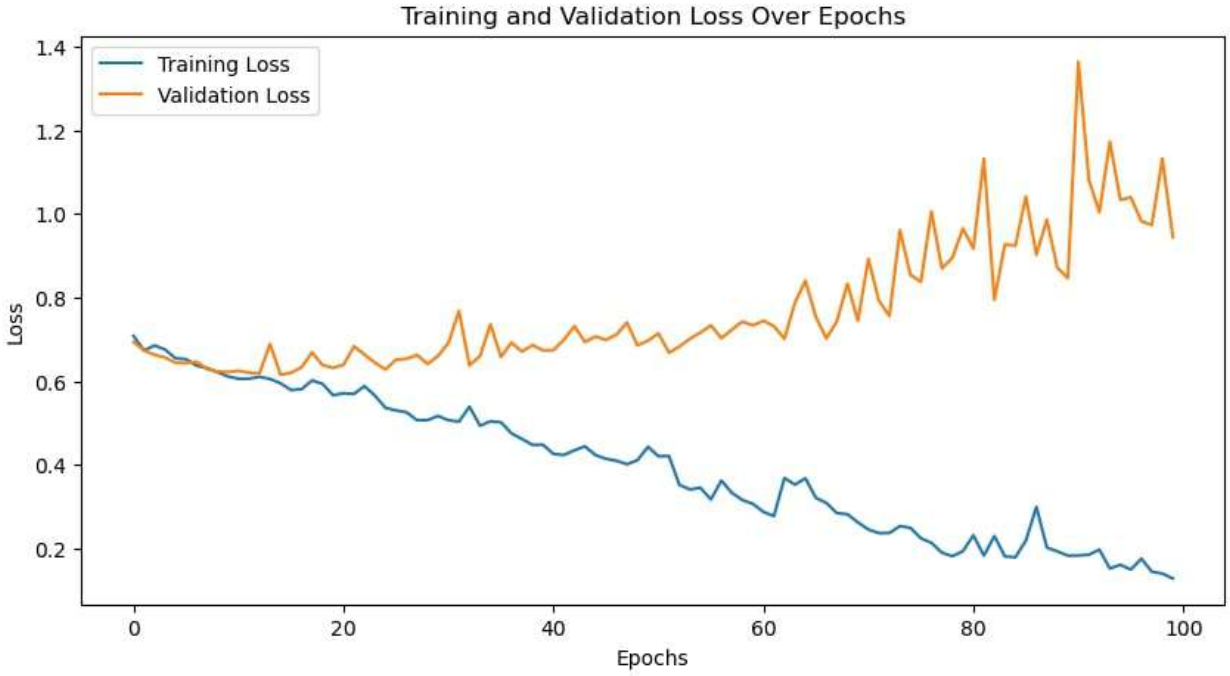

Figura C.53: Curvas de pérdida en el conjunto de entrenamiento y validación, utilizando GRU y el caso rampa.

Figura C.54: Curvas de error en el conjunto de entrenamiento y validación, utilizando GRU y el caso rampa.

C.2.4. Resultados caso *flash*

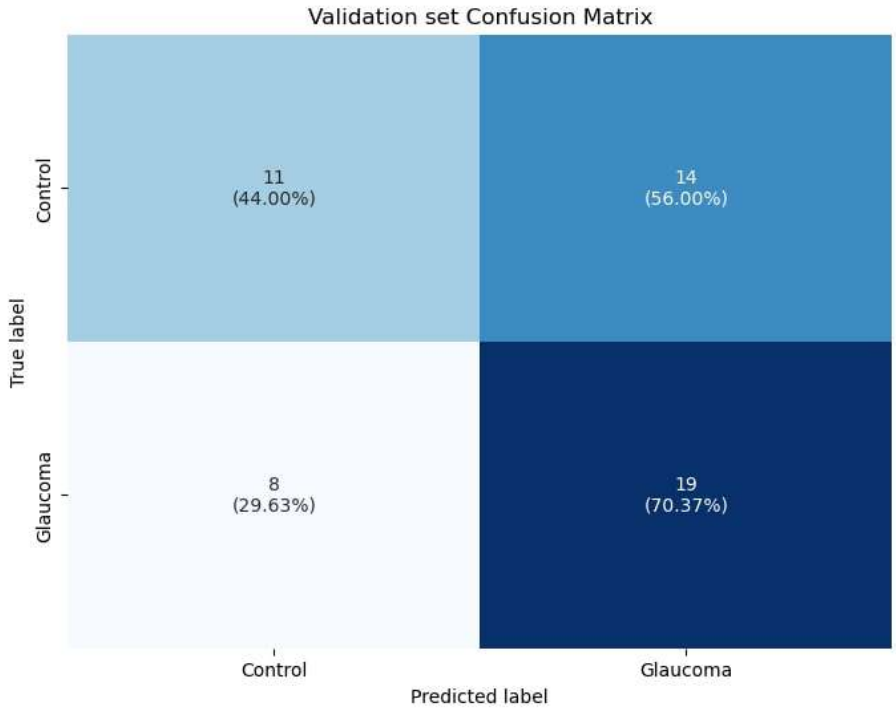

Figura C.55: Matriz de confusión del conjunto de validación, utilizando GRU y el caso *flash*.

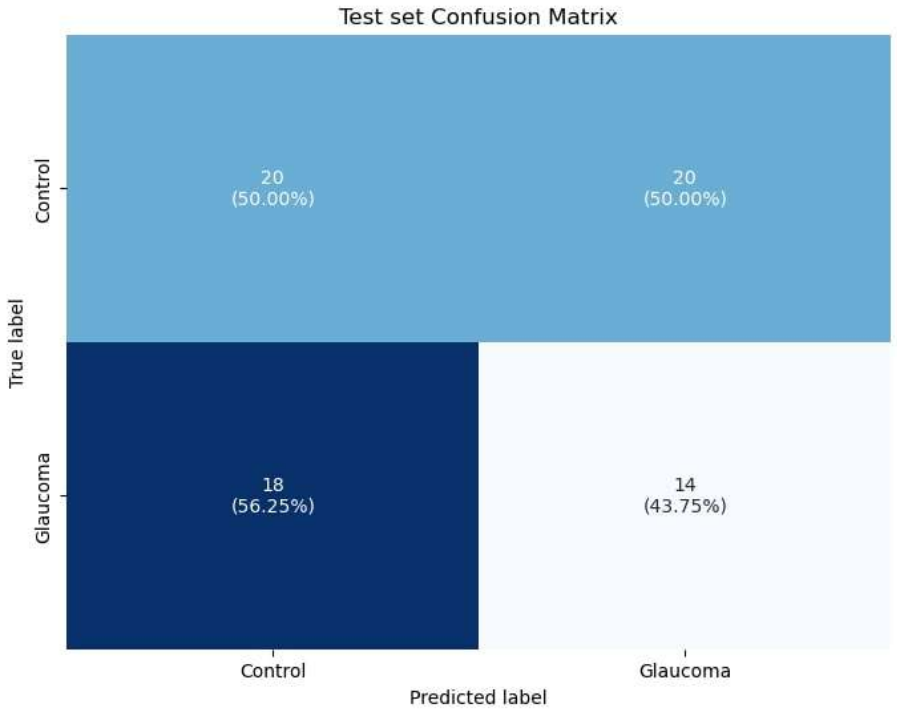

Figura C.56: Matriz de confusión del conjunto de prueba, utilizando GRU y el caso *flash*.

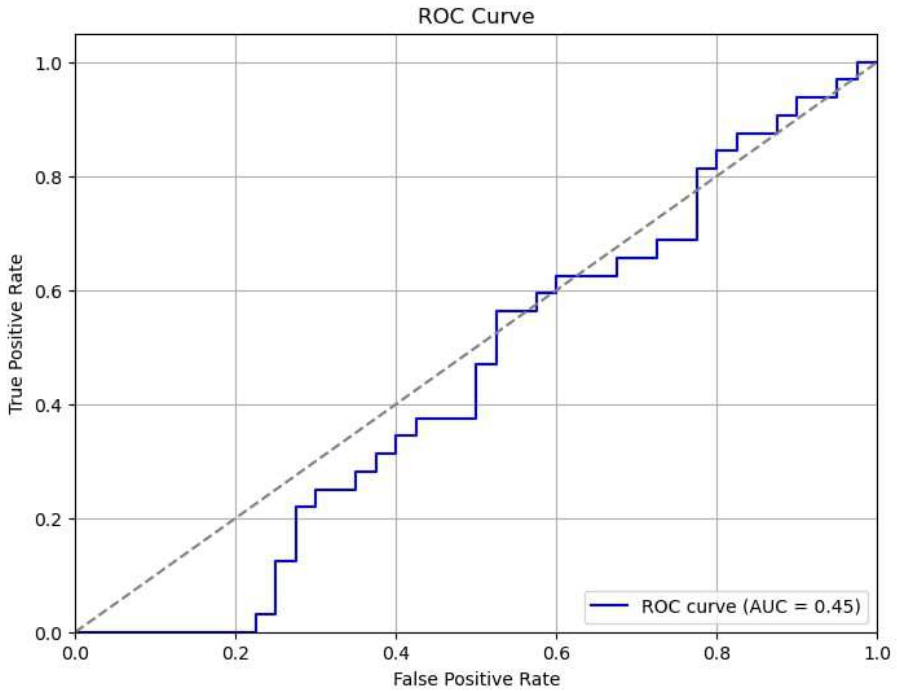

Figura C.57: Curva ROC, utilizando GRU y el caso *flash*.

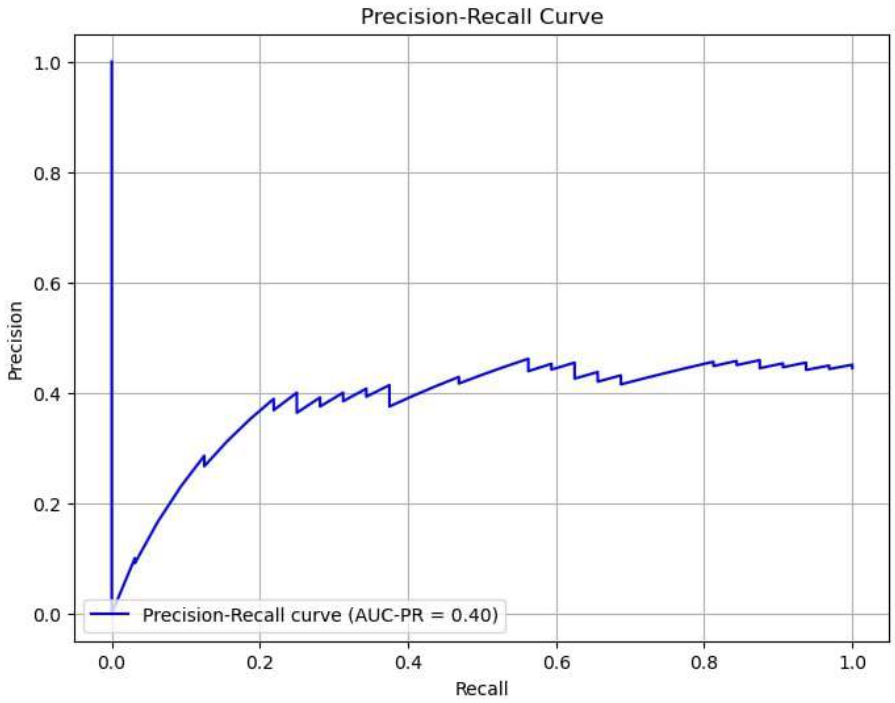

Figura C.58: Curva *precision-recall*, utilizando GRU y el caso *flash*.

Figura C.59: Curvas de pérdida en el conjunto de entrenamiento y validación, utilizando GRU y el caso *flash*.

Figura C.60: Curvas de error en el conjunto de entrenamiento y validación, utilizando GRU y el caso *flash*.

C.2.5. Resultados caso rampa filtrado

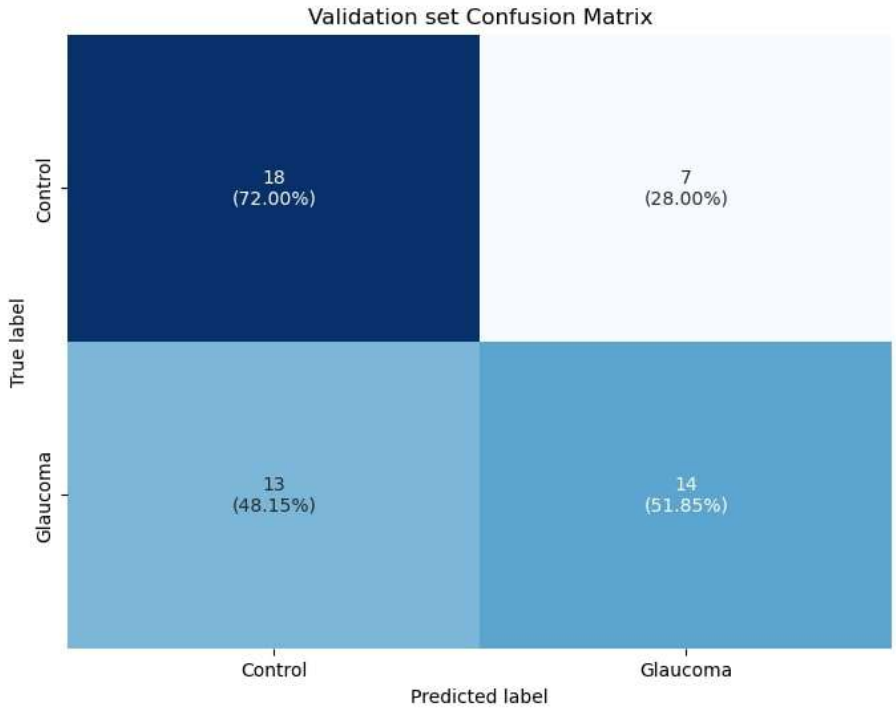

Figura C.61: Matriz de confusión del conjunto de validación, utilizando GRU y el caso rampa filtrado.

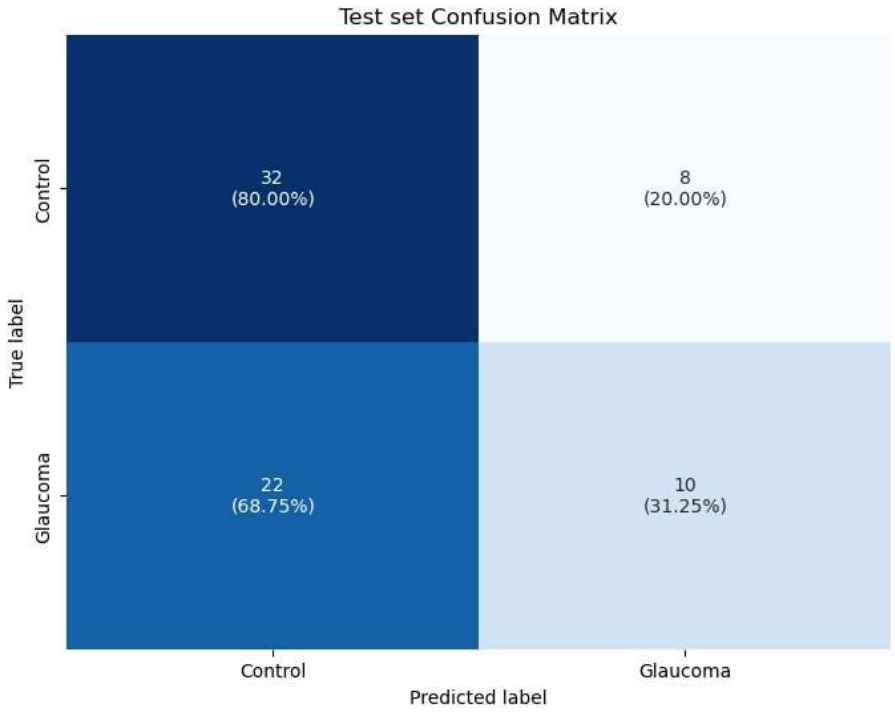

Figura C.62: Matriz de confusión del conjunto de prueba, utilizando GRU y el caso rampa filtrado.

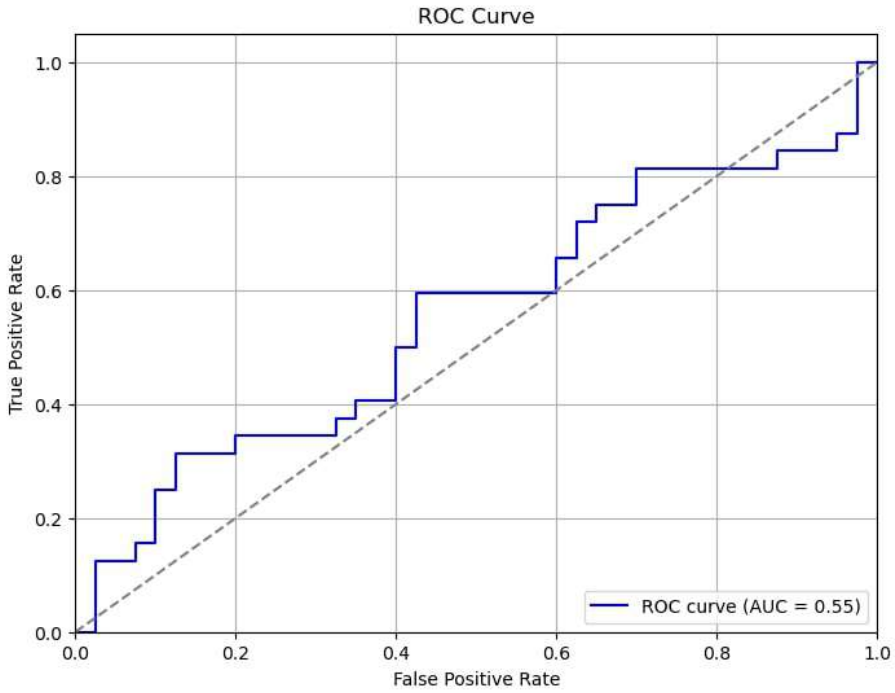

Figura C.63: Curva ROC, utilizando GRU y el caso rampa filtrado.

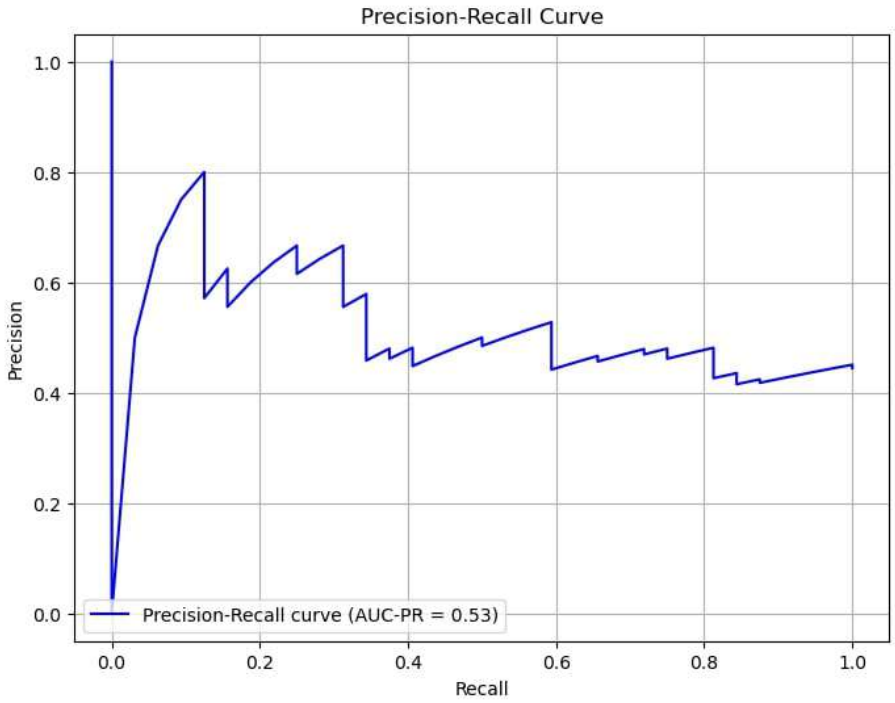

Figura C.64: Curva *precision-recall*, utilizando GRU y el caso rampa filtrado.

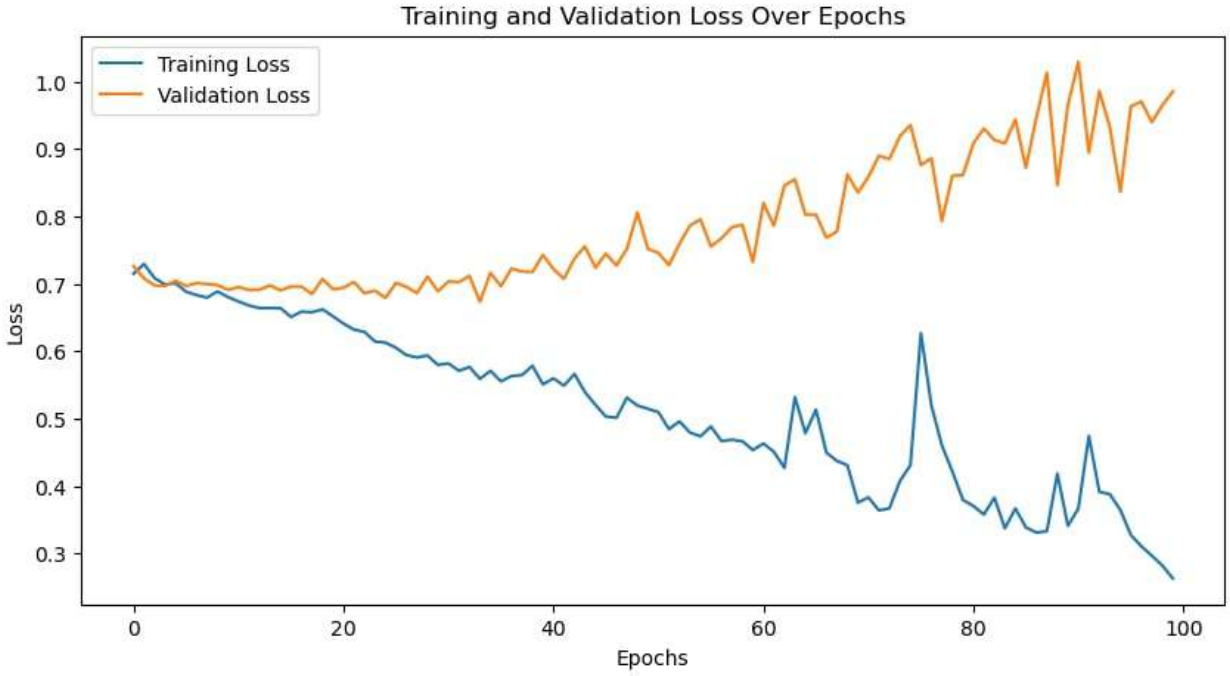

Figura C.65: Curvas de pérdida en el conjunto de entrenamiento y validación, utilizando GRU y el caso rampa filtrado.

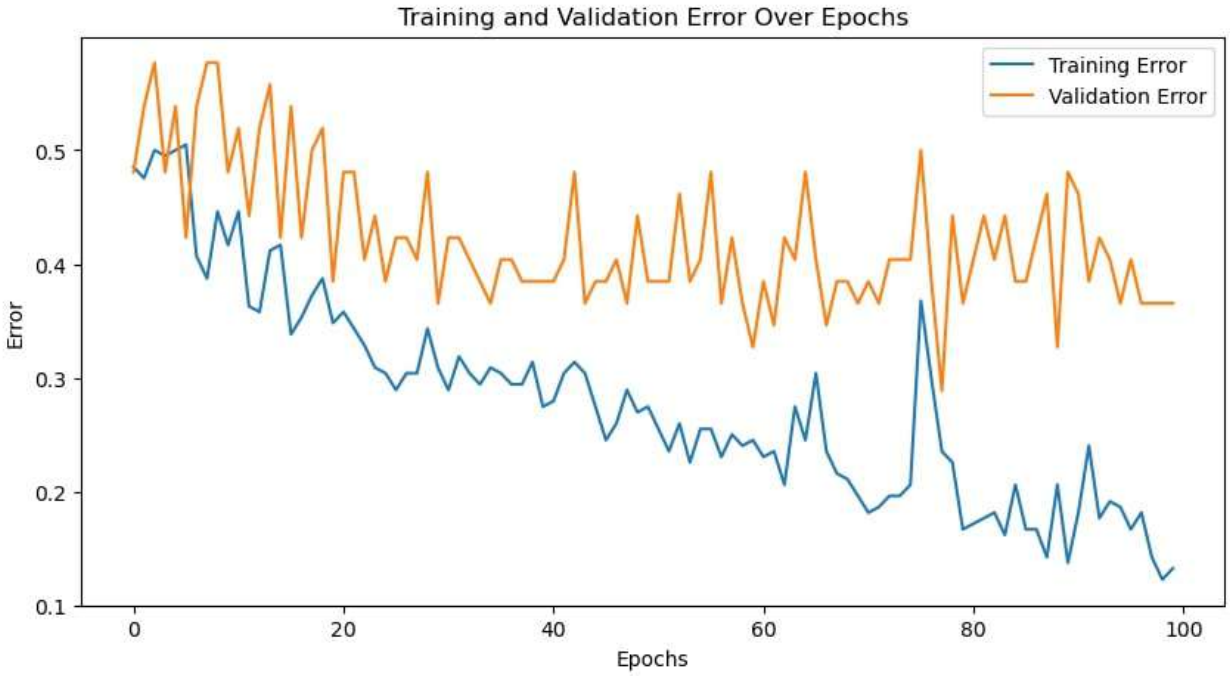

Figura C.66: Curvas de error en el conjunto de entrenamiento y validación, utilizando GRU y el caso rampa filtrado.

C.2.6. Resultados caso *flash* filtrado

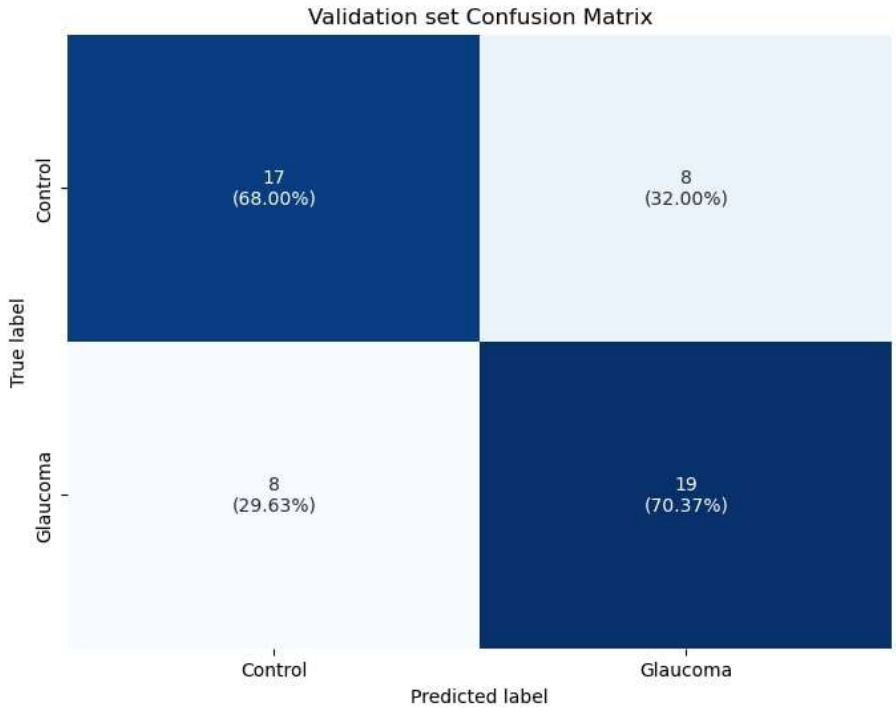

Figura C.67: Matriz de confusión del conjunto de validación, utilizando GRU y el caso *flash* filtrado.

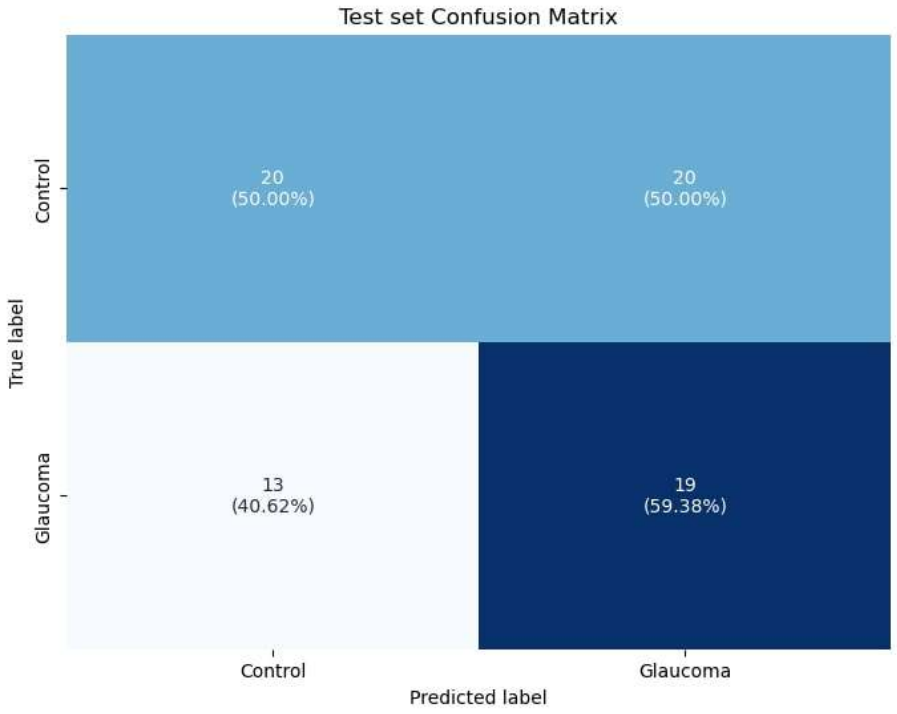

Figura C.68: Matriz de confusión del conjunto de prueba, utilizando GRU y el caso *flash* filtrado.

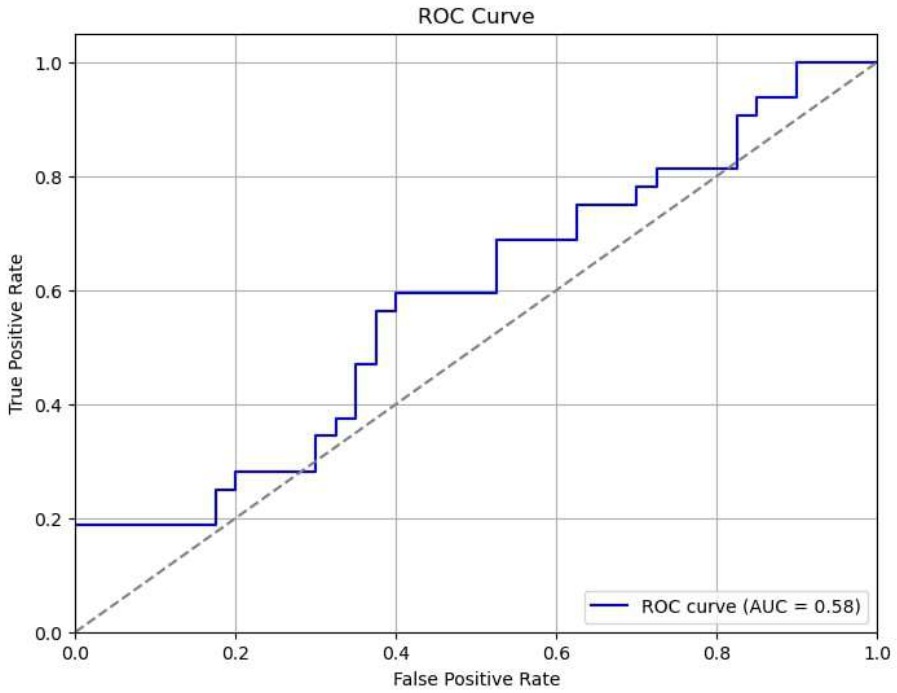

Figura C.69: Curva ROC, utilizando GRU y el caso *flash* filtrado.

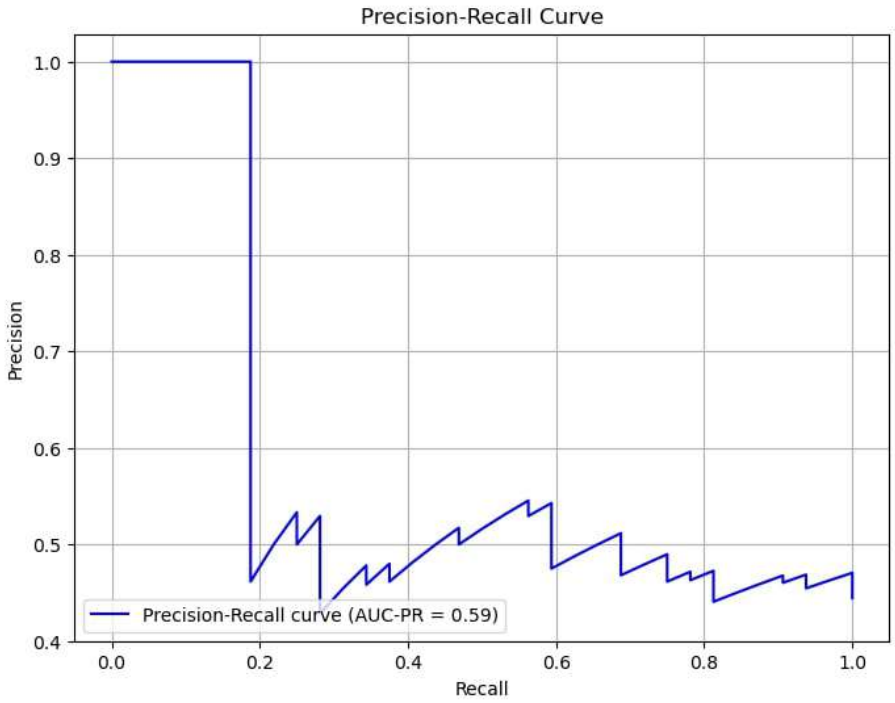

Figura C.70: Curva *precision-recall*, utilizando GRU y el caso *flash* filtrado.

Figura C.71: Curvas de pérdida en el conjunto de entrenamiento y validación, utilizando GRU y el caso *flash* filtrado.

Figura C.72: Curvas de error en el conjunto de entrenamiento y validación, utilizando GRU y el caso *flash* filtrado.

Anexo D. Matrices de confusión

D.1. Matrices de confusión caso base

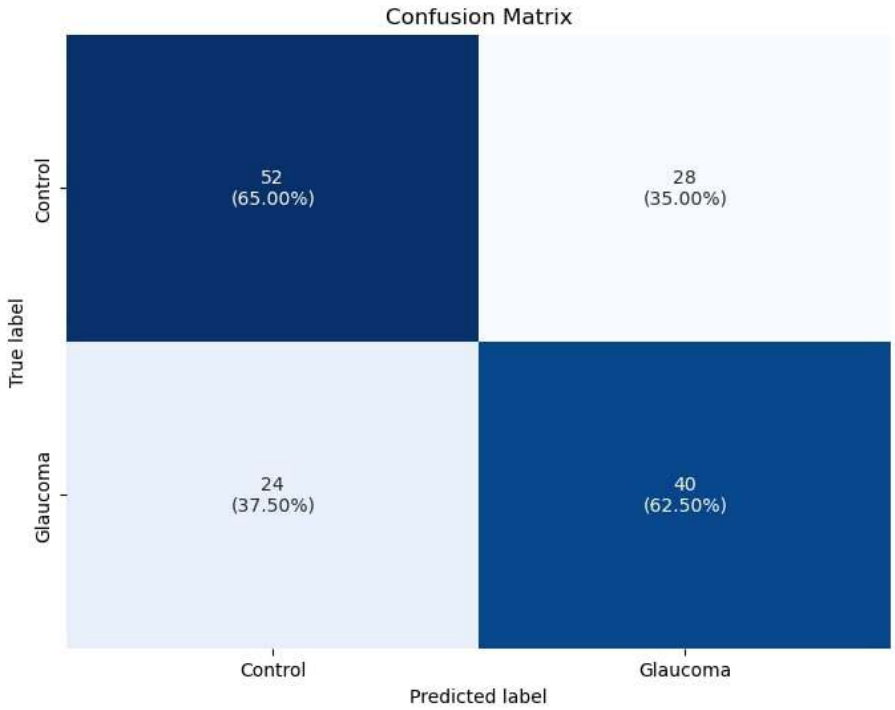

Figura D.1: Matriz de confusión caso base *Normal*, con ventanas de 50 muestras de largo.

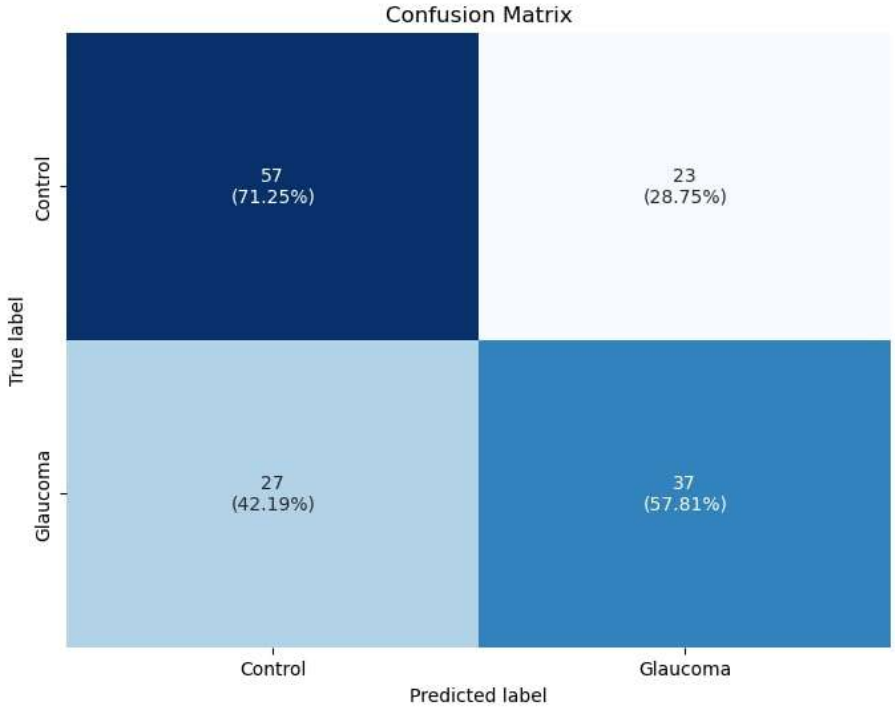

Figura D.2: Matriz de confusión caso base *Normal*, con ventanas de 100 muestras de largo.

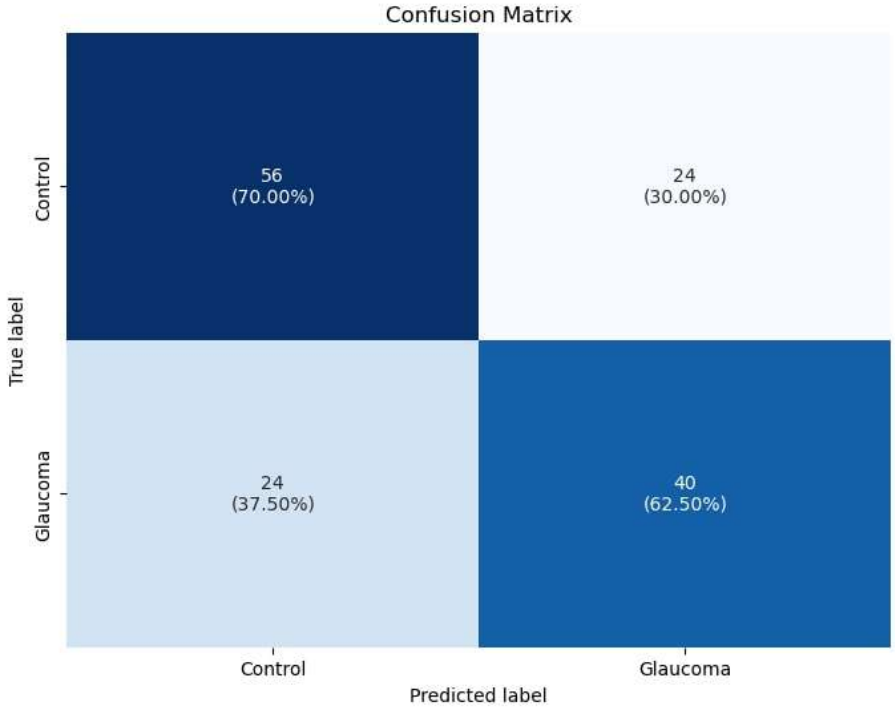

Figura D.3: Matriz de confusión caso base *Normal*, con ventanas de 25 muestras de largo.

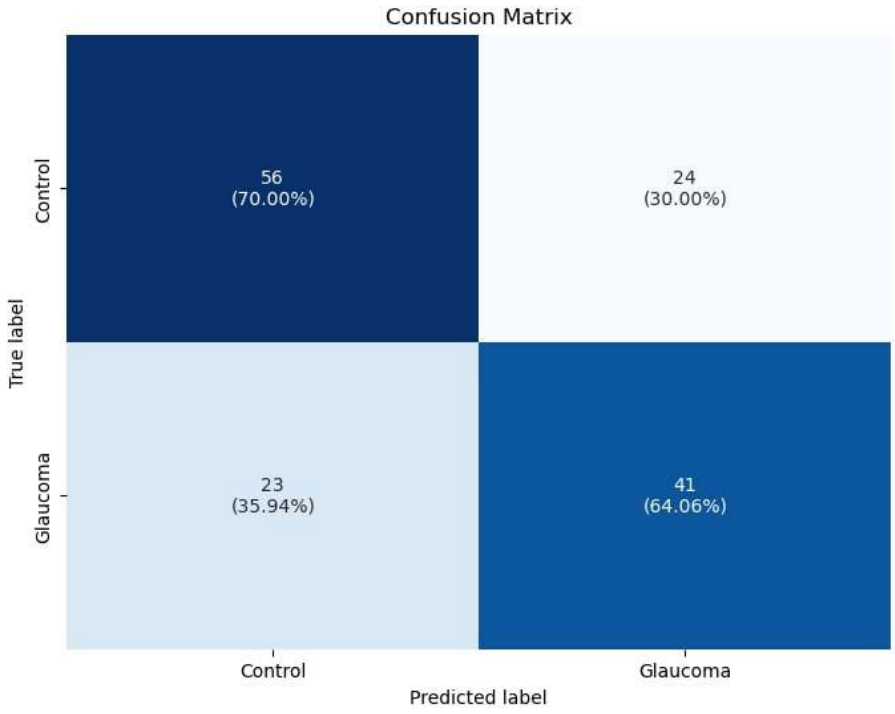

Figura D.4: Matriz de confusión caso base *Normal*, con ventanas de 50 y 100 muestras de largo.

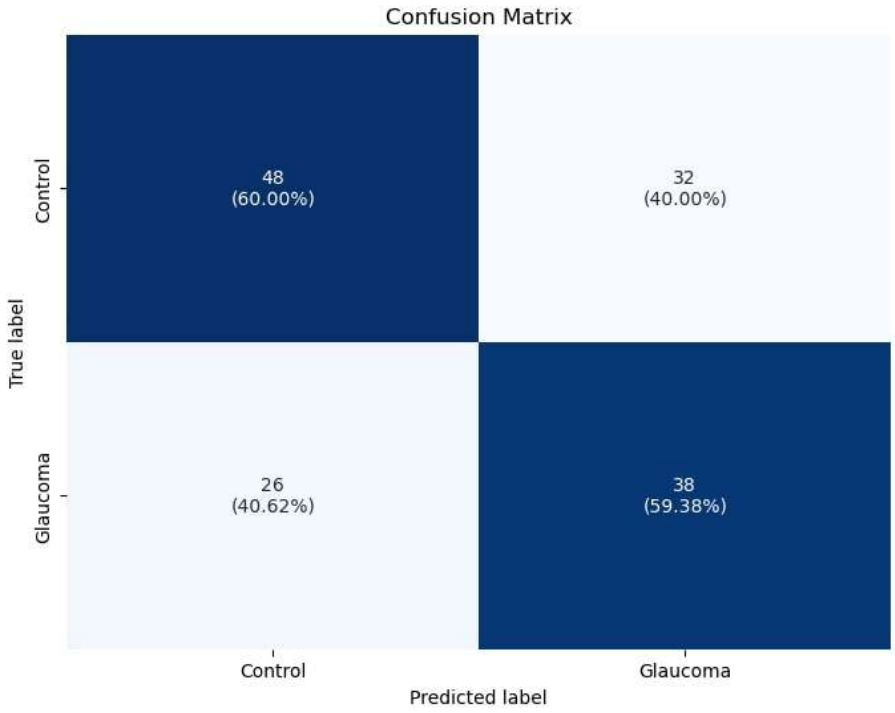

Figura D.5: Matriz de confusión caso base *Filtrado*, con ventanas de 50 muestras de largo.

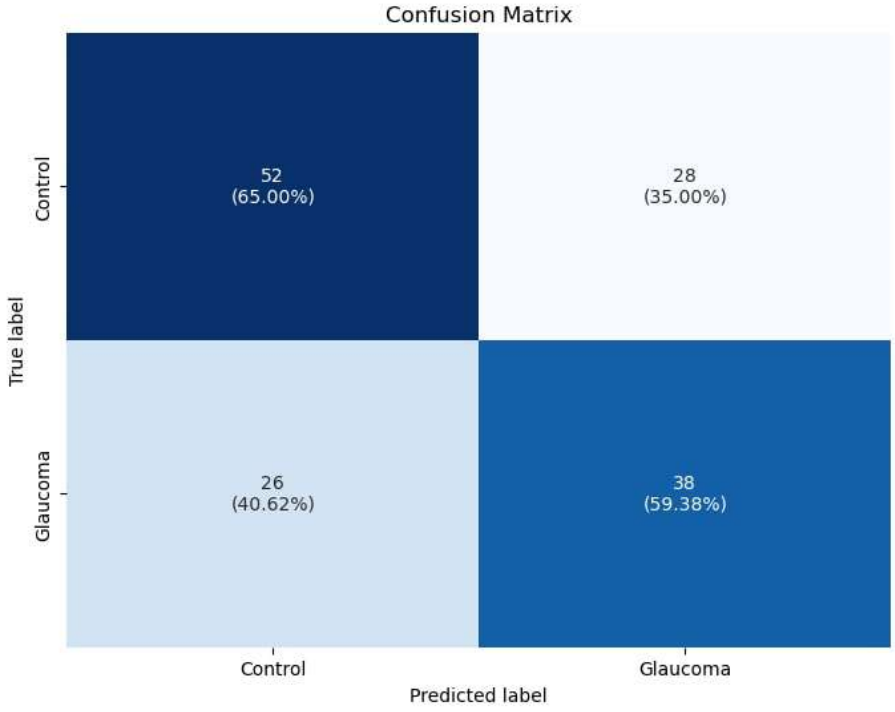

Figura D.6: Matriz de confusión caso base *Filtrado*, con ventanas de 100 muestras de largo.

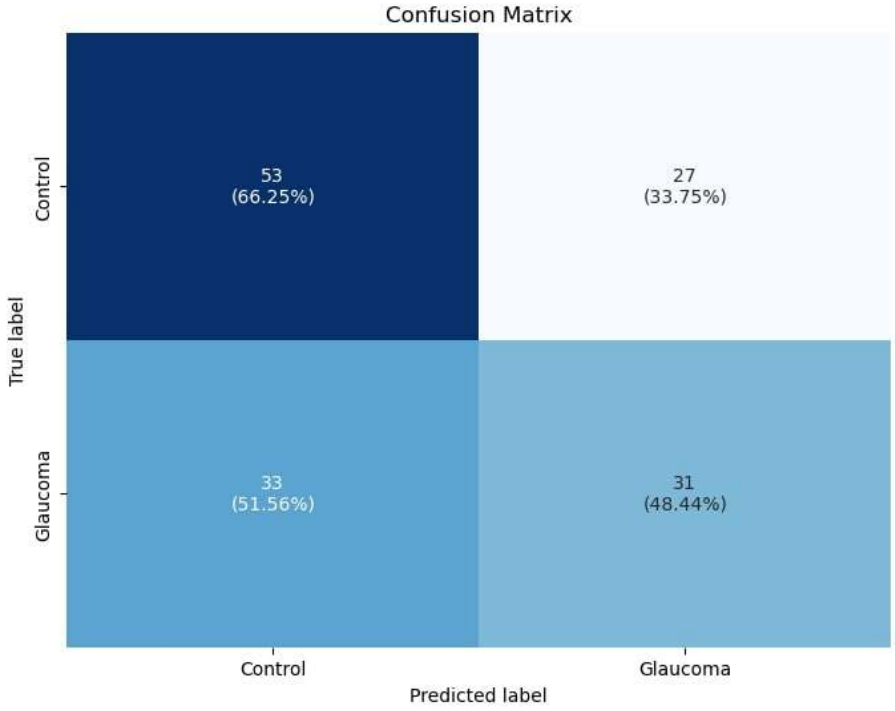

Figura D.7: Matriz de confusión caso base *Filtrado*, con ventanas de 25 muestras de largo.

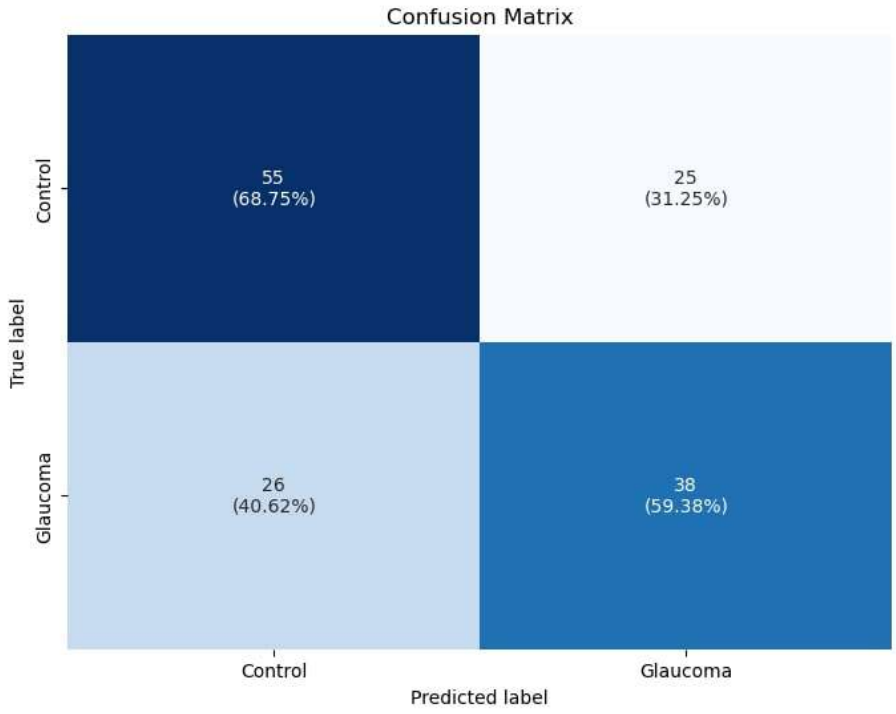

Figura D.8: Matriz de confusión caso base *Filtrado*, con ventanas de 50 y 100 muestras de largo.

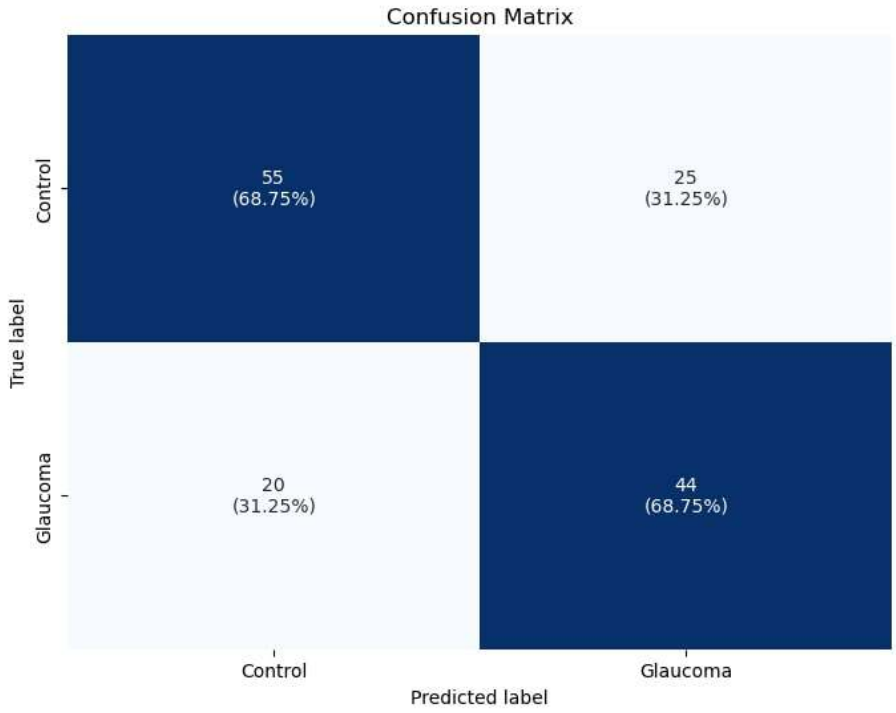

Figura D.9: Matriz de confusión caso base *D.A.*, con ventanas de 50 muestras de largo.

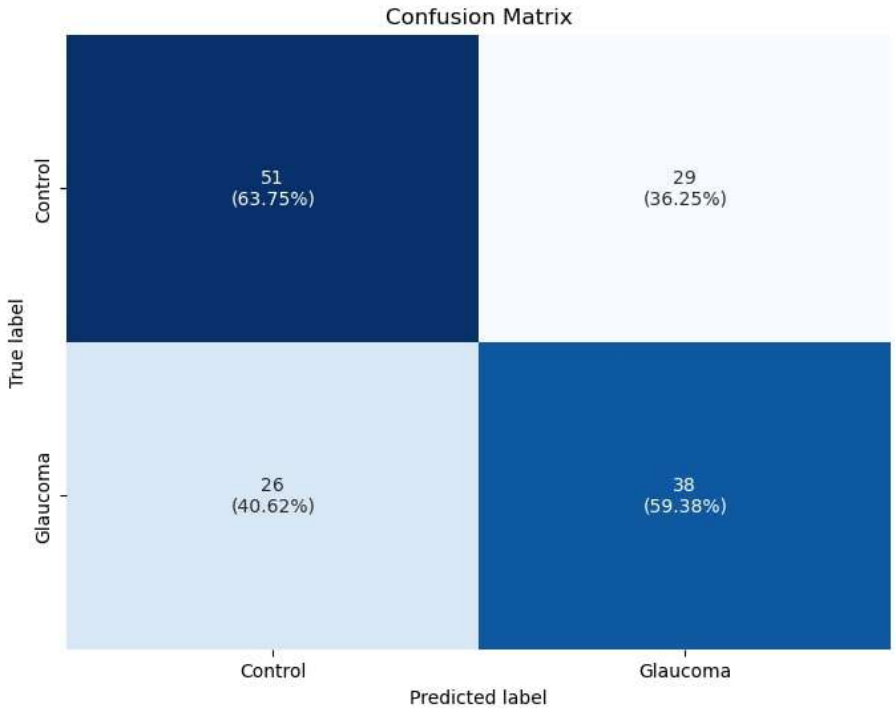

Figura D.10: Matriz de confusión caso base *D.A.*, con ventanas de 100 muestras de largo.

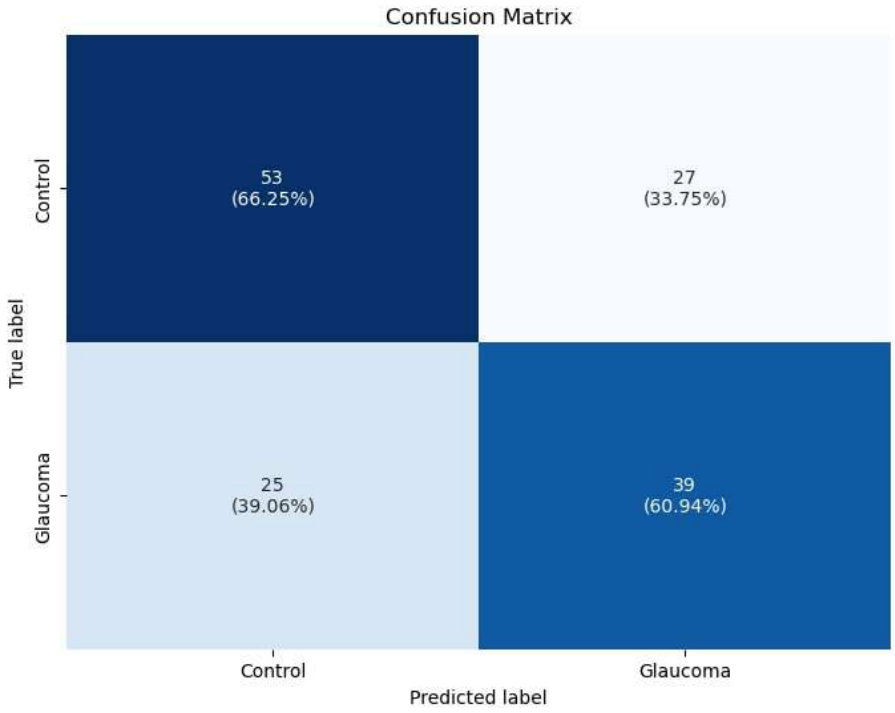

Figura D.11: Matriz de confusión caso base *D.A.*, con ventanas de 25 muestras de largo.

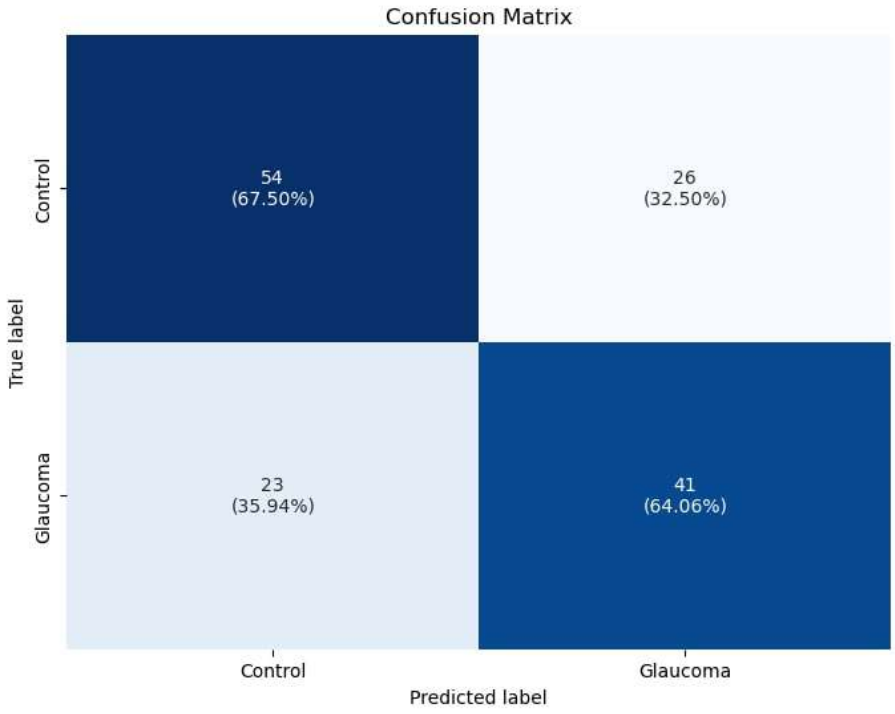

Figura D.12: Matriz de confusión caso base *D.A.*, con ventanas de 50 y 100 muestras de largo.

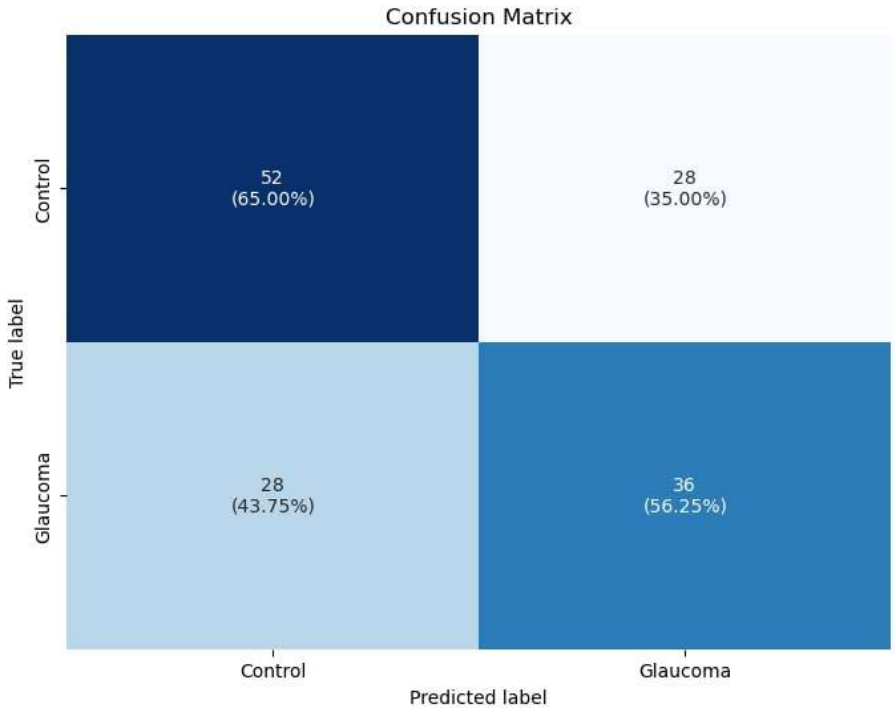

Figura D.13: Matriz de confusión caso base *Filtrado + D.A.*, con ventanas de 50 muestras de largo.

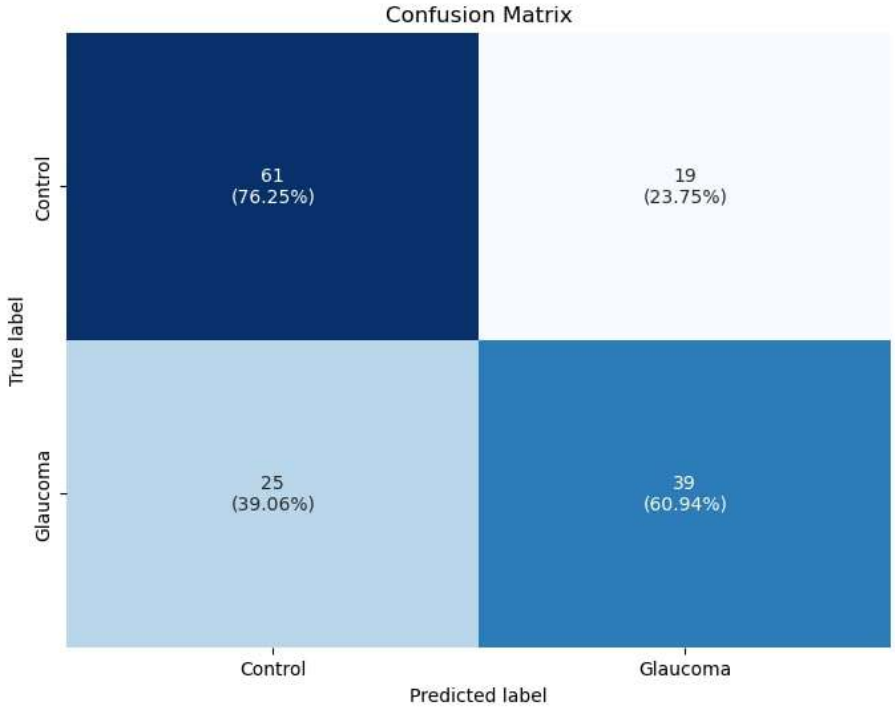

Figura D.14: Matriz de confusión caso base *Filtrado + D.A.*, con ventanas de 100 muestras de largo.

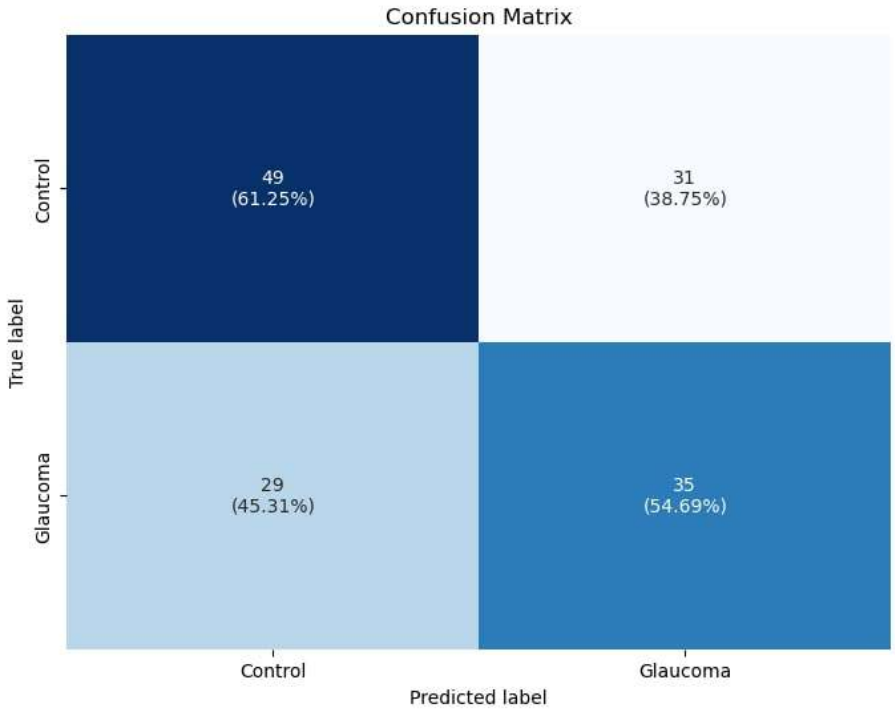

Figura D.15: Matriz de confusión caso base *Filtrado + D.A.*, con ventanas de 25 muestras de largo.

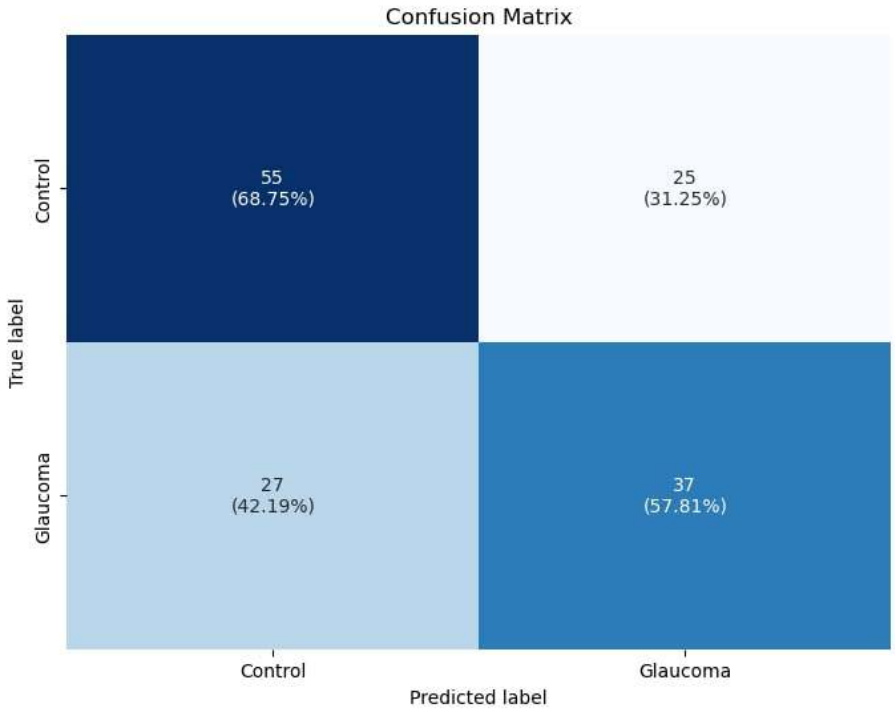

Figura D.16: Matriz de confusión caso base *Filtrado + D.A.*, con ventanas de 50 y 100 muestras de largo.

D.2. Matrices de confusión caso selección de ventanas

D.2.1. Primeros 20 segundos

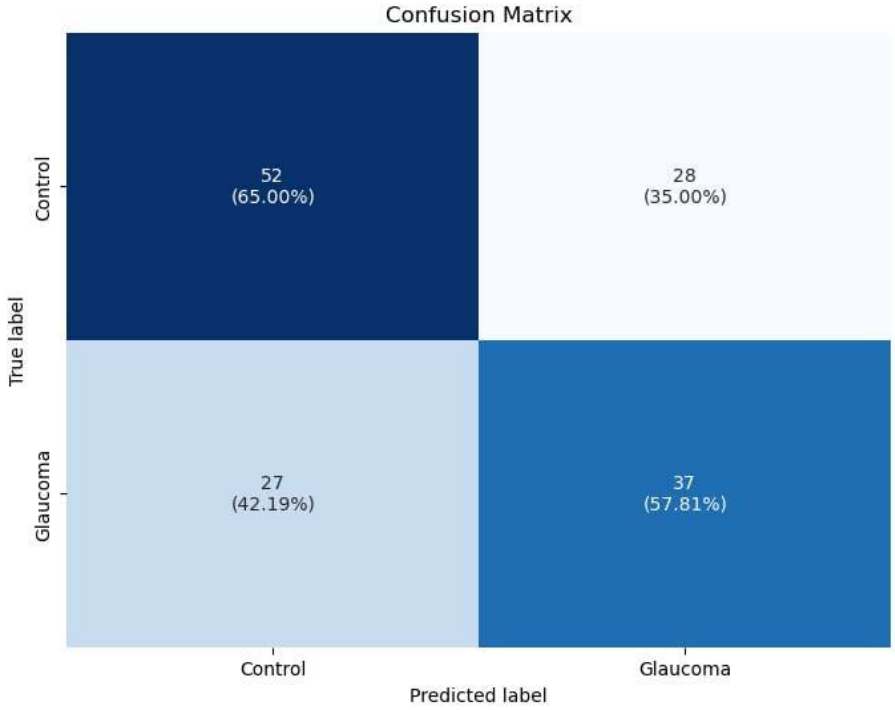

Figura D.17: Matriz de confusión caso selección de primeros 20 segundos *Normal*, con ventanas de 50 muestras de largo.

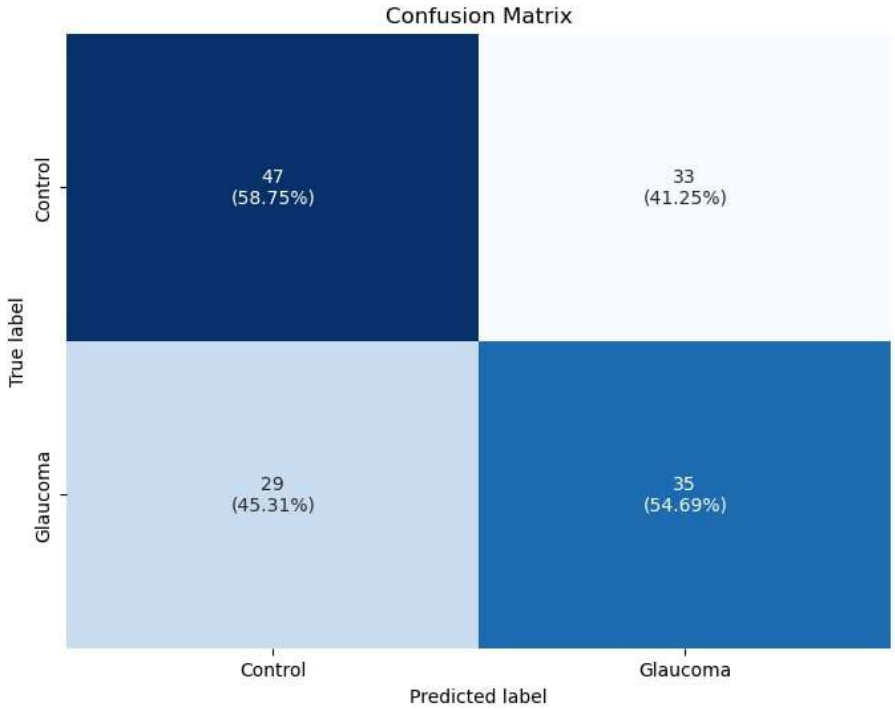

Figura D.18: Matriz de confusión caso selección de primeros 20 segundos *Normal*, con ventanas de 100 muestras de largo.

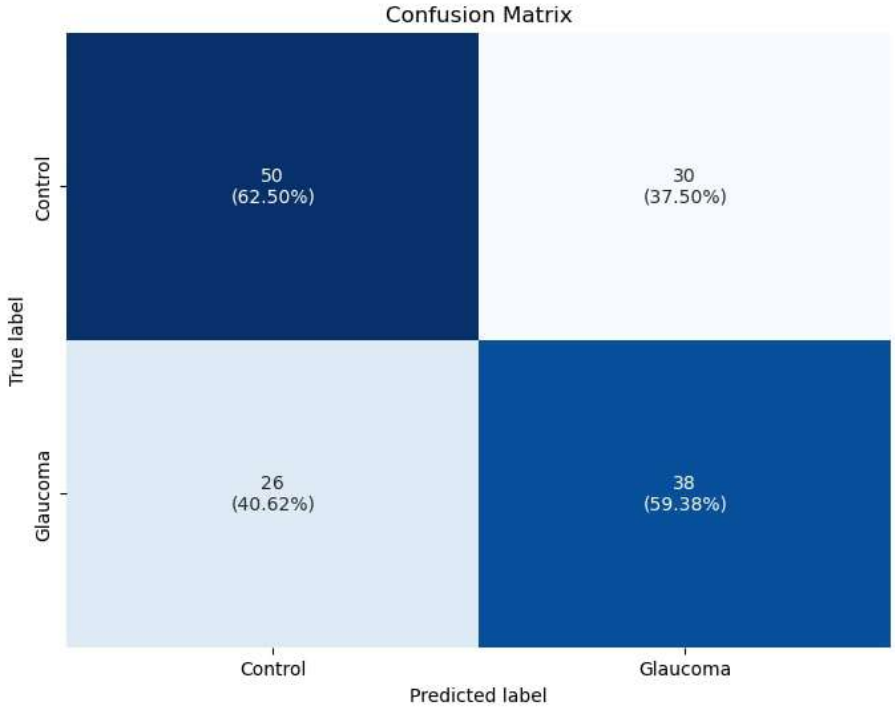

Figura D.19: Matriz de confusión caso selección de primeros 20 segundos *Normal*, con ventanas de 25 muestras de largo.

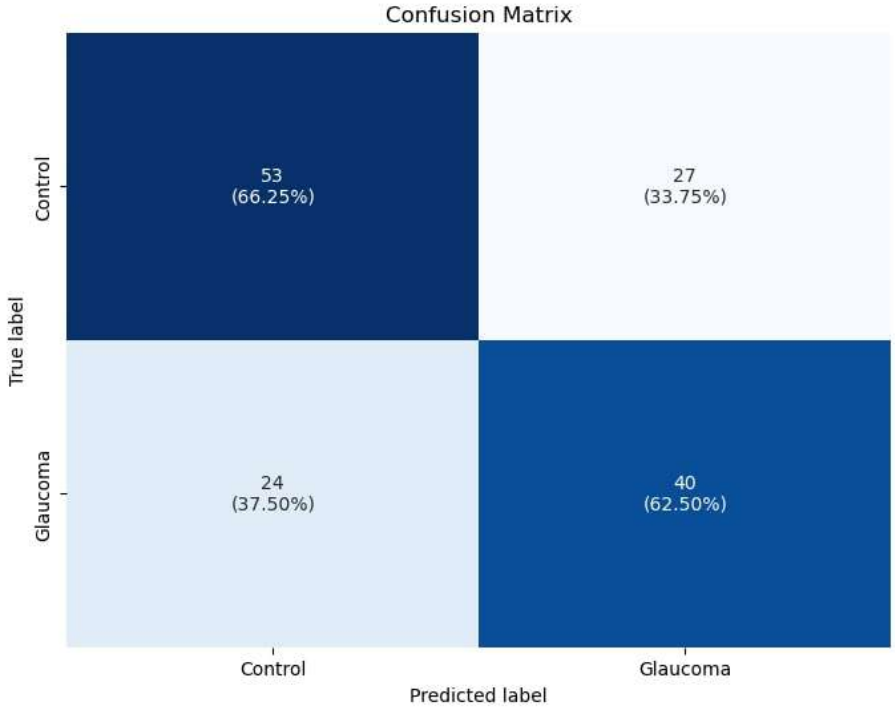

Figura D.20: Matriz de confusión caso selección de primeros 20 segundos *Normal*, con ventanas de 50 y 100 muestras de largo.

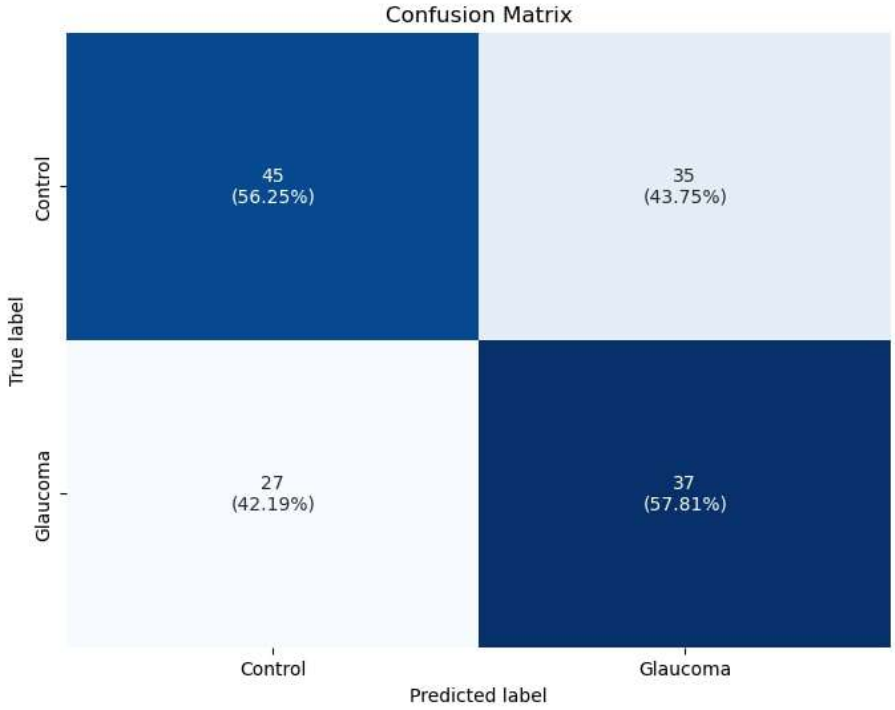

Figura D.21: Matriz de confusión caso selección de primeros 20 segundos *Filtrado*, con ventanas de 50 muestras de largo.

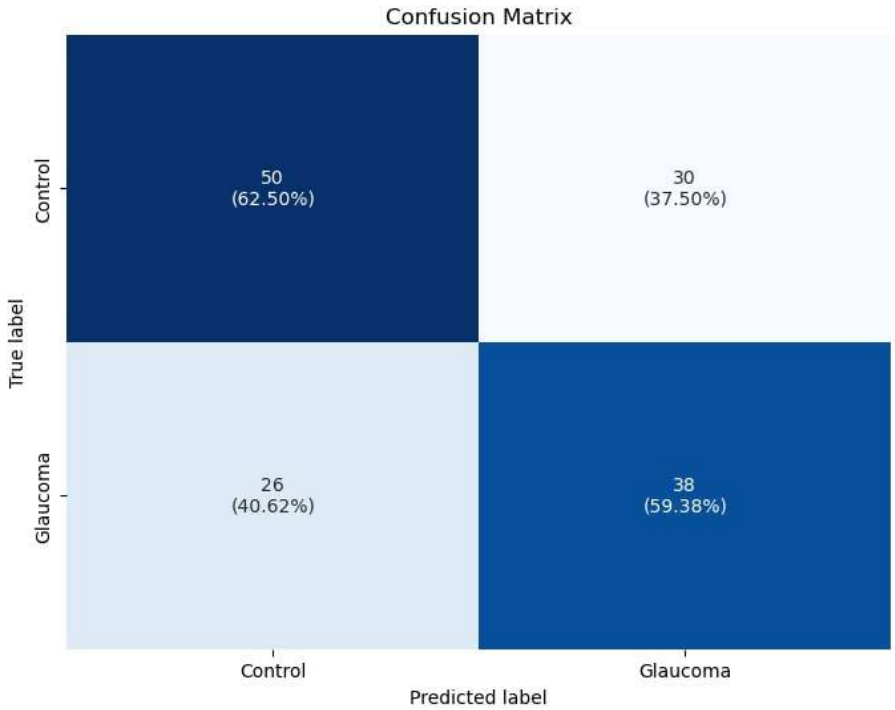

Figura D.22: Matriz de confusión caso selección de primeros 20 segundos *Filtrado*, con ventanas de 100 muestras de largo.

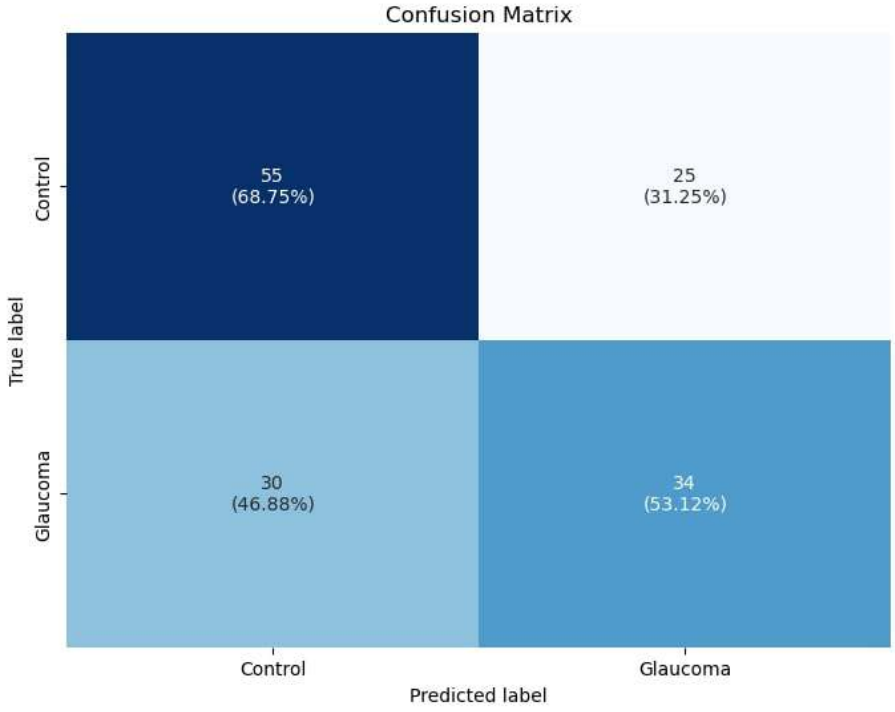

Figura D.23: Matriz de confusión caso selección de primeros 20 segundos *Filtrado*, con ventanas de 25 muestras de largo.

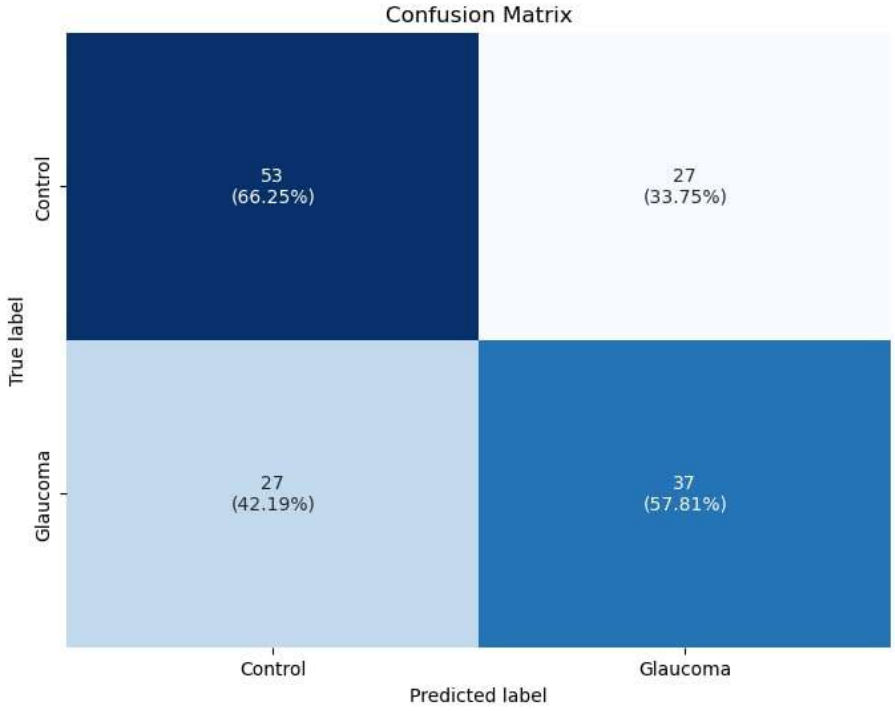

Figura D.24: Matriz de confusión caso selección de primeros 20 segundos *Filtrado*, con ventanas de 50 y 100 muestras de largo.

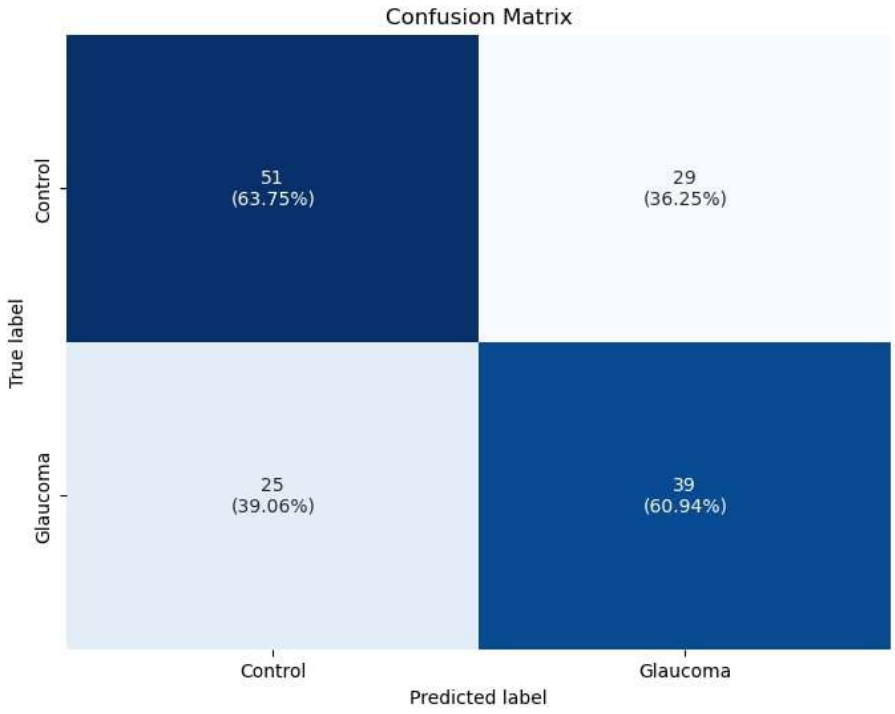

Figura D.25: Matriz de confusión caso selección de primeros 20 segundos *D.A.*, con ventanas de 50 muestras de largo.

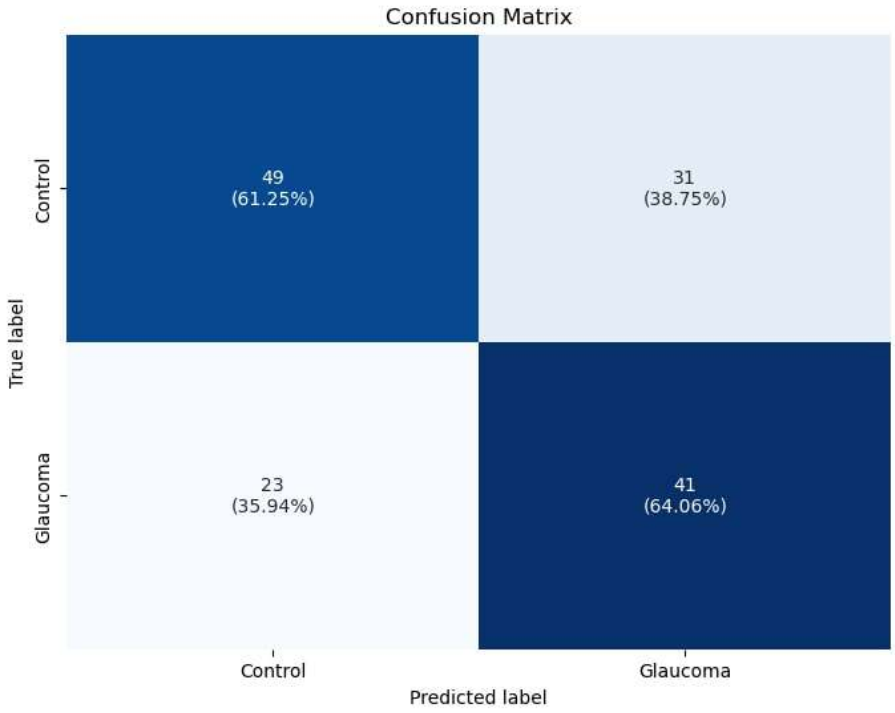

Figura D.26: Matriz de confusión caso selección de primeros 20 segundos *D.A.*, con ventanas de 100 muestras de largo.

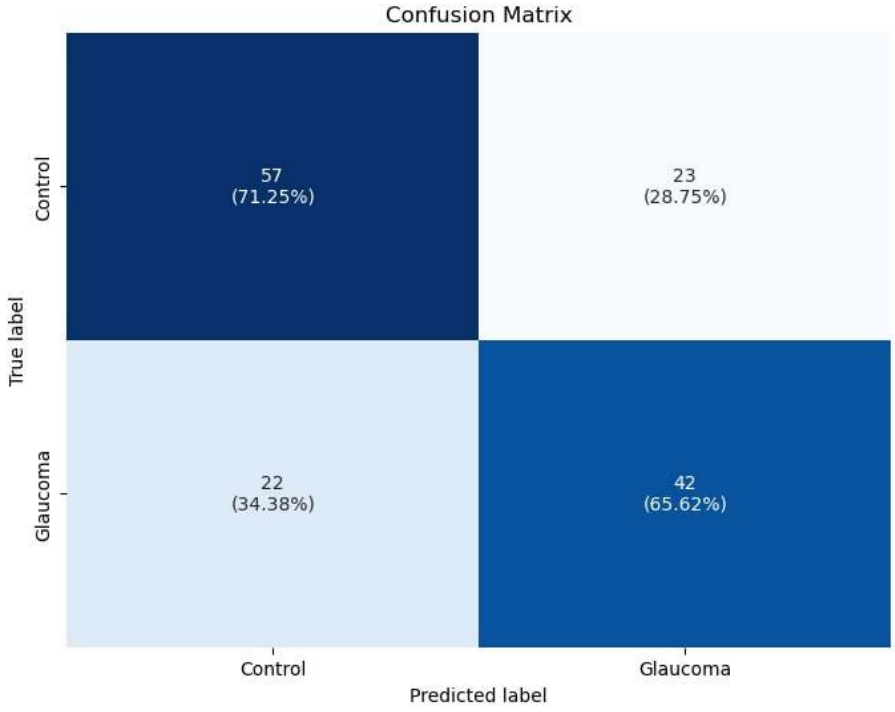

Figura D.27: Matriz de confusión caso selección de primeros 20 segundos *D.A.*, con ventanas de 25 muestras de largo.

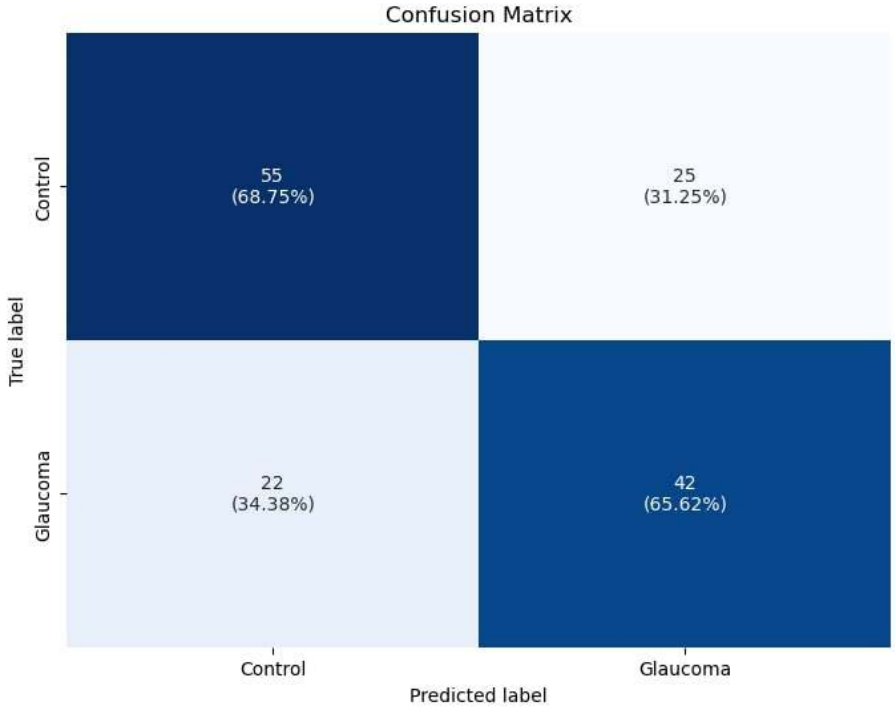

Figura D.28: Matriz de confusión caso selección de primeros 20 segundos *D.A.*, con ventanas de 50 y 100 muestras de largo.

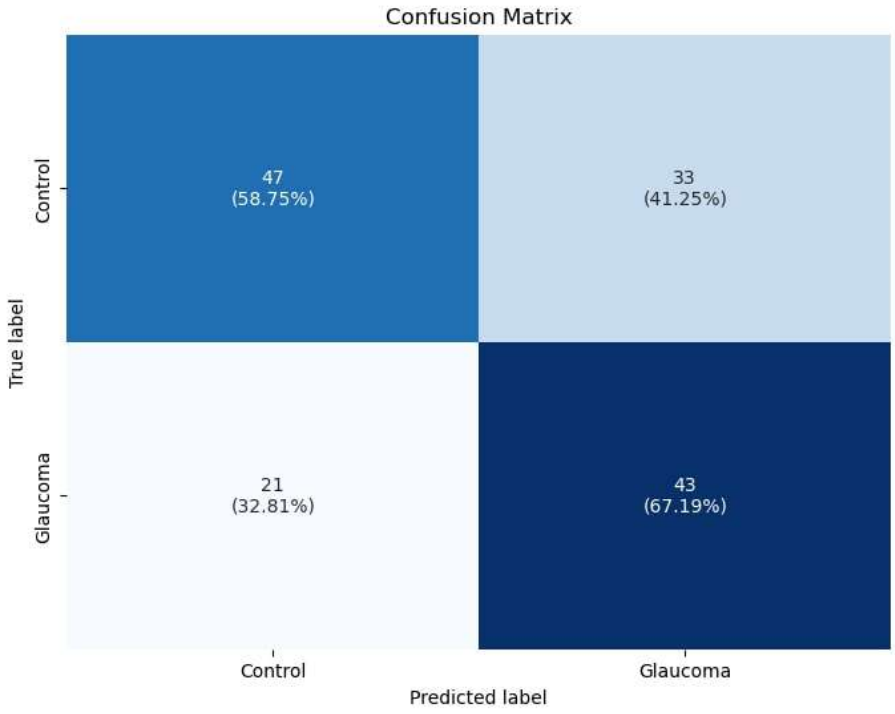

Figura D.29: Matriz de confusión caso selección de primeros 20 segundos *Filtrado + D.A.*, con ventanas de 50 muestras de largo.

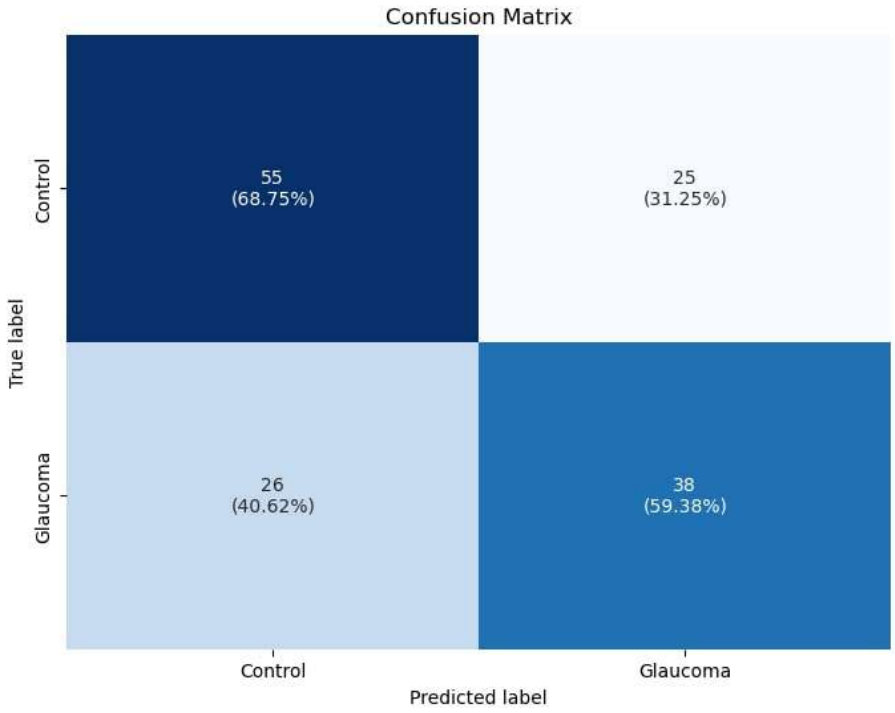

Figura D.30: Matriz de confusión caso selección de primeros 20 segundos *Filtrado + D.A.*, con ventanas de 100 muestras de largo.

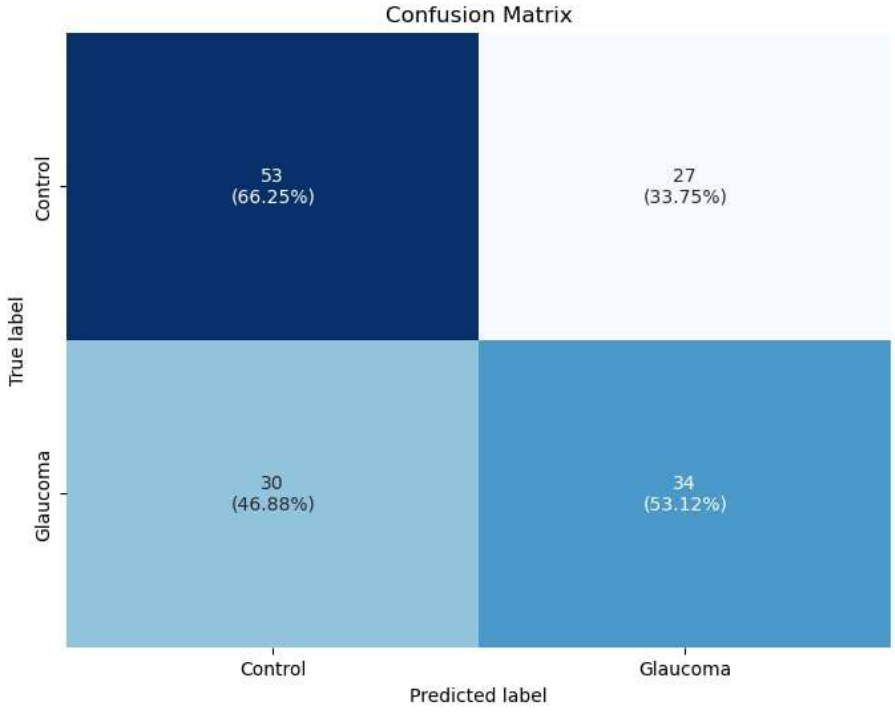

Figura D.31: Matriz de confusión caso selección de primeros 20 segundos *Filtrado + D.A.*, con ventanas de 25 muestras de largo.

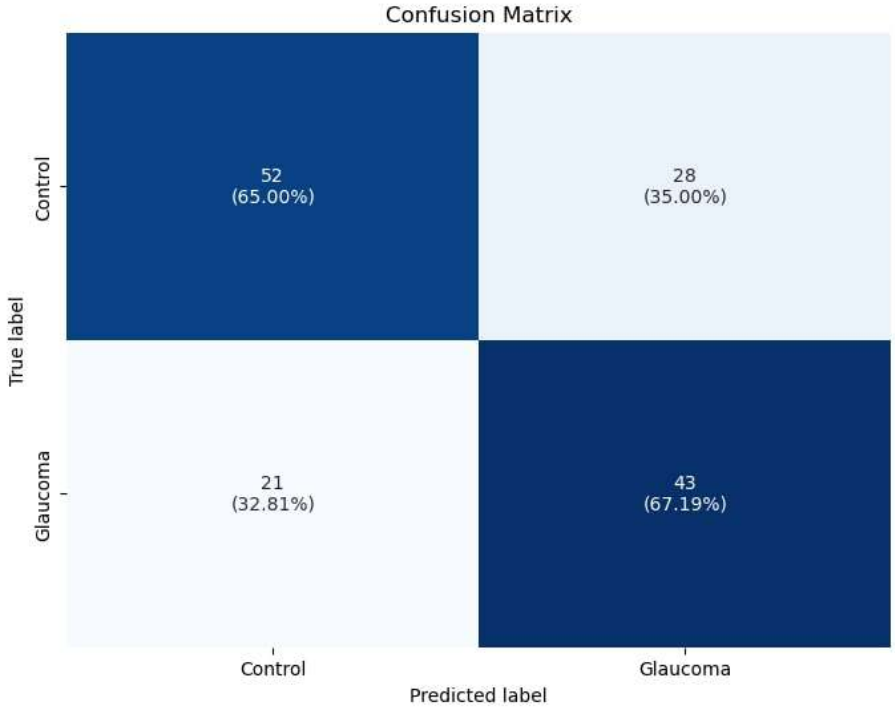

Figura D.32: Matriz de confusión caso selección de primeros 20 segundos *Filtrado + D.A.*, con ventanas de 50 y 100 muestras de largo.

D.2.2. Ventanas aleatorias

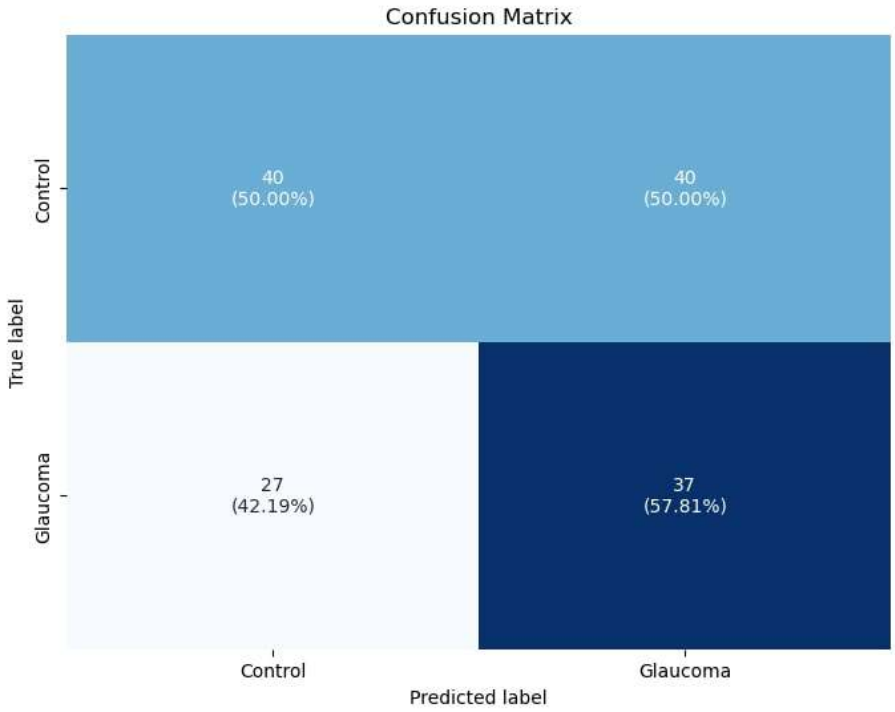

Figura D.33: Matriz de confusión caso selección de ventanas aleatorias *Normal*, con ventanas de 50 muestras de largo.

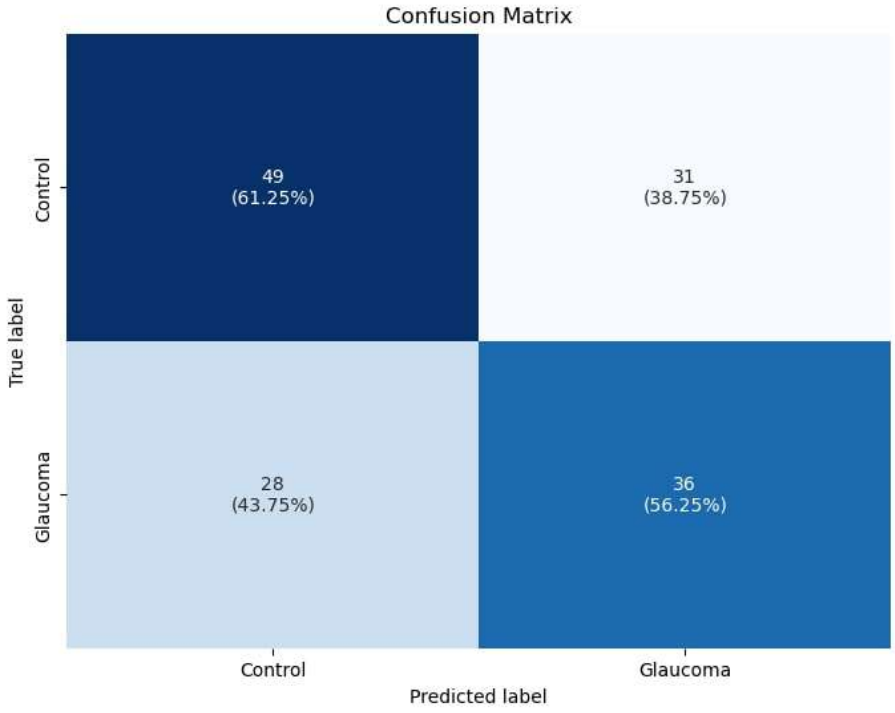

Figura D.34: Matriz de confusión caso selección de ventanas aleatorias *Normal*, con ventanas de 100 muestras de largo.

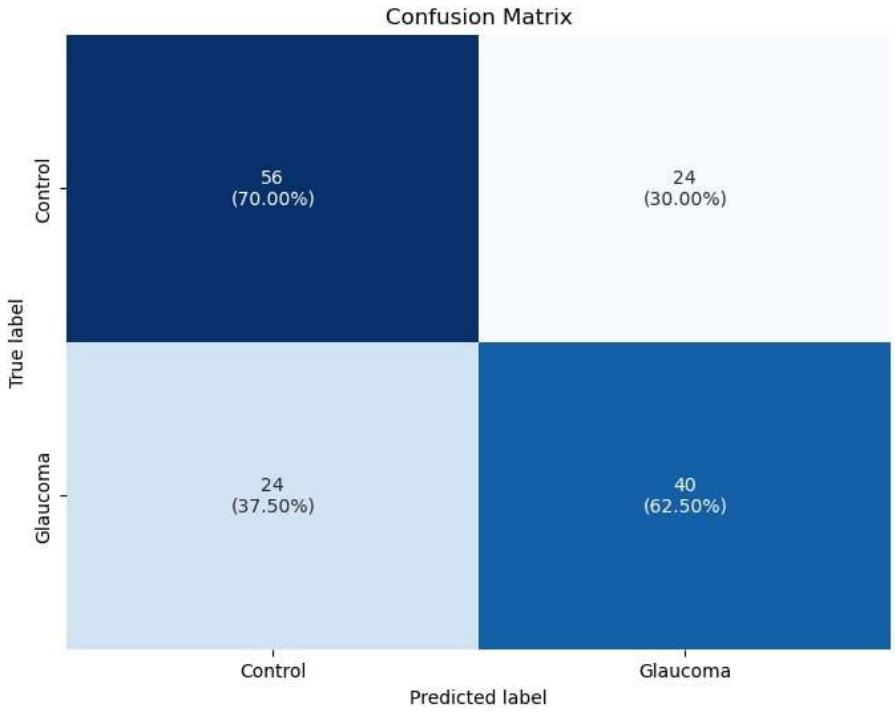

Figura D.35: Matriz de confusión caso selección de ventanas aleatorias *Normal*, con ventanas de 25 muestras de largo.

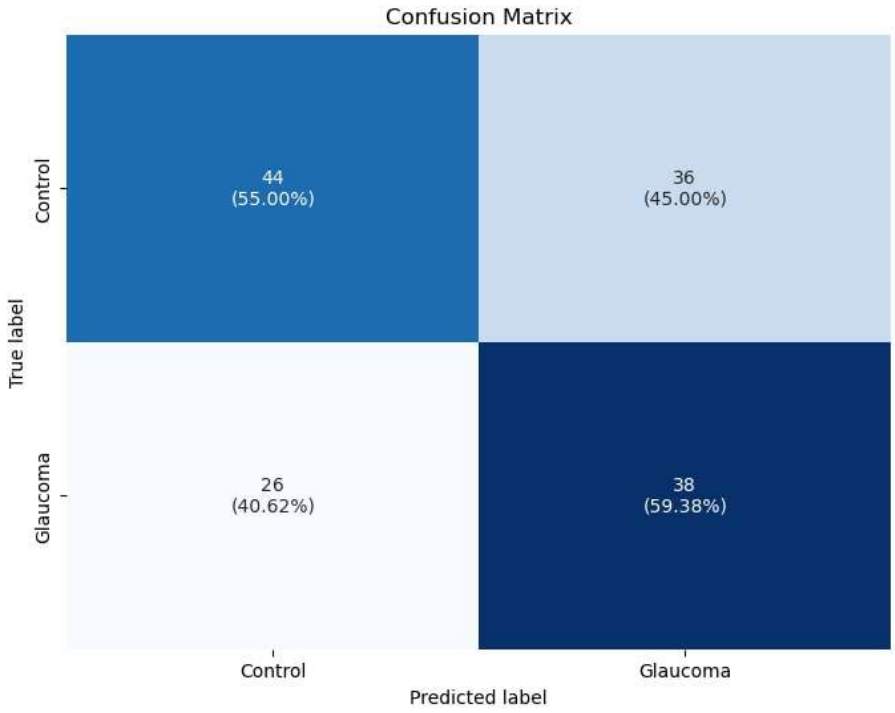

Figura D.36: Matriz de confusión caso selección de ventanas aleatorias *Normal*, con ventanas de 50 y 100 muestras de largo.

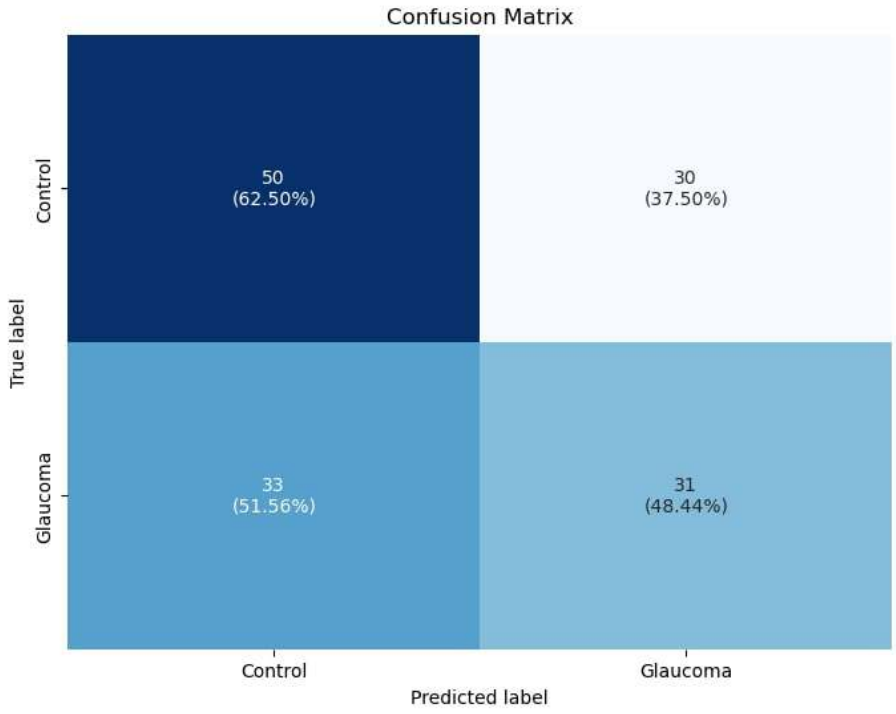

Figura D.37: Matriz de confusión caso selección de ventanas aleatorias *Filtrado*, con ventanas de 50 muestras de largo.

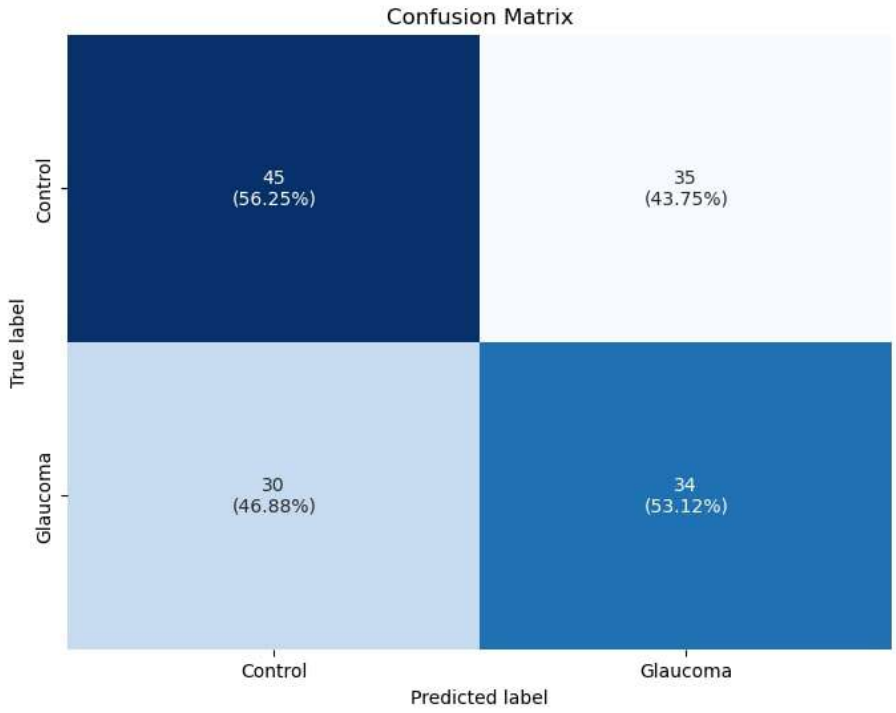

Figura D.38: Matriz de confusión caso selección de ventanas aleatorias *Filtrado*, con ventanas de 100 muestras de largo.

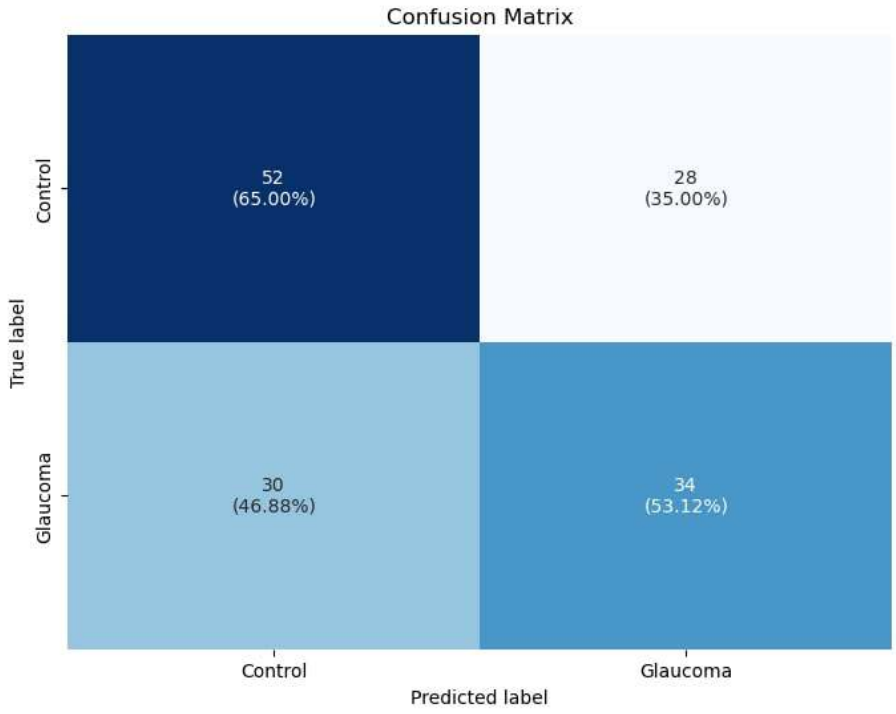

Figura D.39: Matriz de confusión caso selección de ventanas aleatorias *Filtrado*, con ventanas de 25 muestras de largo.

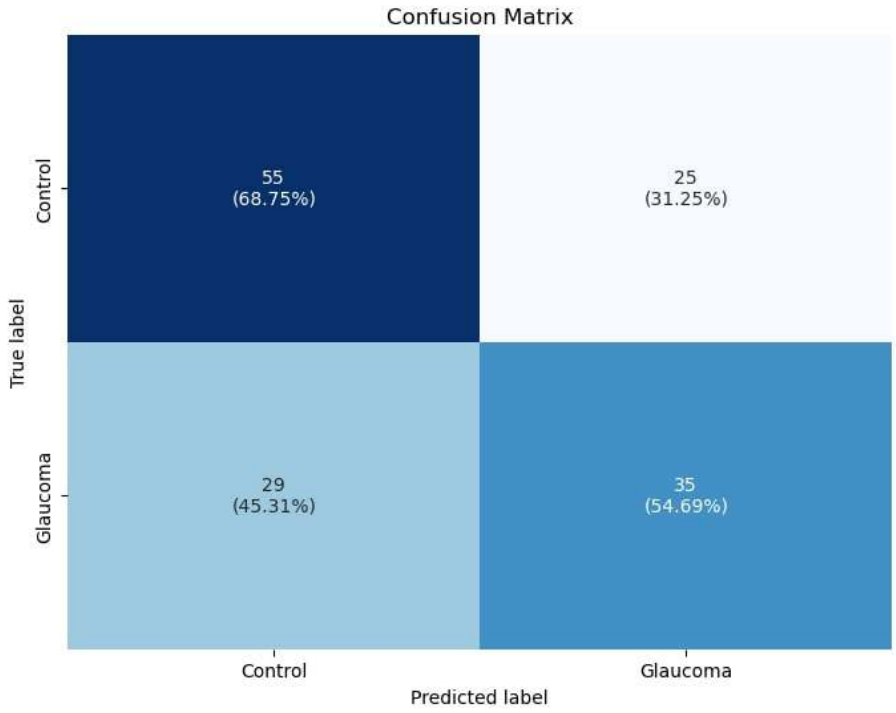

Figura D.40: Matriz de confusión caso selección de ventanas aleatorias *Filtrado*, con ventanas de 50 y 100 muestras de largo.

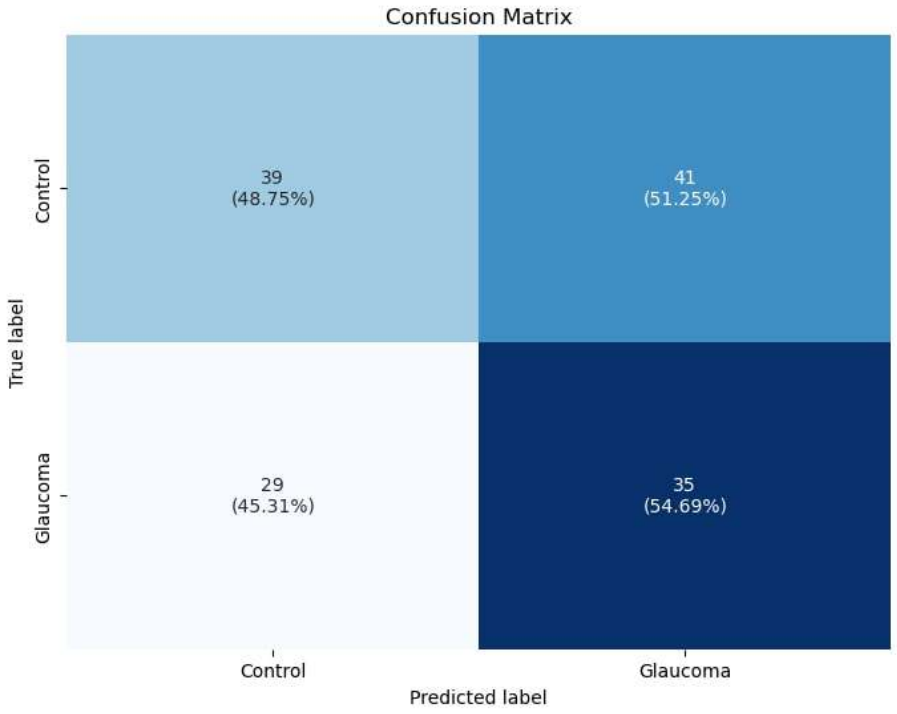

Figura D.41: Matriz de confusión caso selección de ventanas aleatorias *D.A.*, con ventanas de 50 muestras de largo.

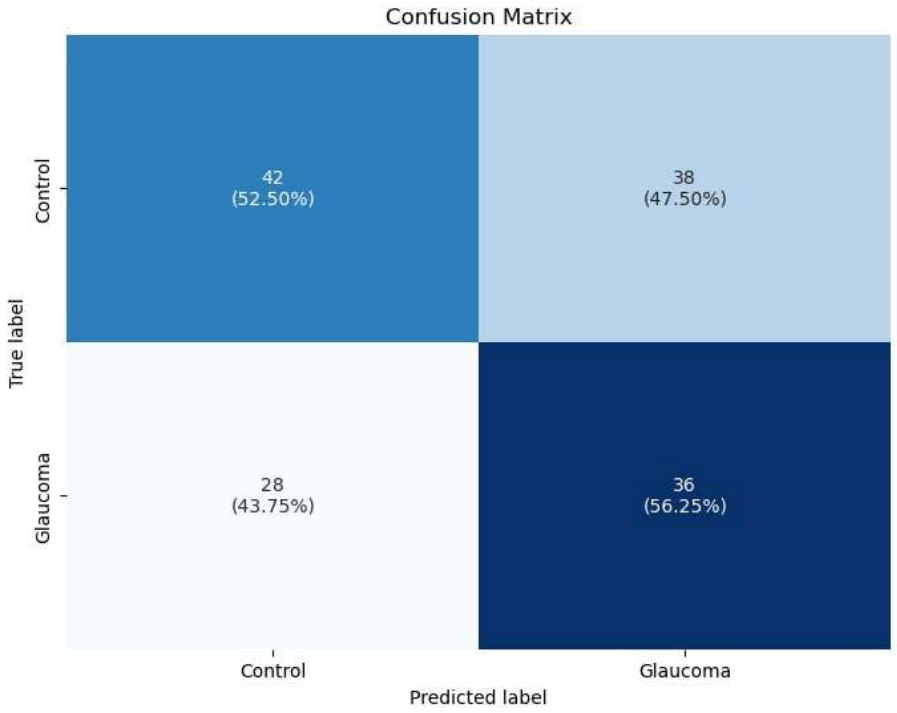

Figura D.42: Matriz de confusión caso selección de ventanas aleatorias *D.A.*, con ventanas de 100 muestras de largo.

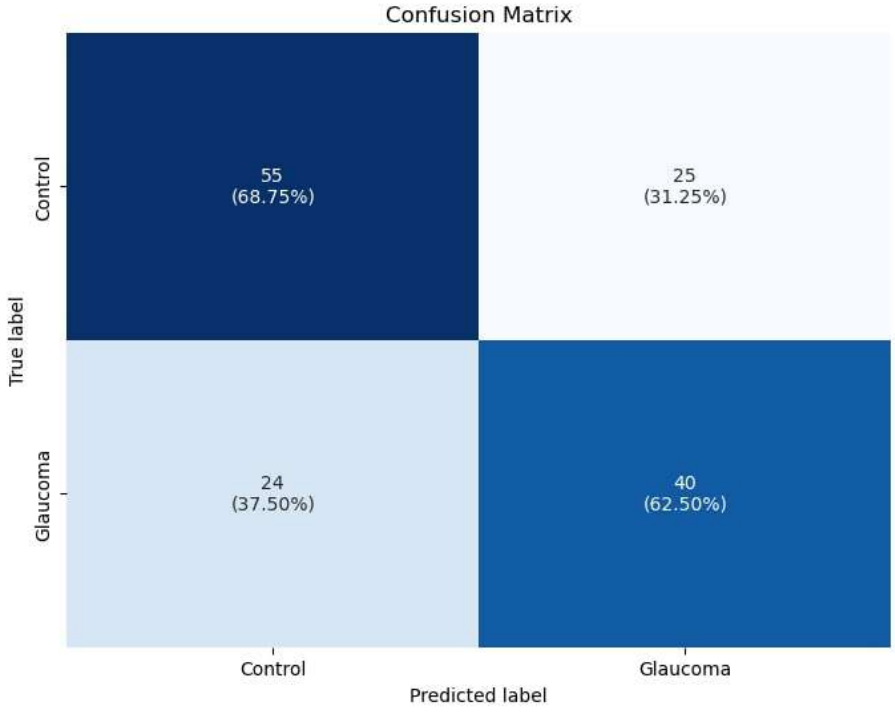

Figura D.43: Matriz de confusión caso selección de ventanas aleatorias *D.A.*, con ventanas de 25 muestras de largo.

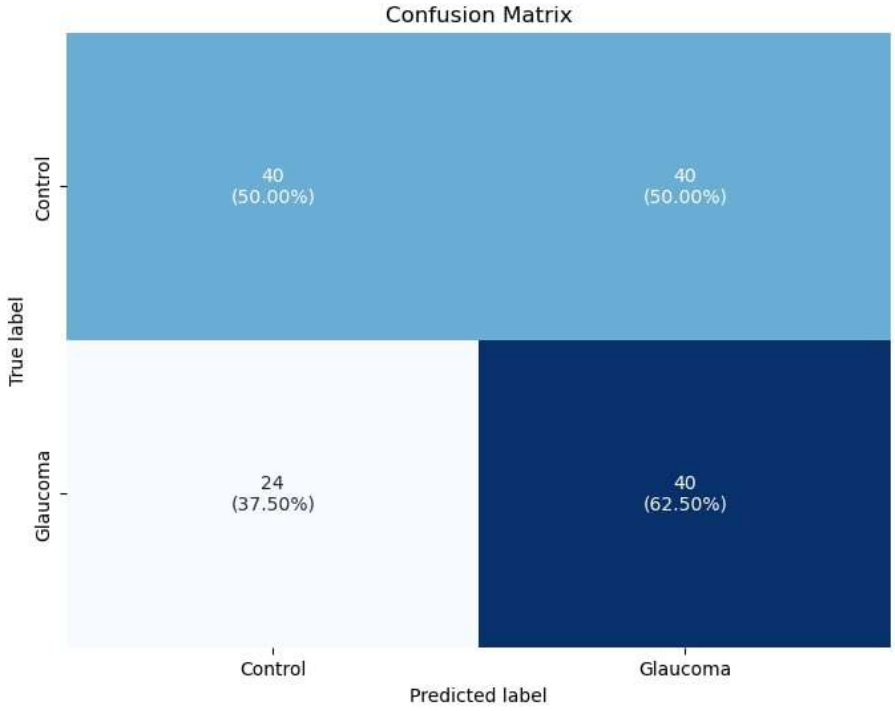

Figura D.44: Matriz de confusión caso selección de ventanas aleatorias *D.A.*, con ventanas de 50 y 100 muestras de largo.

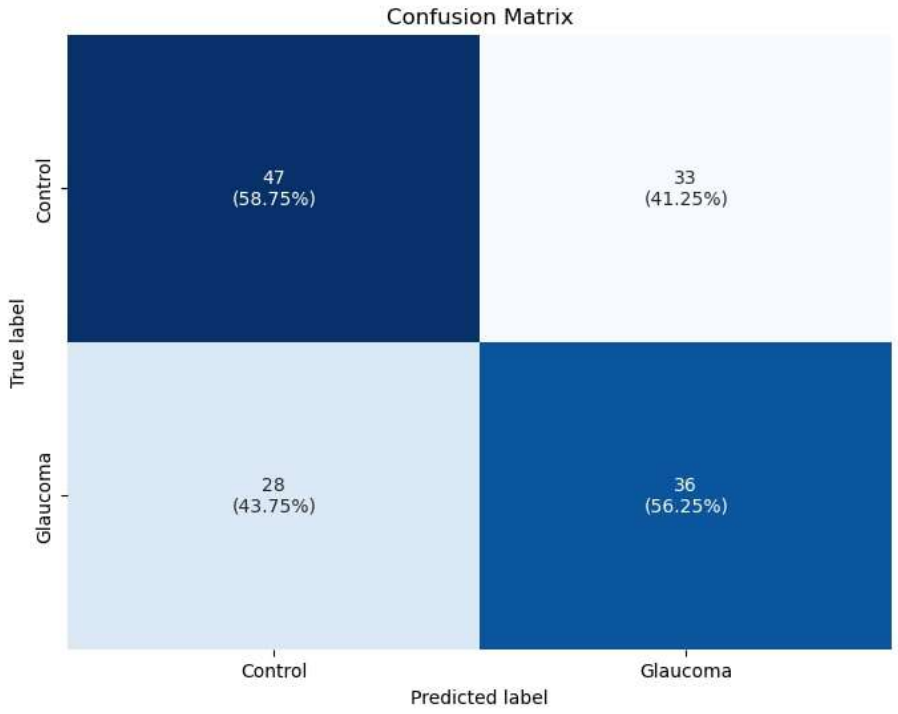

Figura D.45: Matriz de confusión caso selección de ventanas aleatorias *Filtrado + D.A.*, con ventanas de 50 muestras de largo.

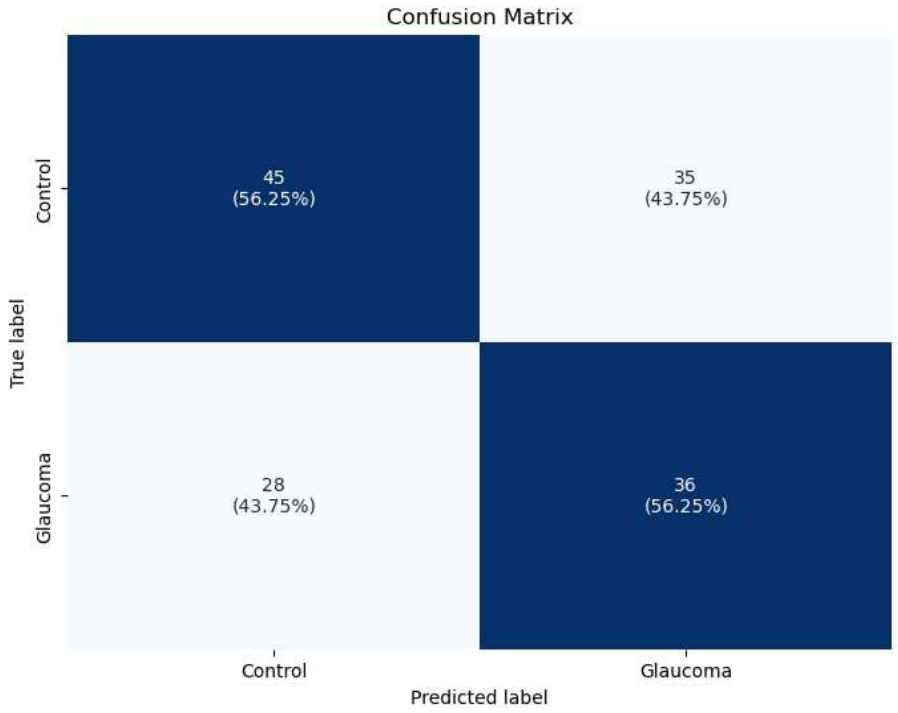

Figura D.46: Matriz de confusión caso selección de ventanas aleatorias *Filtrado + D.A.*, con ventanas de 100 muestras de largo.

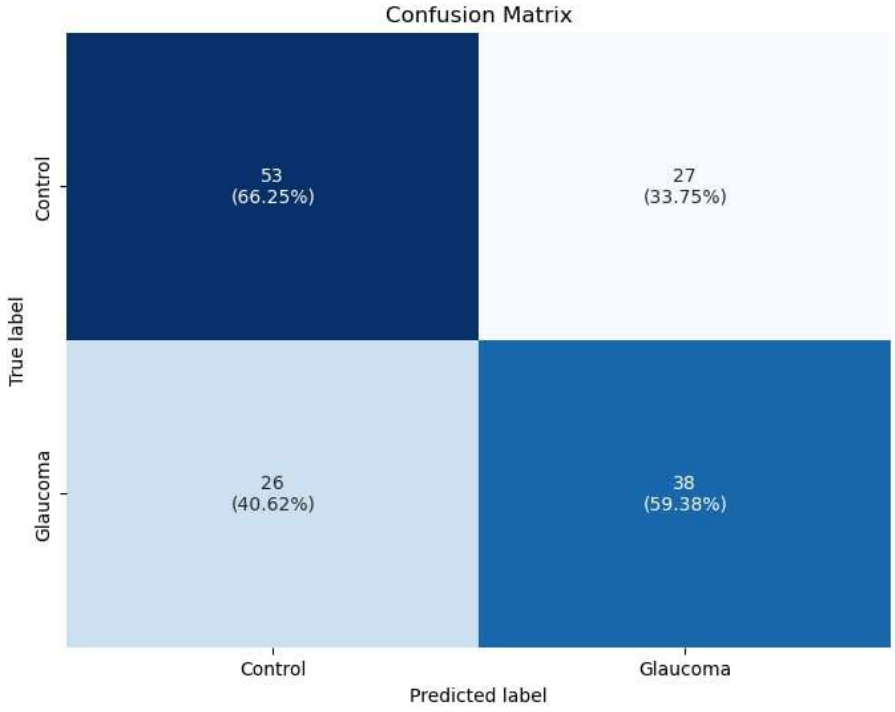

Figura D.47: Matriz de confusión caso selección de ventanas aleatorias *Filtrado + D.A.*, con ventanas de 25 muestras de largo.

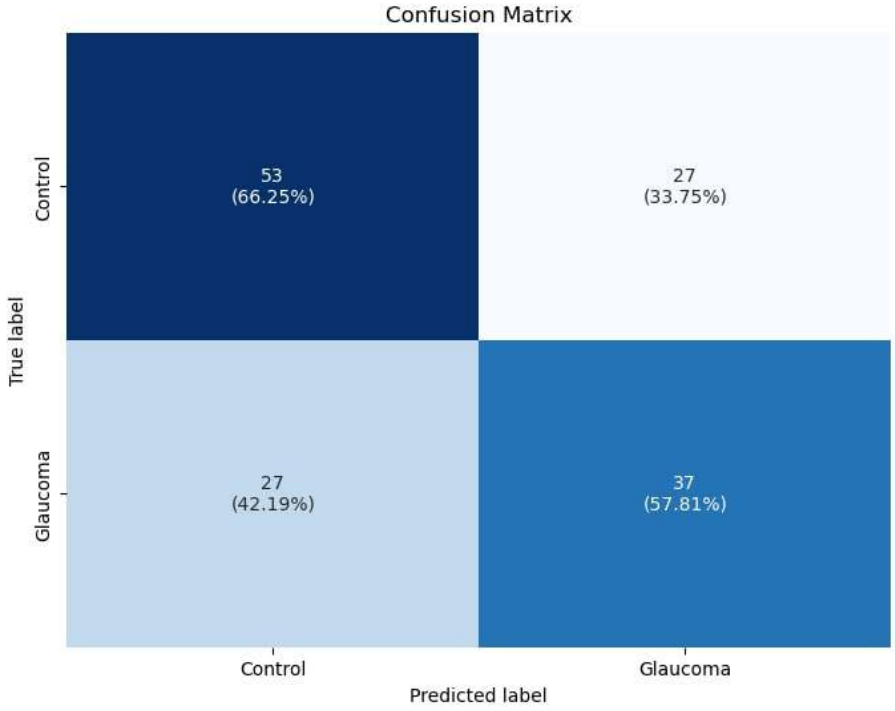

Figura D.48: Matriz de confusión caso selección de ventanas aleatorias *Filtrado + D.A.*, con ventanas de 50 y 100 muestras de largo.

D.3. Matrices de confusión caso aislamiento de características

D.3.1. Selección de respuestas ante estímulos tipo *flash*

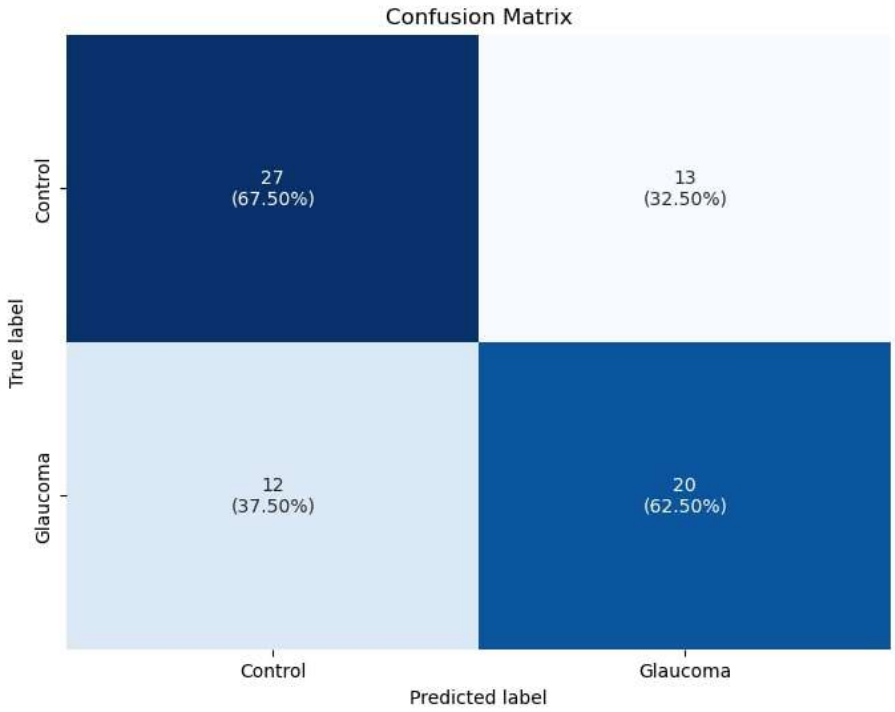

Figura D.49: Matriz de confusión caso selección de respuestas ante estímulos tipo *flash Normal*, con ventanas de 50 muestras de largo.

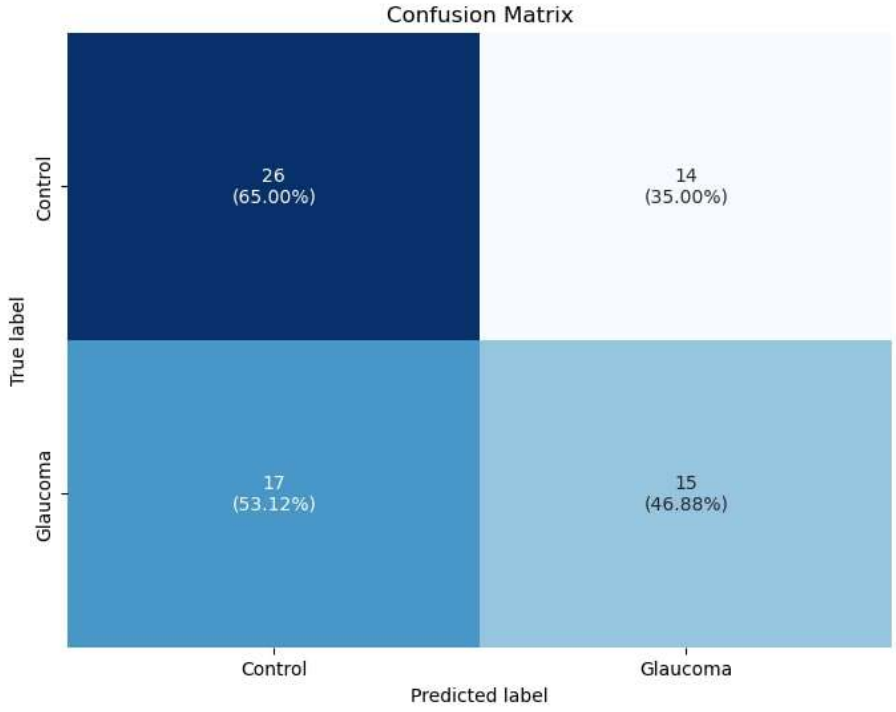

Figura D.50: Matriz de confusión caso selección de respuestas ante estímulos tipo *flash Normal*, con ventanas de 100 muestras de largo.

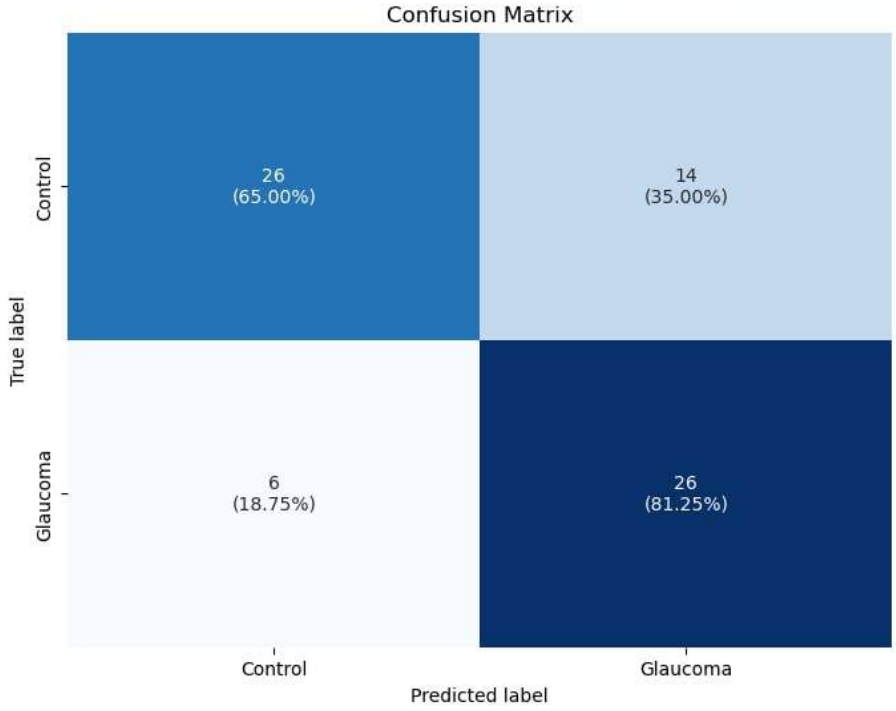

Figura D.51: Matriz de confusión caso selección de respuestas ante estímulos tipo *flash Normal*, con ventanas de 25 muestras de largo.

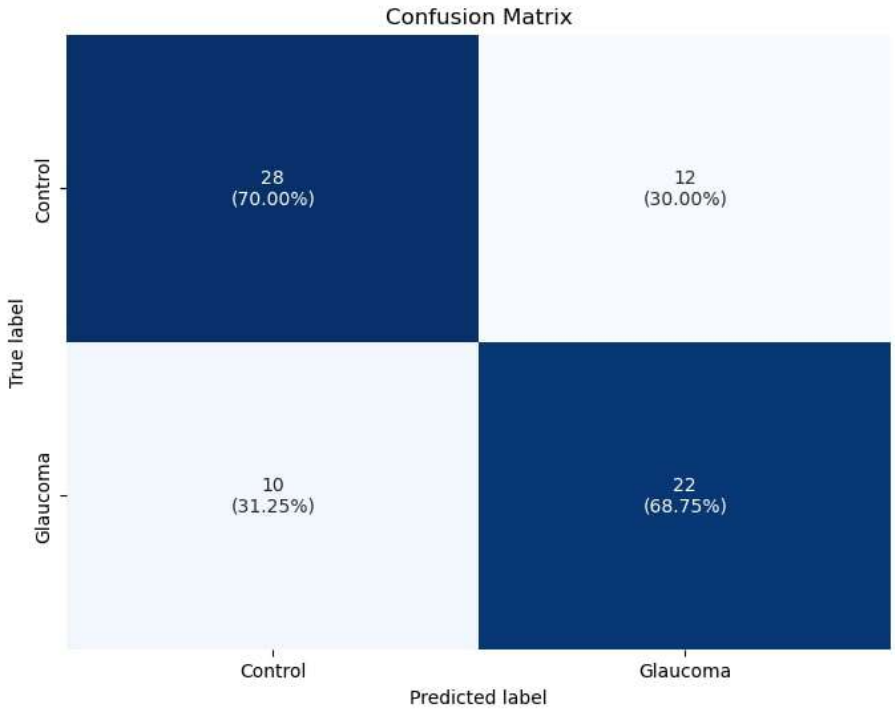

Figura D.52: Matriz de confusión caso selección de respuestas ante estímulos tipo *flash Normal*, con ventanas de 50 y 100 muestras de largo.

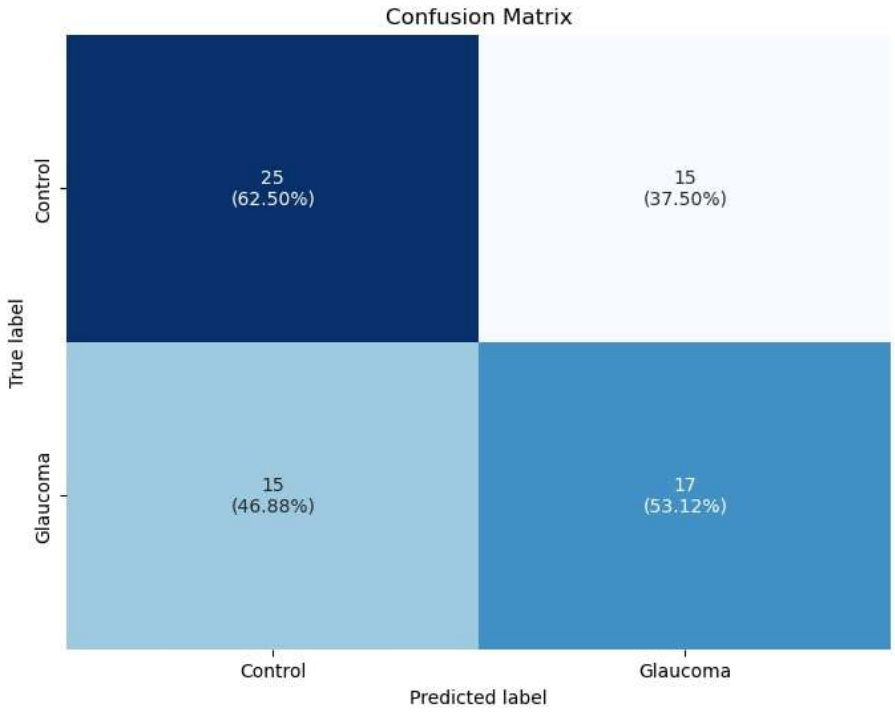

Figura D.53: Matriz de confusión caso selección de respuestas ante estímulos tipo *flash Normal*, con ventanas de 15 muestras de largo.

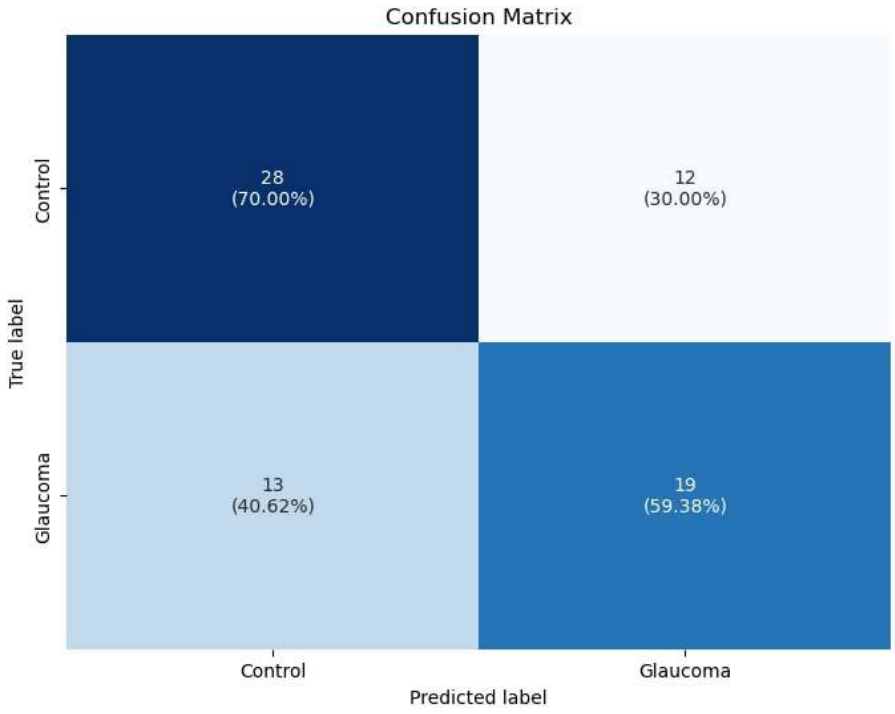

Figura D.54: Matriz de confusión caso selección de respuestas ante estímulos tipo *flash Filtrado*, con ventanas de 50 muestras de largo.

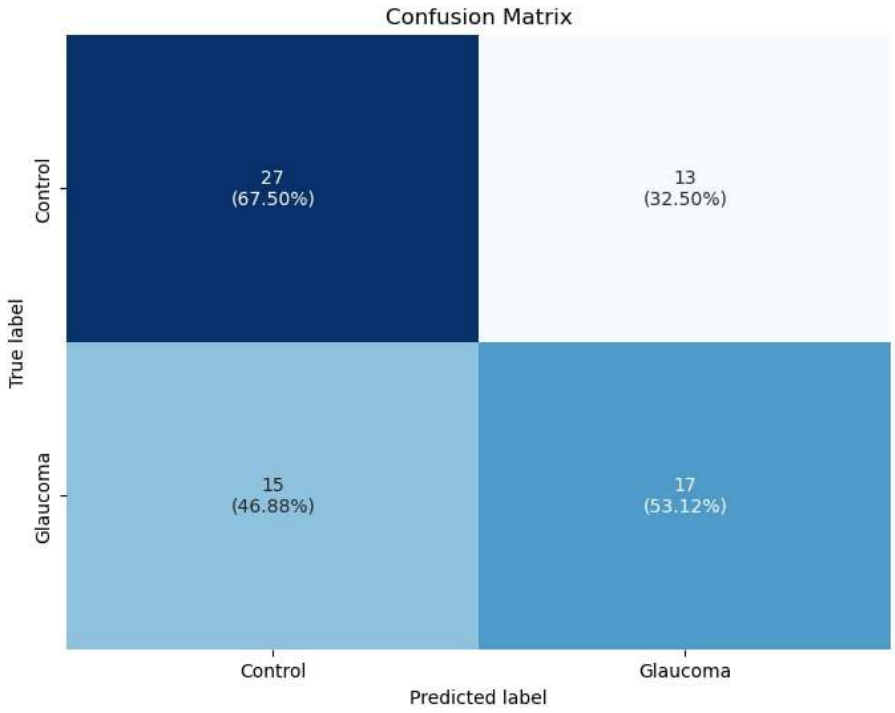

Figura D.55: Matriz de confusión caso selección de respuestas ante estímulos tipo *flash Filtrado*, con ventanas de 100 muestras de largo.

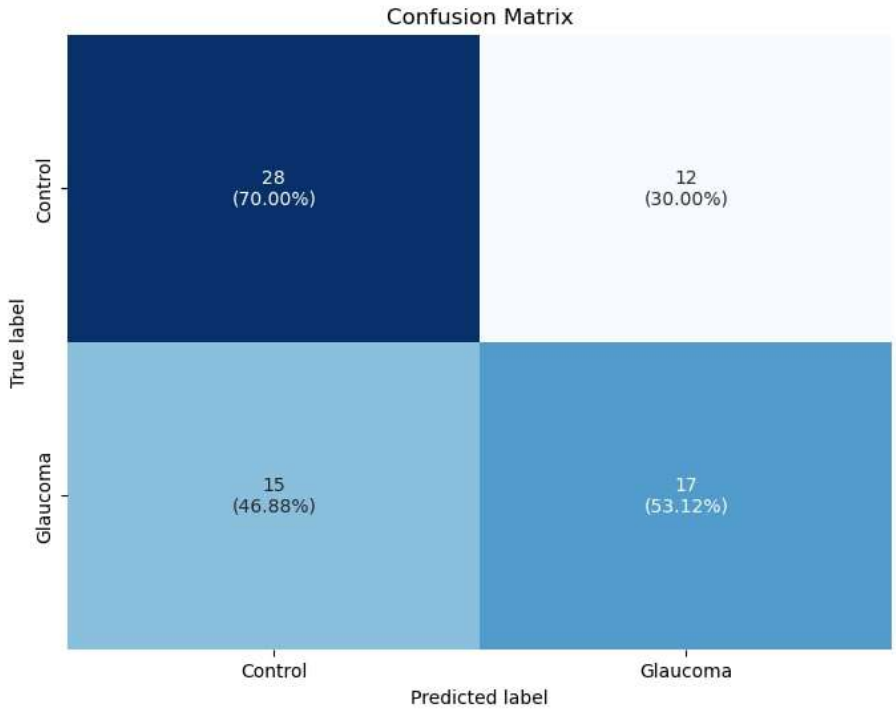

Figura D.56: Matriz de confusión caso selección de respuestas ante estímulos tipo *flash Filtrado*, con ventanas de 25 muestras de largo.

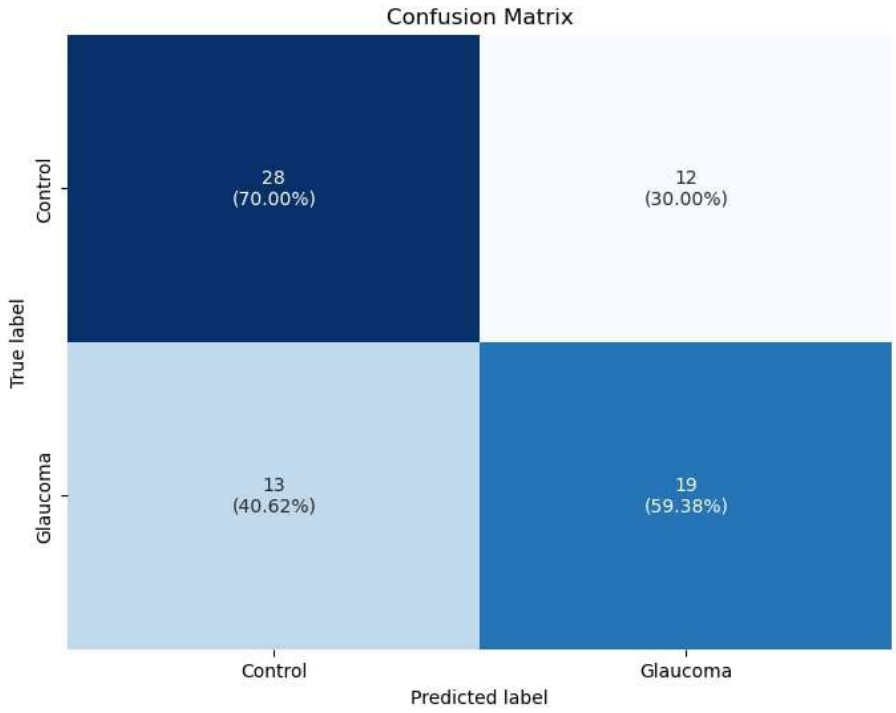

Figura D.57: Matriz de confusión caso selección de respuestas ante estímulos tipo *flash Filtrado*, con ventanas de 50 y 100 muestras de largo.

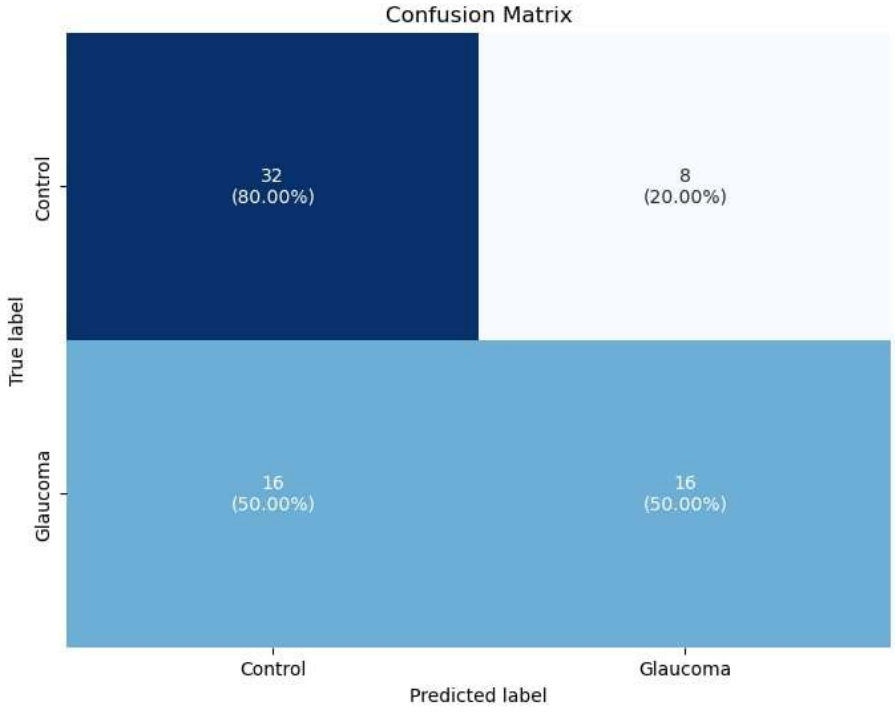

Figura D.58: Matriz de confusión caso selección de respuestas ante estímulos tipo *flash Filtrado*, con ventanas de 15 muestras de largo.

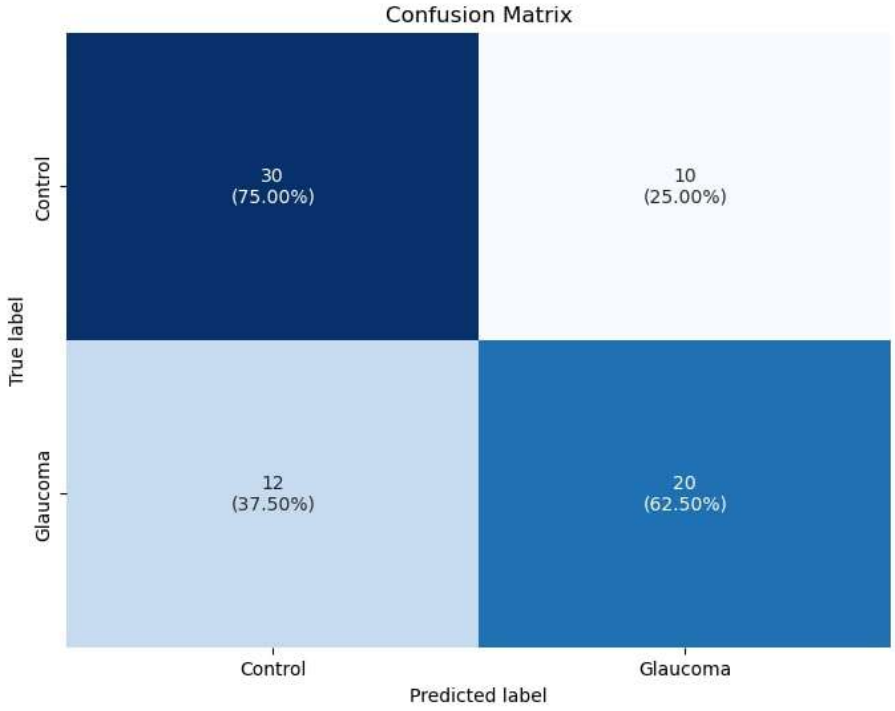

Figura D.59: Matriz de confusión caso selección de respuestas ante estímulos tipo *flash D.A.*, con ventanas de 50 muestras de largo.

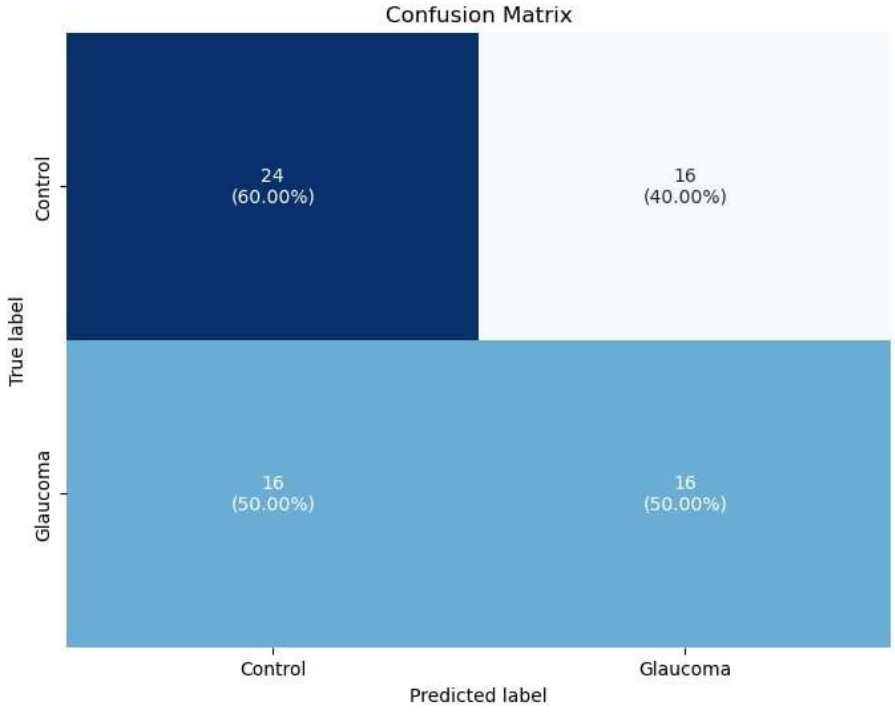

Figura D.60: Matriz de confusión caso selección de respuestas ante estímulos tipo *flash D.A.*, con ventanas de 100 muestras de largo.

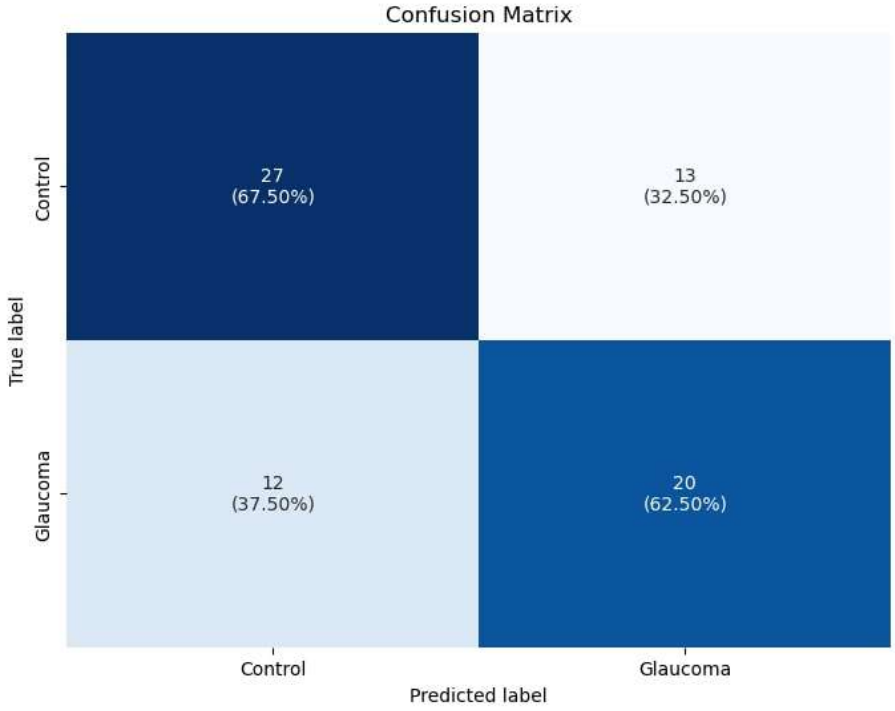

Figura D.61: Matriz de confusión caso selección de respuestas ante estímulos tipo *flash D.A.*, con ventanas de 25 muestras de largo.

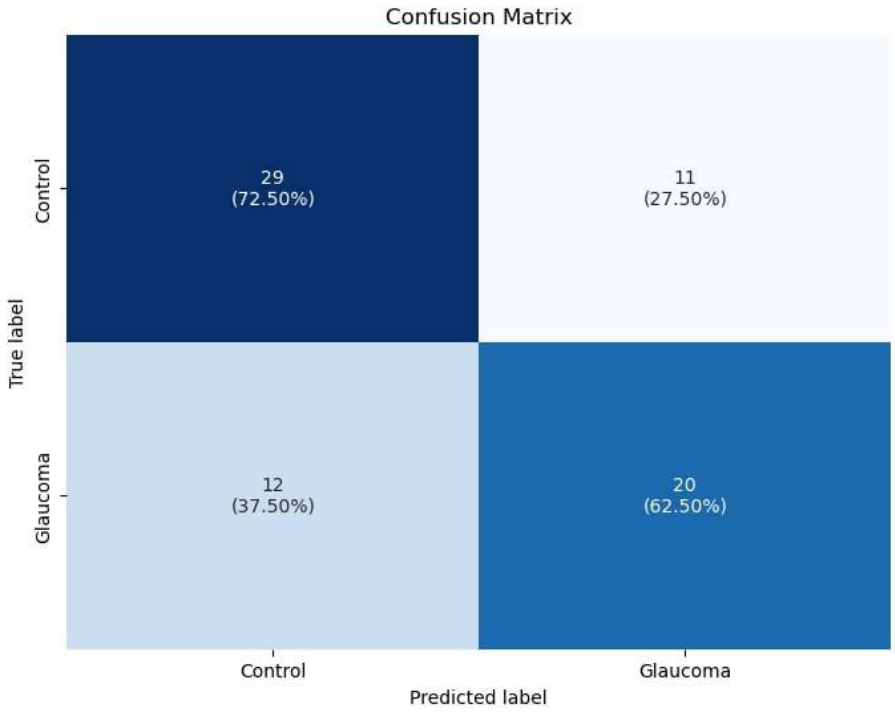

Figura D.62: Matriz de confusión caso selección de respuestas ante estímulos tipo *flash D.A.*, con ventanas de 50 y 100 muestras de largo.

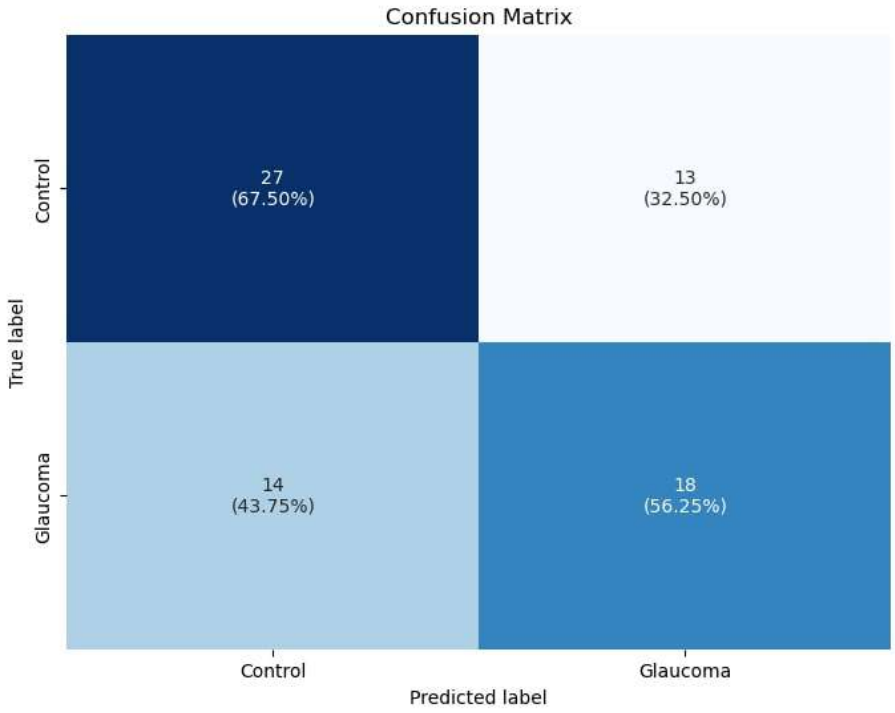

Figura D.63: Matriz de confusión caso selección de respuestas ante estímulos tipo *flash D.A.*, con ventanas de 15 muestras de largo.

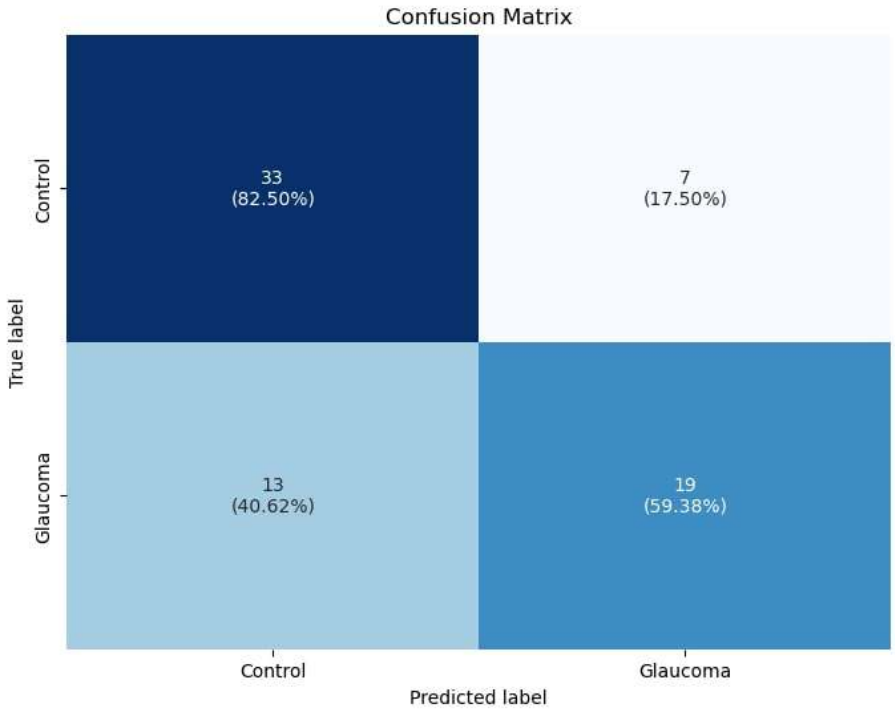

Figura D.64: Matriz de confusión caso selección de respuestas ante estímulos tipo *flash Filtrado + D.A.*, con ventanas de 50 muestras de largo.

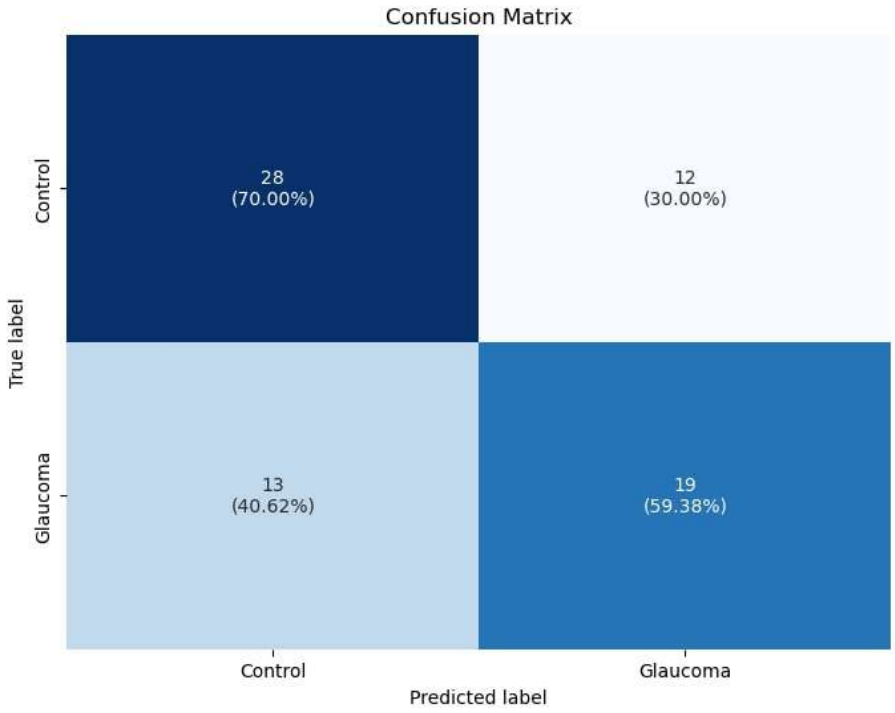

Figura D.65: Matriz de confusión caso selección de respuestas ante estímulos tipo *flash Filtrado + D.A.*, con ventanas de 100 muestras de largo.

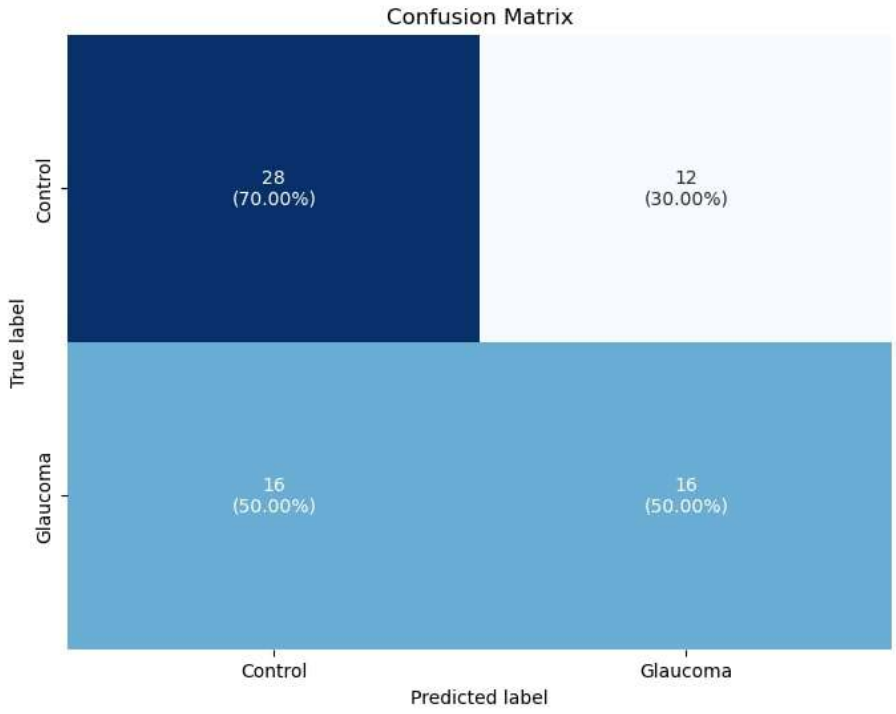

Figura D.66: Matriz de confusión caso selección de respuestas ante estímulos tipo *flash Filtrado + D.A.*, con ventanas de 25 muestras de largo.

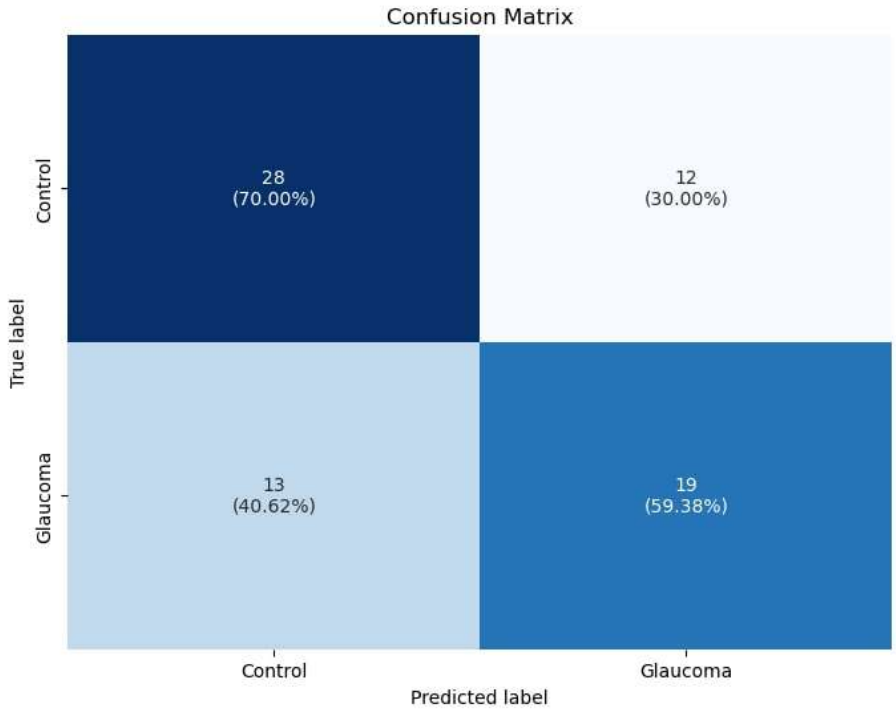

Figura D.67: Matriz de confusión caso selección de respuestas ante estímulos tipo *flash Filtrado + D.A.*, con ventanas de 50 y 100 muestras de largo.

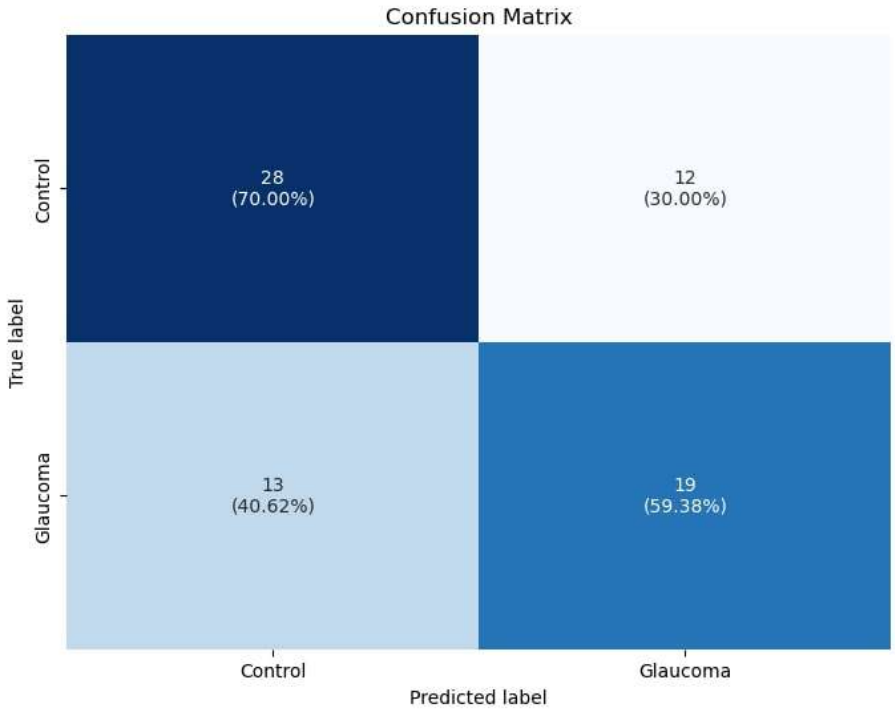

Figura D.68: Matriz de confusión caso selección de respuestas ante estímulos tipo *flash Filtrado + D.A.*, con ventanas de 15 muestras de largo.

D.3.2. Selección de respuestas ante estímulos tipo rampa

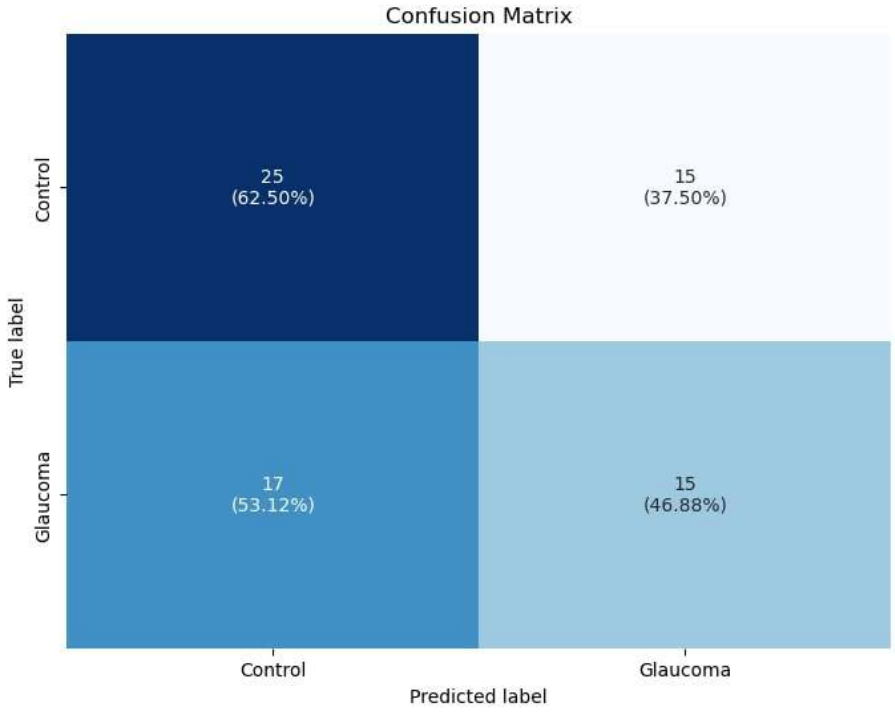

Figura D.69: Matriz de confusión caso selección de respuestas ante estímulos tipo rampa *Normal*, con ventanas de 50 muestras de largo.

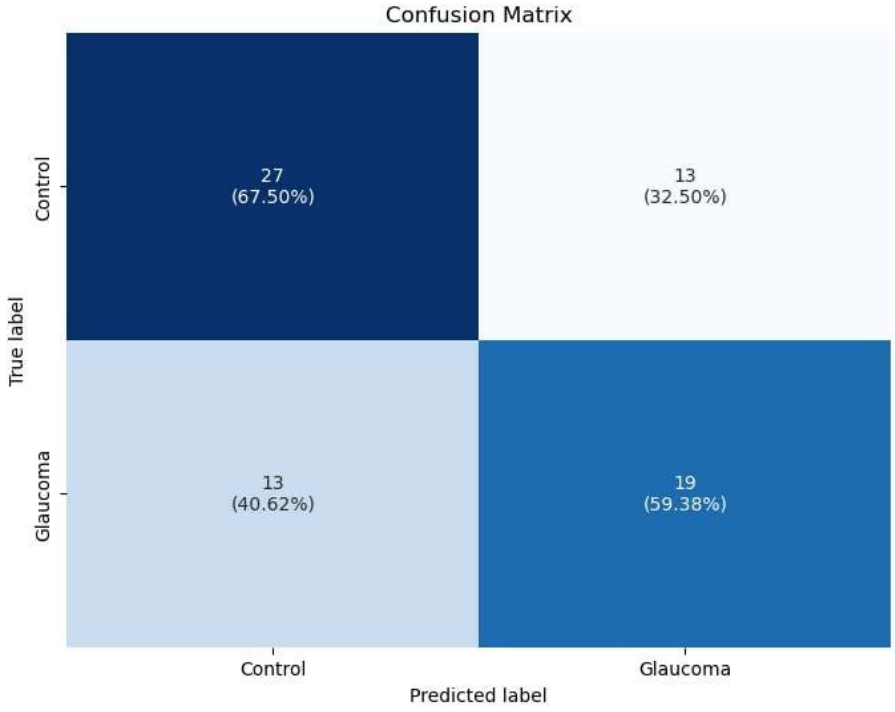

Figura D.70: Matriz de confusión caso selección de respuestas ante estímulos tipo rampa *Normal*, con ventanas de 100 muestras de largo.

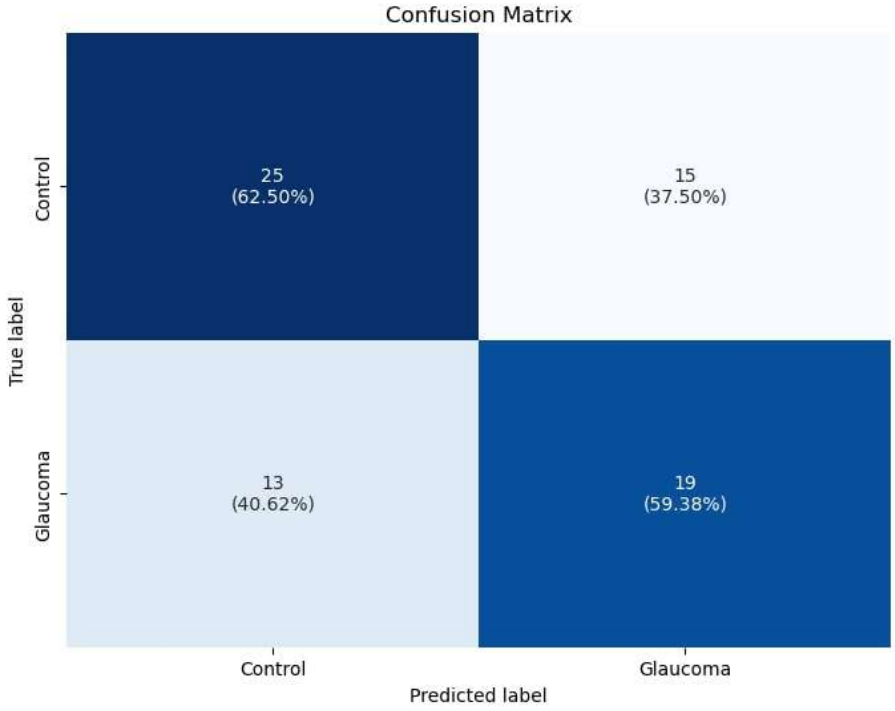

Figura D.71: Matriz de confusión caso selección de respuestas ante estímulos tipo rampa *Normal*, con ventanas de 25 muestras de largo.

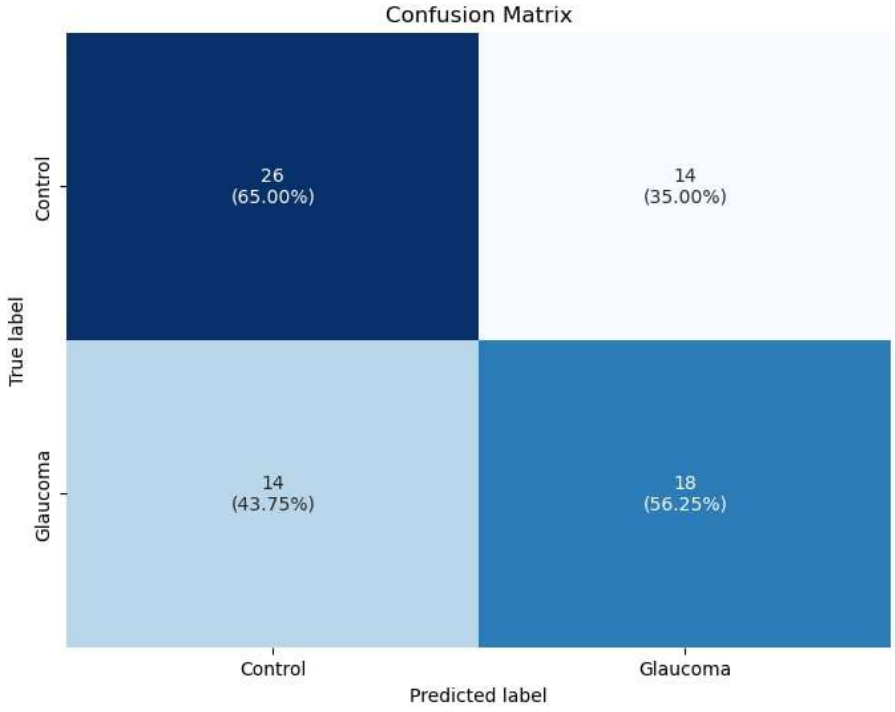

Figura D.72: Matriz de confusión caso selección de respuestas ante estímulos tipo rampa *Normal*, con ventanas de 50 y 100 muestras de largo.

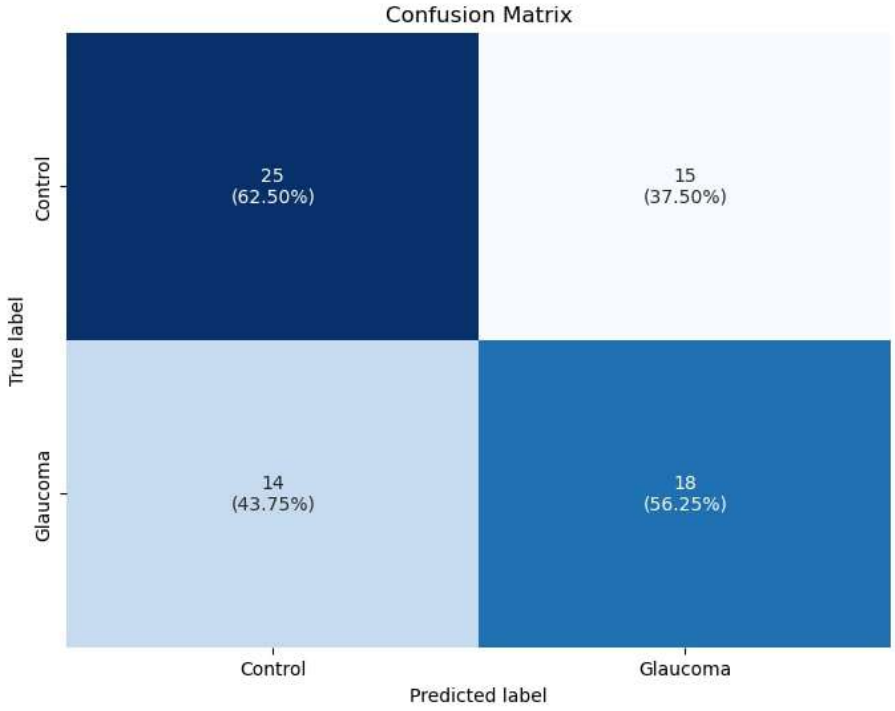

Figura D.73: Matriz de confusión caso selección de respuestas ante estímulos tipo rampa *Normal*, con ventanas de 150 muestras de largo.

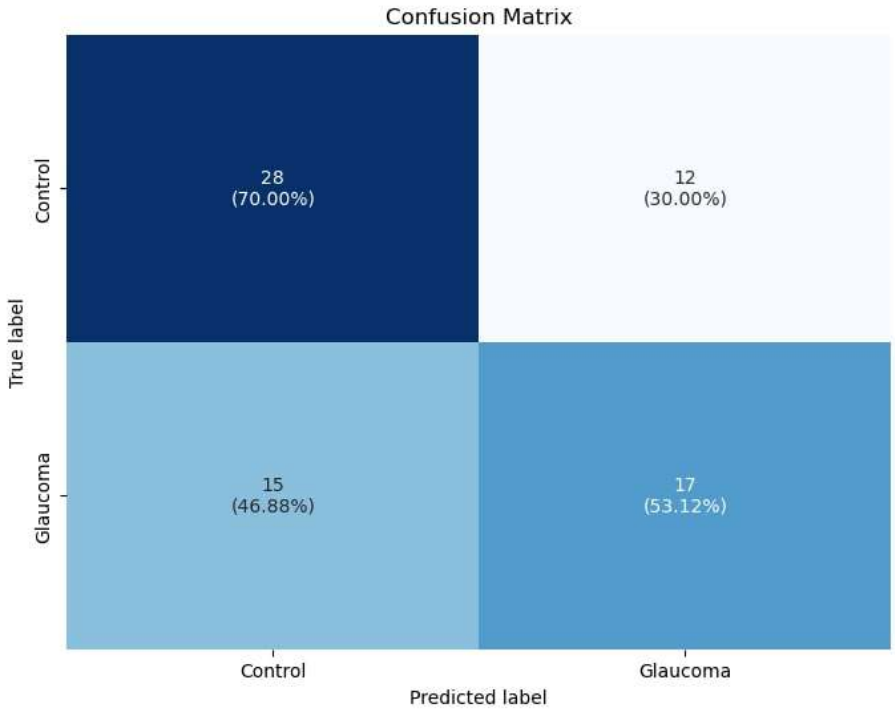

Figura D.74: Matriz de confusión caso selección de respuestas ante estímulos tipo rampa *Filtrado*, con ventanas de 50 muestras de largo.

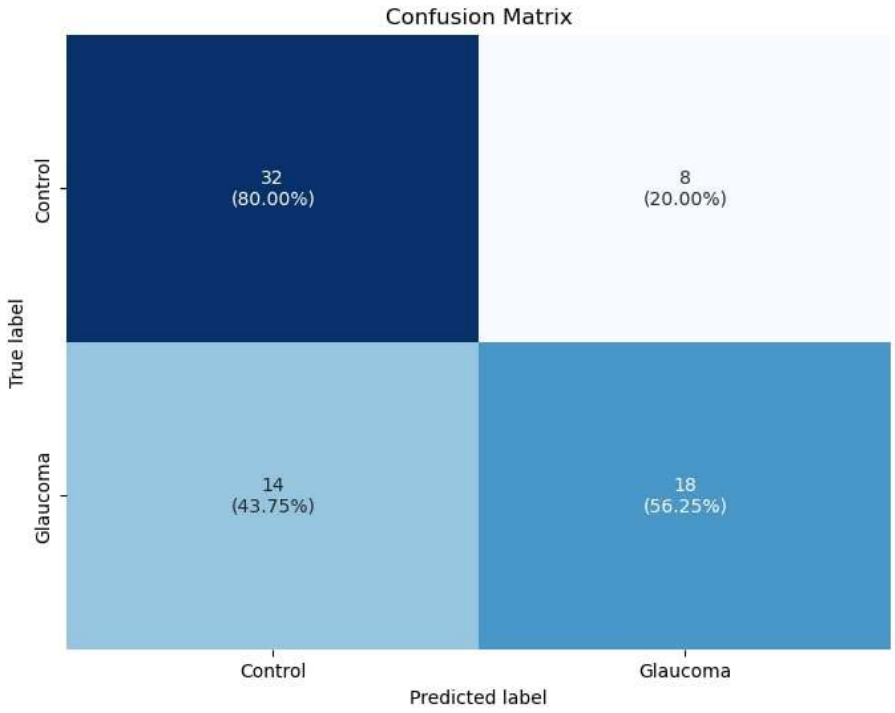

Figura D.75: Matriz de confusión caso selección de respuestas ante estímulos tipo rampa *Filtrado*, con ventanas de 100 muestras de largo.

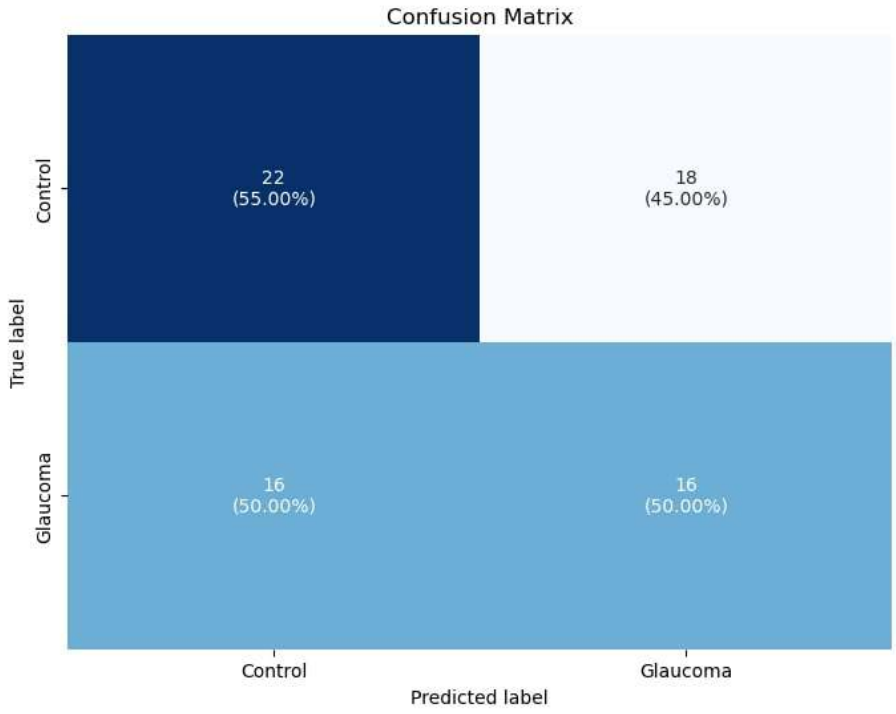

Figura D.76: Matriz de confusión caso selección de respuestas ante estímulos tipo rampa *Filtrado*, con ventanas de 25 muestras de largo.

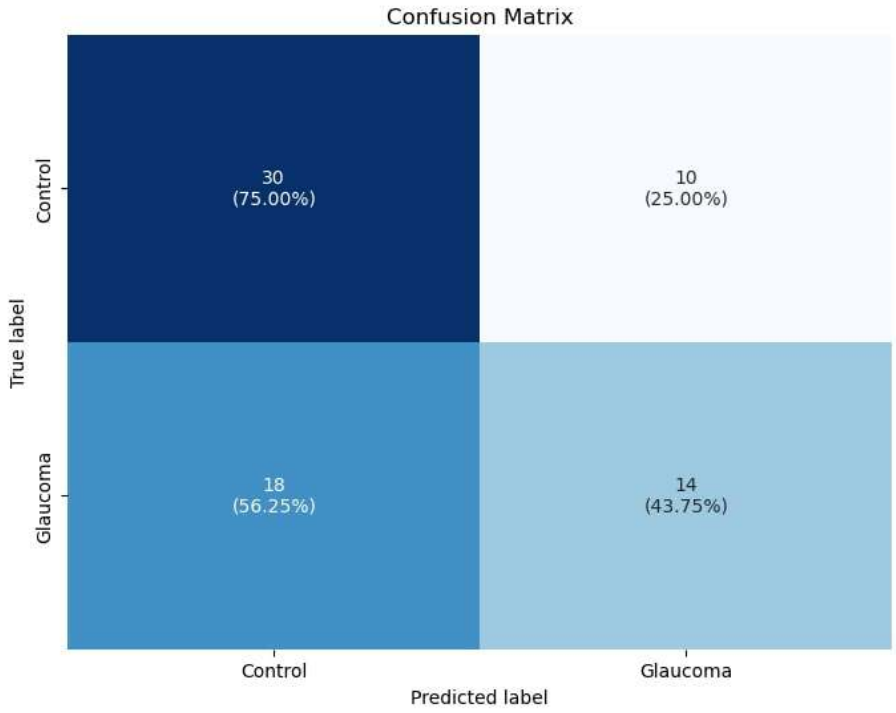

Figura D.77: Matriz de confusión caso selección de respuestas ante estímulos tipo rampa *Filtrado*, con ventanas de 50 y 100 muestras de largo.

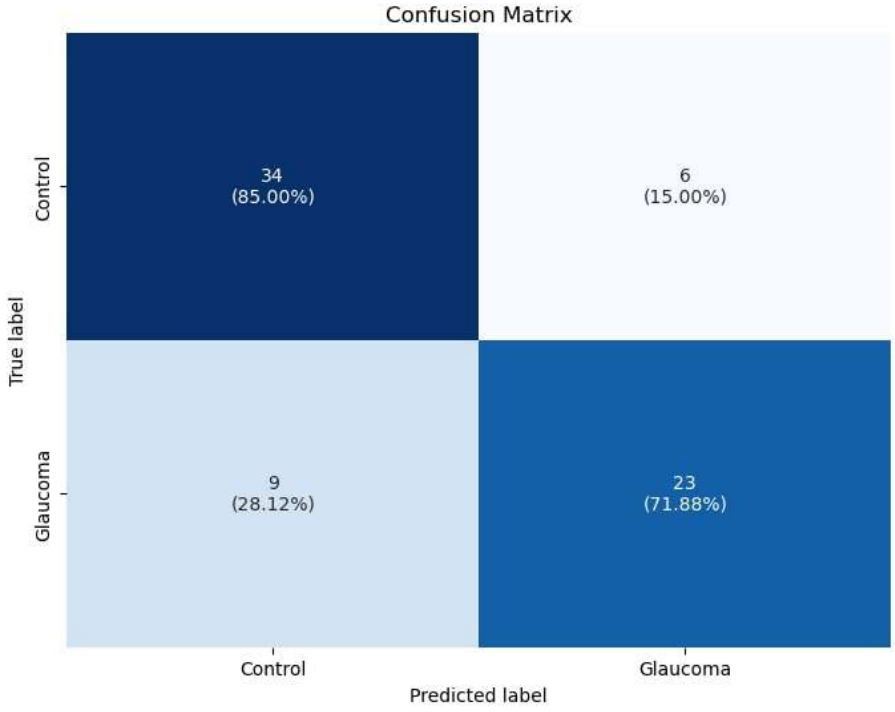

Figura D.78: Matriz de confusión caso selección de respuestas ante estímulos tipo rampa *Filtrado*, con ventanas de 150 muestras de largo.

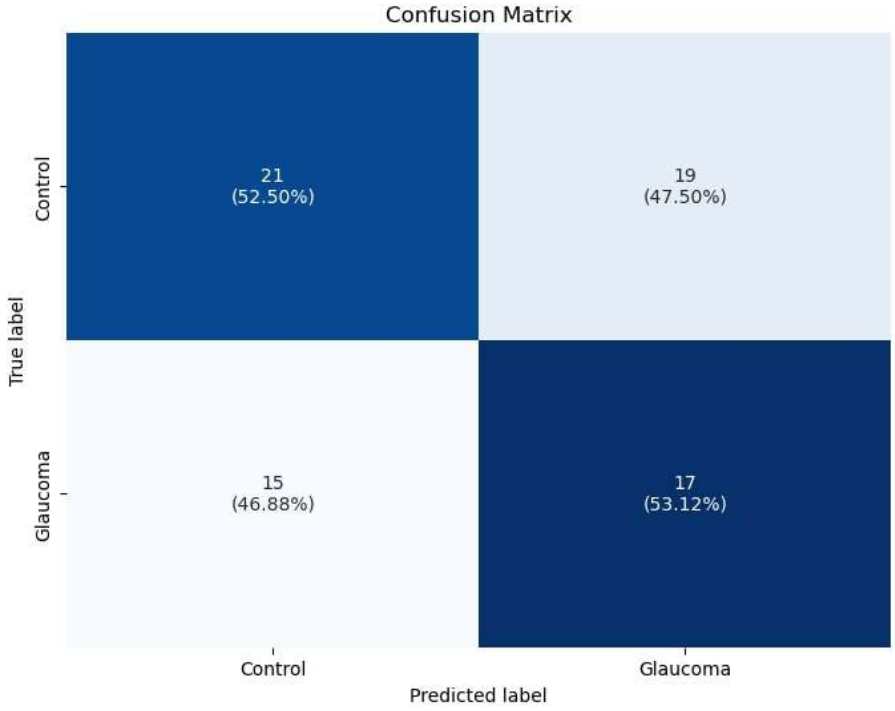

Figura D.79: Matriz de confusión caso selección de respuestas ante estímulos tipo rampa *D.A.*, con ventanas de 50 muestras de largo.

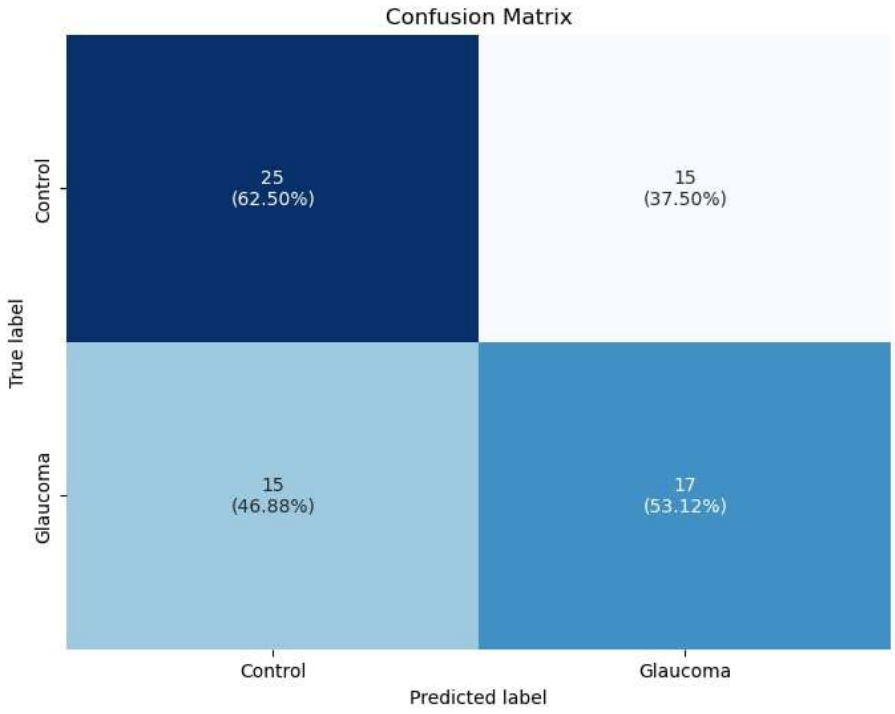

Figura D.80: Matriz de confusión caso selección de respuestas ante estímulos tipo rampa *D.A.*, con ventanas de 100 muestras de largo.

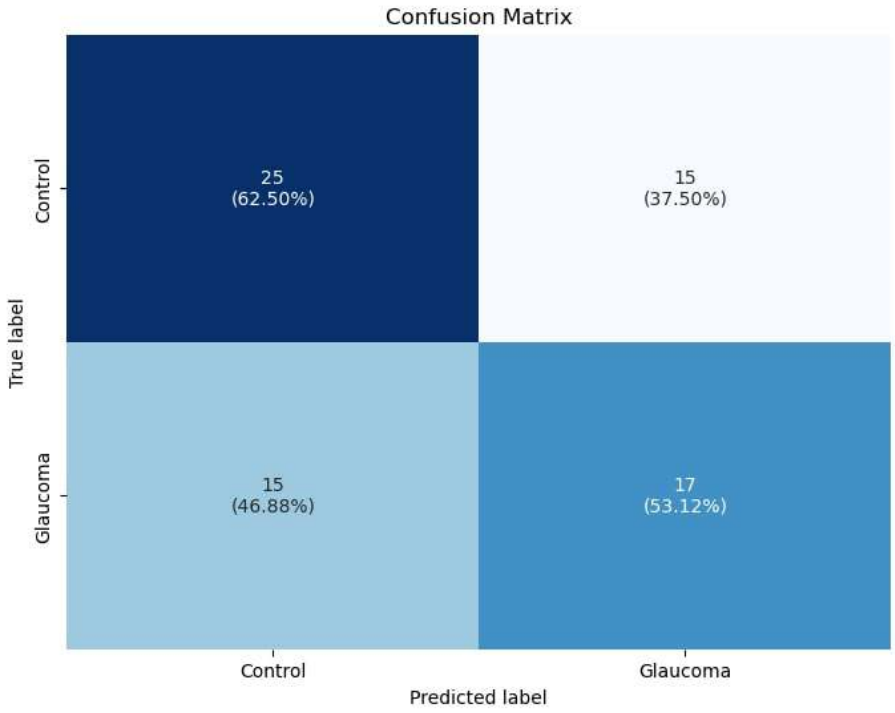

Figura D.81: Matriz de confusión caso selección de respuestas ante estímulos tipo rampa *D.A.*, con ventanas de 25 muestras de largo.

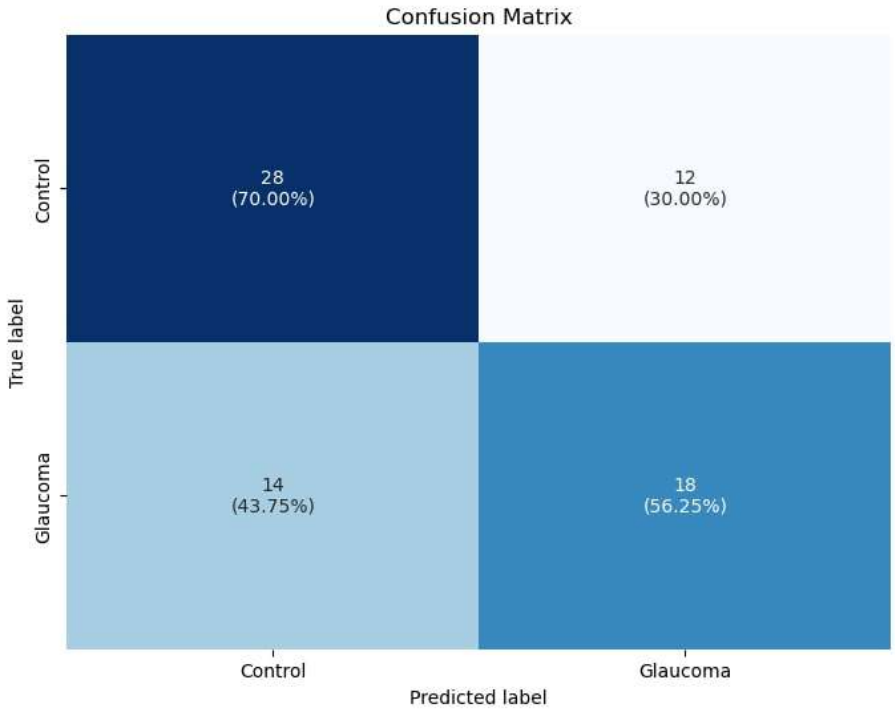

Figura D.82: Matriz de confusión caso selección de respuestas ante estímulos tipo rampa *D.A.*, con ventanas de 50 y 100 muestras de largo.

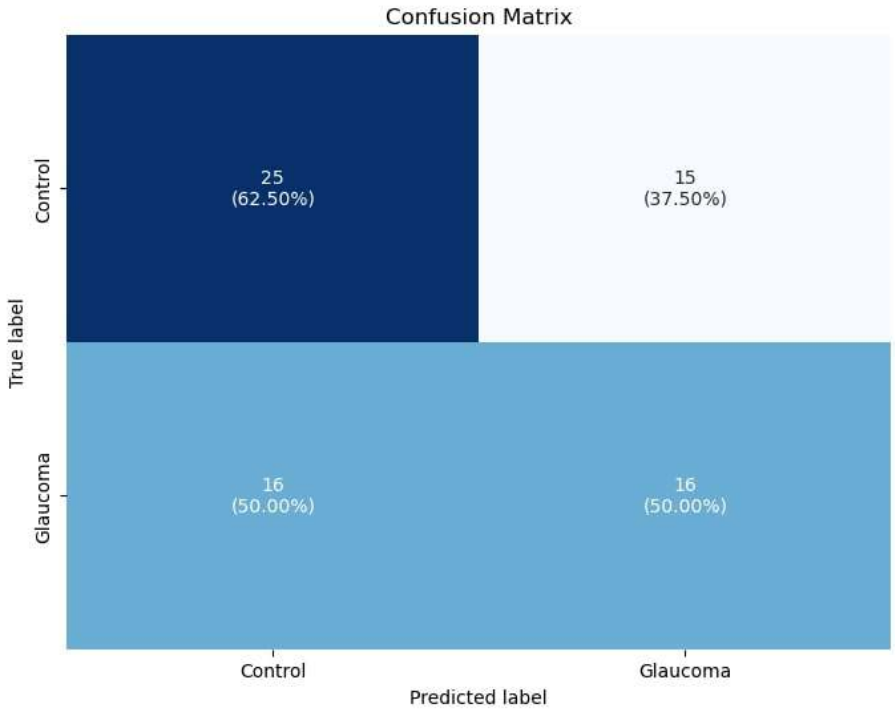

Figura D.83: Matriz de confusión caso selección de respuestas ante estímulos tipo rampa *D.A.*, con ventanas de 150 muestras de largo.

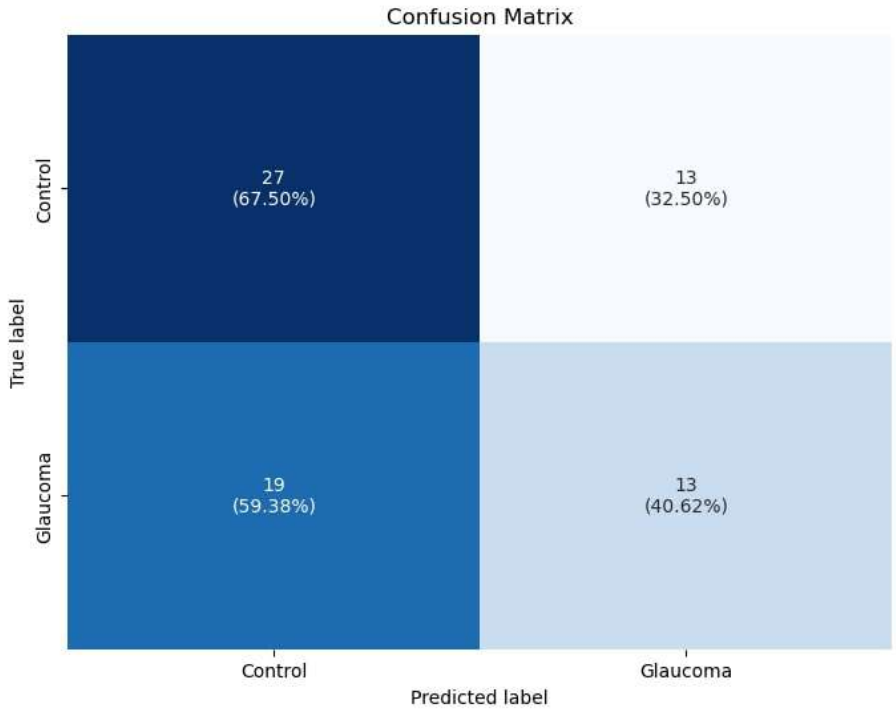

Figura D.84: Matriz de confusión caso selección de respuestas ante estímulos tipo rampa *Filtrado + D.A.*, con ventanas de 50 muestras de largo.

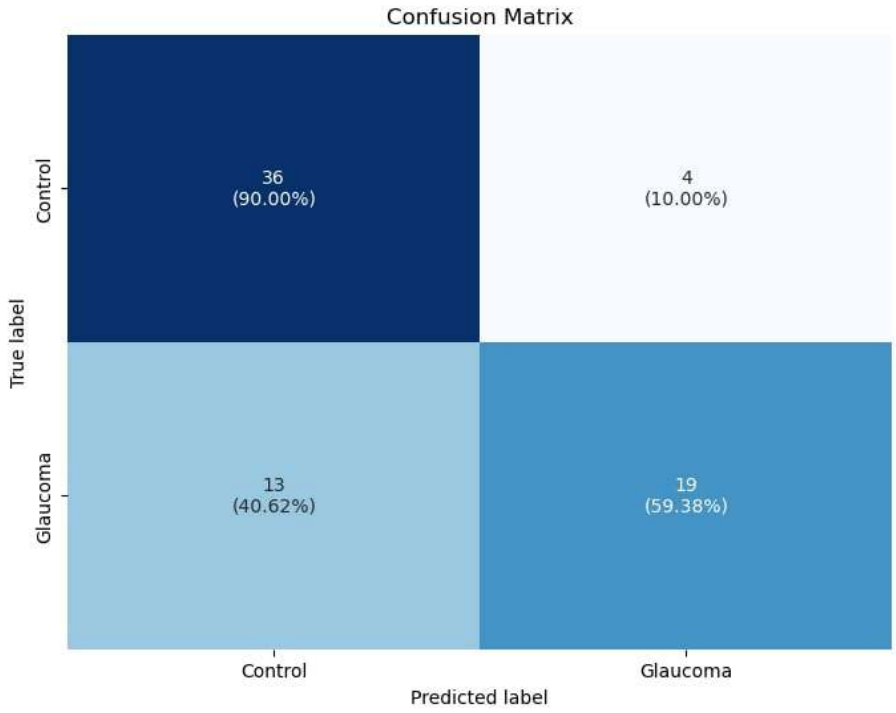

Figura D.85: Matriz de confusión caso selección de respuestas ante estímulos tipo rampa *Filtrado + D.A.*, con ventanas de 100 muestras de largo.

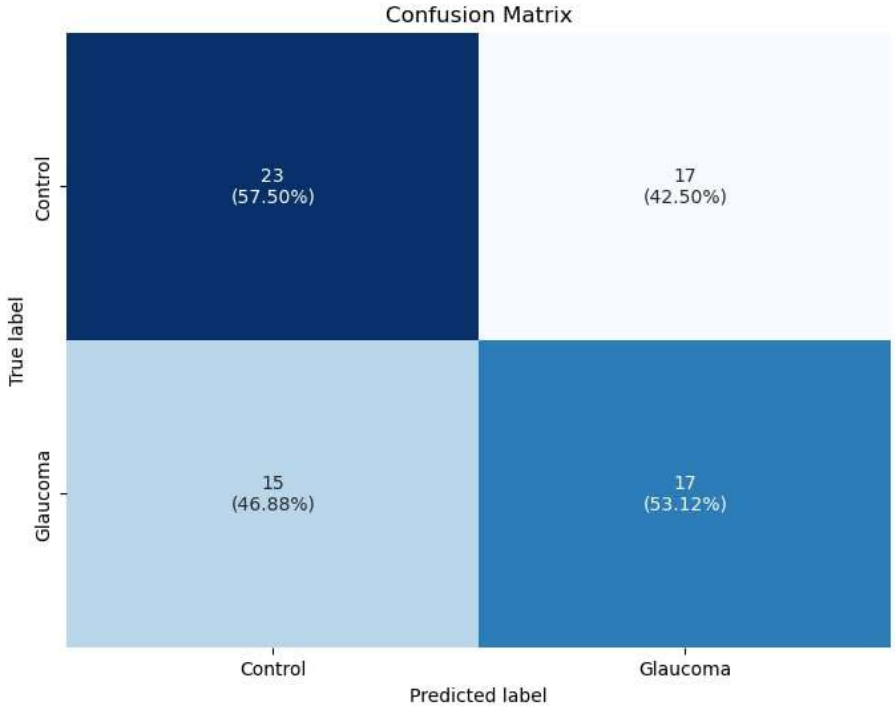

Figura D.86: Matriz de confusión caso selección de respuestas ante estímulos tipo rampa *Filtrado + D.A.*, con ventanas de 25 muestras de largo.

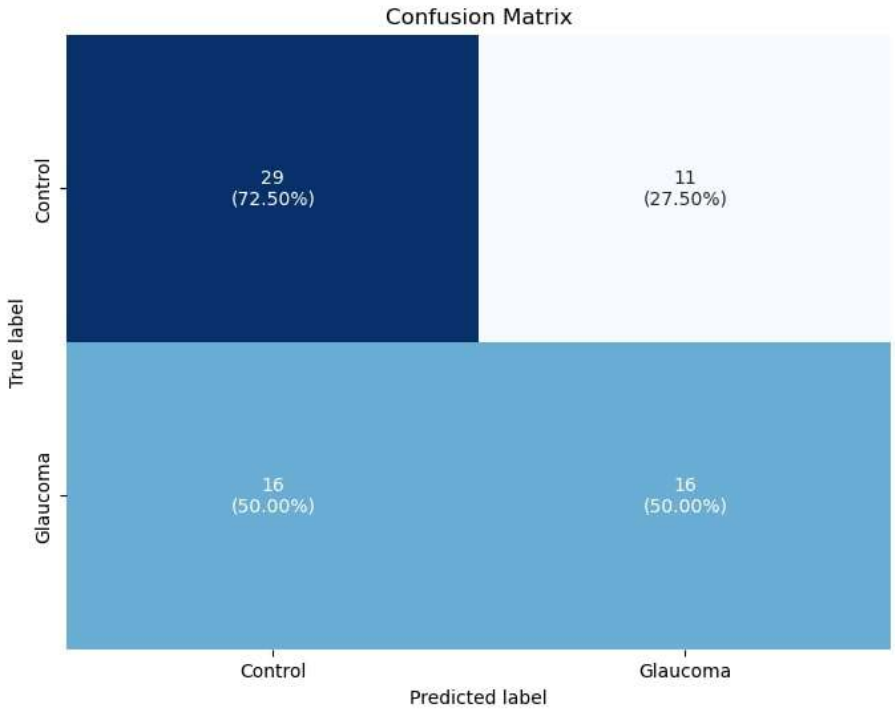

Figura D.87: Matriz de confusión caso selección de respuestas ante estímulos tipo rampa *Filtrado + D.A.*, con ventanas de 50 y 100 muestras de largo.

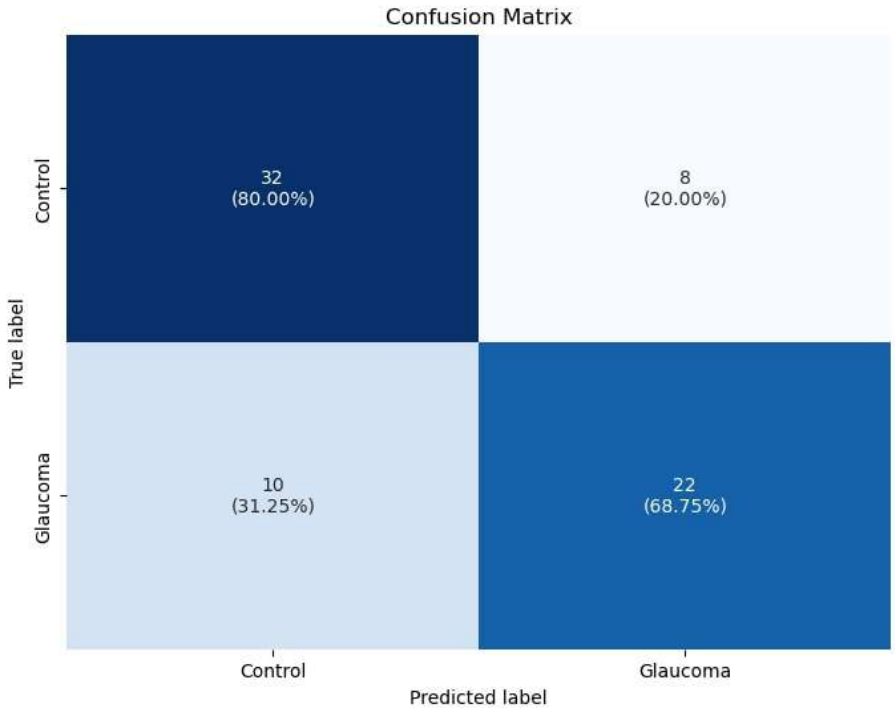

Figura D.88: Matriz de confusión caso selección de respuestas ante estímulos tipo rampa *Filtrado + D.A.*, con ventanas de 150 muestras de largo.