UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MÉXICO

INSTITUTO NACIONAL DE PSIQUIATRÍA RAMÓN DE LA FUENTE MUÑIZ

INSTITUTO NACIONAL DE PSIQUIATRÍA RAMÓN DE LA FUENTE MUÑIZ

FACULTAD DE MEDICINA

Clasificadores de inteligencia artificial para detectar deterioro cognitivo en adultos mayores en reposo, por medio de técnicas fractales

T E S I S

QUE PARA OBTENER EL GRADO DE

MAESTRA EN FÍSICA MÉDICA

P R E S E N T A

Brenda Fernanda Noguez Ruiz

Comité de Tutores: **M. en C. Eleni Mitsoura**

Dra. Erika Elizabeth Rodríguez Torres

Dr. Felipe Humberto Contreras Alcalá

TOLUCA, ESTADO DE MÉXICO 2024

Índice general

Resumen

El Deterioro Cognitivo Leve (DCL) es un síndrome caracterizado por la pérdida de memoria relacionada con la edad, que puede manifestarse a través de problemas de memoria, lenguaje o capacidad de juicio. Aunque México experimenta un aumento anual de casos, aún carece de estudios biológicos sistematizados como los realizados en otros países como Estados Unidos, Canadá y Argentina. En este contexto, la Electroencefalografía (EEG) emerge como una alternativa viable, dado su costo relativamente bajo en comparación con otros métodos como biomarcadores, Tomografía por Emisión de Positrones (TEP) o Resonancia Magnética (RM).

El análisis de la actividad cerebral mediante el Electroencefalograma (EEG) se ha relacionado con diversas patologías. Al emplear técnicas fractales en la evaluación de estas ondas cerebrales, se vislumbra la posibilidad de obtener resultados prometedores para la detección del DCL.

Con el propósito de lograr un diagnóstico temprano y preciso, resulta crucial adoptar nuevas tecnologías, como las técnicas fractales, en combinación con clasificadores de Inteligencia Artificial (IA). En este contexto, la IA desempeña un papel esencial, permitiendo predicciones a partir de datos y aprendizaje automático.

Los clasificadores, tales como árboles de decisión, bosques aleatorios y redes neuronales, son algoritmos que asignan información a clases. Estos se emplean para prever etiquetas de instancias y se están convirtiendo en herramientas cada vez más utilizadas en el análisis de imágenes médicas.

Este estudio se centró en la aplicación de técnicas fractales a los datos obtenidos del EEG en adultos mayores del grupo control y aquellos con DCL en estado de reposo. La comparación se llevó a cabo mediante la selección y entrenamiento de un clasificador de IA, con el objetivo de aportar una nueva perspectiva a esta área de investigación.

Abstract

Mild Cognitive Impairment (MCI) is a syndrome characterized by age-related memory loss that can manifest itself as problems with memory, language, or judgment. Although Mexico is experiencing an annual increase in cases, it still lacks the systematic biological studies conducted in other countries such as the United States, Canada and Argentina. In this context, Electroencephalography (EEG) is emerging as a viable alternative due to its relatively low cost compared to other methods such as biomarkers, Positron Emission Tomography (PET) or Magnetic Resonance Imaging (MRI).

The analysis of brain activity by EEG has been linked to various pathologies. By applying fractal techniques to the evaluation of these brain waves, the possibility of obtaining promising results for the detection of MCI becomes apparent.

In order to achieve early and accurate diagnosis, it is crucial to adopt new technologies such as fractal techniques in combination with Artificial Intelligence (AI) classifiers. In this context, AI plays an essential role by enabling predictions based on data and machine learning.

Classifiers, such as decision trees, random forests, and neural networks, are algorithms that assign information to classes. They are used to predict labels for instances and are increasingly used in medical image analysis.

This study focused on applying fractal techniques to EEG data obtained from older adults in the control group and those with MCI at rest. The comparison was performed by selecting and training an AI classifier with the aim of providing a new perspective to this area of research.

Abreviaturas

DCL / MCI Deterioro Cognitivo Leve / Mild Cognitive Impairment

EEG Electroencefalografía / Electroencefalograma / Electroencephalography

TEP / PET Tomografía por Emisión de Positrones / Positron Emission Tomography

RM / MRI Resonancia Magnética / Magnetic Resonance Imaging

IA / AI Inteligencia Artificial / Artificial Intelligence

ENASEM Encuesta Nacional sobre Salud y Envejecimiento en México

SVM Máquina de Vectores de Soporte / Support Vector Machine

CNN Red Neuronal Convolucional / Convolutional Neural Network

EM Esclerosis Múltiple

ADNI Alzheimer Disease's Neuroimaging Initiative

DFA Análisis de Fluctuación sin Tendencia / Detrended Fluctuation Analysis

CAP Potencial de Acción Compuesto / Compound Action Potential

ECG Electrocardiograma

KNN K-vecinos más cercanos / K-Nearest Neighbors

XGBoost Extreme Gradient Boosting

TC Tomografía Computarizada

EOG Electrooculografía

EMG Electromiografía

PSG Polisomnografía

LOOCV Leave One Out Cross Validation

VP Verdaderos Positivos

FP Falsos Positivos

VN Verdaderos Negativos

FN Falsos Negativos

Ho Hipótesis nula

Ha Hipótesis alterna

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

El DCL es un síndrome neuropsiquiátrico cuyo aumento anual en México resulta alarmante **[36]**. La Encuesta Nacional sobre Salud y Envejecimiento en México (ENASEM) reporta una tasa de incidencia global estimada de 31.4 casos por cada 1,000 personas-año en sujetos con deterioro cognitivo no demencial **[16]**.

Las neuronas son las responsables de mandar información para ejecutar una determinada función a partir de los impulsos eléctricos, mejor conocidos como ondas cerebrales. El EEG es una herramienta de diagnóstico que registra la actividad de las ondas cerebrales. Se puede utilizar para obtener información sobre la frecuencia de las oscilaciones cerebrales **[30, 38]**, que se pueden clasificar en cinco categorías principales: delta (0.1 Hz-3.5 Hz), theta (3.5 Hz-7.5 Hz), alpha (7.5 Hz-12.5 Hz), beta (12.5 Hz-30 Hz) y gamma (30 Hz-100 Hz). Algunas investigaciones sugieren que la evaluación de dichas ondas mediante análisis fractal se han asociado a diferentes patologías **[14, 33, 19]**.

Los fractales son objetos cuya estructura se repite a diferentes escalas, es decir; tienen la propiedad de la autosimilitud. El término fractal se deriva del latín fractus, que significa fracturado o quebrado y se le atribuye al matemático Mandelbrot, quien en 1975 introdujo el término en su obra The fractal geometry of nature **[26]**.

En la actualidad, la Inteligencia Artificial (IA) ha adquirido un papel fundamental en numerosos sectores, gracias a la capacidad de las máquinas para utilizar algoritmos de autoaprendizaje y extraer conocimiento de los datos con el propósito de realizar predicciones **[35]**.

Los clasificadores son algoritmos que asignan información a una clase. Son una subcategoría del aprendizaje supervisado cuyo objetivo radica en predecir las etiquetas de clase categóricas de nuevas instancias, basándose en observaciones previas. Estas etiquetas de clase son valores discretos y no ordenados que representan la pertenencia de las instancias a diferentes grupos **[35]**.

Existen varios métodos de clasificación ampliamente utilizados para asignar etiquetas de clase a instancias. Algunos de estos métodos incluyen los árboles de decisión, bosques aleatorios, Máquinas de Vectores de Soporte (SVM), Naive Bayes y redes neuronales. Cada uno de estos métodos tiene sus propias características y se adapta a diferentes tipos de problemas **[35, 45]**.

Una red neuronal artificial es un procesador distribuido masivamente paralelo compuesto por unidades de procesamiento simples que almacenan conocimiento y lo ponen a disposición para su uso. Modelan la forma en que el cerebro realiza una tarea o función específica de interés **[40]**.

Las Redes Neuronales Convolucionales (CNN), se han utilizado en aplicaciones de imágenes médicas desde la década de 1990 en estructura pulmonar, detección de nódulos y clasificación del tejido mamario **[5, 43]**. En el análisis de imágenes cerebrales, el trabajo pionero apareció en la Esclerosis Múltiple (EM) de Maleki et al. **[5]**. Hoy en día, las arquitecturas de CNN profundas se utilizan ampliamente en la resonancia magnética cerebral para preprocesar datos, detectar y segmentar lesiones como tumores, tejidos completos y estructuras subcorticales **[5]**.

A pesar de que las CNN resultan ser una herramienta eficiente en el tiempo de evaluación de imágenes médicas para diagnóstico asistido por computadora, hoy en día no se cuenta con trabajos de investigación donde se utilicen este tipo de redes neuronales para análisis de imágenes fractales.

Debido a que se ha demostrado la existencia de la relación entre dimensión fractal del EEG y la cognición humana **[47]**, el presente trabajo se centró en la comparación de las dimensiones fractales obtenidas en reposo de adultos mayores mediante las señales del grupo control y DCL a partir de clasificadores IA.

1.1. Planteamiento del problema

En México, el diagnóstico del DCL se realiza a través de una evaluación clínica minuciosa realizada por especialistas. Sin embargo, hasta ahora, no se han llevado a cabo estudios a gran escala como los que se están implementando en otros países a través de la Iniciativa de Neuroimagen de la Enfermedad de Alzheimer (Alzheimer Disease's Neuroimaging Initiative, ADNI).

Aunque sería ideal implementar en México iniciativas similares, se podría considerar el uso de la EEG. Esta técnica resulta relativamente económica y podría ser utilizada para evaluar, detectar y dar seguimiento a pacientes con DCL, lo que la convierte en una alternativa más accesible para el sistema de salud mexicano.

Capítulo 2

Estado del arte

En 1988 el término DCL fue introducido por Reisberg como un proceso degenerativo que antecedía a la demencia **[17]**, un síndrome que implica la perdida parcial o combinada de funciones cognitivas **[50]**. La detección temprana, resulta de gran importancia para proporcionar una planificación del tratamiento adecuada que permita implementar intervenciones farmacológicas y/o no farmacológicas **[5, 43]**.
La instauración discreta del DCL hace difícil su distinción del envejecimiento normal, la depresión, la baja inteligencia previa o la demencia. Esto, sumando que las pruebas de diagnóstico no son precisas y se requiere tiempo para su realización **[12]**.
Algunos neurocientíficos se han dado cuenta de que el trazado del EEG corresponde con un fractal, debido a que el conjunto de datos de señales fluctuantes continuas, representa procesos que no son lineales, estacionarios y tienen características caóticas **[39]**. La geometría fractal resulta ser una herramienta útil para detectar patrones y diferencias en estudios fisiológicos **[13, 33, 19, 47, 37, 34, 31]**. Se ha determinado que los distintos tipos de fibras musculares presentan una estructura fractal mediante la recopilación de datos a partir del músculo de la pata de la rata. Se utilizó una técnica histoquímica (ATP asa básica) para discernir los distintos tipos de fibras e identificar su localización dentro del corte histológico **[37]**.
Otra investigación sugiere la aplicación del Análisis de Fluctuación sin Tendencia (DFA) de Potenciales de Acción Compuestos (CAP) registrados en los nervios cutáneos de ratas diabéticas, donde se realizó un estudio inicial en ratas normales para verificar si las fluctuaciones en el área CAP, de prueba a prueba, mostraban un comportamiento autosimilar o una estructura fractal mediante el DFA y gráficos de Poincaré. Asimismo, se determinó si estas fluctuaciones en el CAP variaban con la inducción de la diabetes **[34]**.
De manera adicional, se ha evidenciado que la aplicación in vitro de péptidos, como la bombesina, estimula la frecuencia respiratoria y aumenta la aparición de suspiros. No

obstante, la distinción entre las formas de onda de suspiros y no suspiros presenta dificultades, lo que complica el estudio de sus propiedades en condiciones experimentales. La utilidad del análisis fractal se ha destacado al caracterizar suspiros y no suspiros, ya que reveló que estos tienen exponentes de Hurst similares y que la aplicación de bombesina solo redujo el exponente de Hurst en los no suspiros **[31]**.

Desde sus inicios, los algoritmos de aprendizaje automático fueron diseñados y utilizados para analizar conjuntos de datos médicos, destacándose especialmente en el ámbito del diagnóstico médico, abordando problemas diagnósticos especializados y de menor escala **[23]**. En el campo de cardiología, se implementó el algoritmo de agrupamiento difuso con el objetivo de reducir el tiempo de procesamiento de las señales del electrocardiograma (ECG), logrando esto mediante la minimización del número de muestras de datos sin perder información esencial para el análisis de las señales cardíacas **[22, 10]**.

El análisis de datos de voz emerge como un aspecto crucial en la presente década, enfocado en la investigación e identificación de técnicas de diagnóstico efectivas para la enfermedad de Parkinson **[22, 42]**. Un estudio propone diagnosticar la enfermedad mediante el análisis de conjuntos de datos de voz de 31 participantes, haciendo uso de algoritmos de aprendizaje automático. En este proceso, se han empleado clasificadores como las SVM y el método de los K-Vecinos más cercanos (KNN) **[22, 42]**.

Un estudio adicional plantea la creación de un modelo capaz de prever la diabetes en una fase temprana con la máxima precisión posible, empleando tres algoritmos de clasificación de aprendizaje automático: Árbol de Decisión, SVM y Naive Bayes. Los resultados obtenidos indican que Naive Bayes supera, logrando una precisión del 0.76 **[22, 41]**.

En el ámbito de la física médica, los clasificadores de inteligencia artificial se han empleado, entre otras aplicaciones, para la detección de pólipos en el colon mediante colonografía por Tomografía Computarizada (TC), utilizando redes neuronales y árboles binarios recursivos **[21]**.

Asimismo, en el sector salud, los recientes avances en técnicas de inteligencia artificial han propiciado aplicaciones exitosas de la IA **[23, 22]**. Las diversas técnicas disponibles pueden ser utilizadas para extraer información relevante de extensas cantidades de datos clínicos **[23, 22]**. De igual forma, los métodos de inteligencia artificial son entrenados de manera que poseen la capacidad de aprender de forma autónoma, corregir errores y generar resultados con una alta precisión **[22]**.

Por lo previamente mencionado, la utilización de clasificadores IA en el análisis fractal puede resultar ser una herramienta que facilite la detección temprana del DCL en un período de tiempo más reducido. En este sentido, el objetivo es identificar un clasificador

IA capaz de llevar a cabo el análisis fractal de participantes, diferenciando entre aquellos con y sin deterioro cognitivo. Este clasificador se propone como una herramienta de diagnóstico médico después de ser entrenado adecuadamente.

2.1. Pregunta de investigación

¿Es posible identificar Deterioro Cognitivo Leve analizando el patrón de ondas cerebrales a partir de imágenes fractales en reposo de adultos mayores mediante clasificadores de Inteligencia Artificial?

2.2. Hipótesis

Es posible identificar Deterioro Cognitivo Leve en el análisis de imágenes fractales en reposo de adultos mayores mediante clasificadores de Inteligencia Artificial.

2.3. Objetivo general

Seleccionar y entrenar un clasificador IA para identificar DCL a partir del análisis de imágenes fractales obtenidas del patrón de ondas cerebrales de adultos mayores en reposo.

2.4. Objetivos específicos

- Recopilar y preparar el conjunto de datos obtenidos por el EEG de adultos mayores control y DCL en reposo.
- Seleccionar un algoritmo de clasificación de IA para analizar las imágenes fractales en reposo de adultos mayores control y DCL.
- Entrenar, verificar y mejorar el clasificador IA.
- Comparar las imágenes fractales y determinar las diferencias entre adultos mayores controles contra DCL.

Capítulo 3

Marco Teórico

El DCL se define como la reducción de las funciones cognitivas y puede estar asociado con alteraciones cerebrales que se producen en las etapas iniciales del Alzheimer u otras enfermedades neurodegenerativas que conllevan a la demencia **[4, 28]**. Por lo tanto, es crucial realizar una evaluación integral adecuada del paciente para detectar y diagnosticar esta afección en sus etapas tempranas.

El EEG es un registro que mide las señales eléctricas generadas por las células cerebrales, o más específicamente, el curso temporal de los potenciales de campo extracelular producidos por su acción sincrónica **[6]**. Estas señales se registran mediante electrodos colocados en el cuero cabelludo o directamente en la corteza cerebral. Recientemente, el uso del EEG ha demostrado ser una herramienta prometedora en la clasificación de la actividad cerebral, especialmente en lo que respecta a trastornos cerebrales **[29, 47, 36, 38]**. Esto se debe a que las señales EEG, al ser caóticas y constituir series de tiempo no lineales, son susceptibles de ser analizadas utilizando técnicas fractales **[29, 2]**. Estas técnicas permiten examinar las características intrínsecas de las señales EEG, lo que puede proporcionar información valiosa para el diagnóstico y seguimiento de condiciones como el DCL.

La geometría fractal se dedica a investigar y analizar los aspectos geométricos que mantienen su configuración y características fundamentales a través de distintas escalas de observación **[13, 26]**. A partir de estos análisis, es posible determinar la dimensión fractal, cuyo propósito es cuantificar la complejidad y densidad de un fractal dentro del espacio métrico en el que se desarrolla **[3]**. Esta dimensión fractal ofrece un enfoque objetivo para comparar y clasificar diversos tipos de fractales, considerando su nivel de complejidad y detalle geométrico mediante el uso del aprendizaje automático. Este último nos permite extraer conocimiento de los datos mediante la aplicación de diversos algoritmos **[25]**. La elección del algoritmo adecuado depende del tipo de problema a resolver, el número de variables, el tipo de modelo que mejor se adapte, entre otros factores **[25]**.

Capítulo 4

Metodología

La metodología se compuso de tres partes fundamentales: el enfoque biológico, que detalla cada procedimiento para la obtención de las series de tiempo registradas en adultos mayores durante la vigilia en reposo; el enfoque matemático, que se encarga del análisis de estas series; y el enfoque computacional, donde se describe en detalle cada uno de los clasificadores de inteligencia artificial utilizados.

4.1. Método biológico

Se observó el funcionamiento eléctrico del cerebro de 18 participantes (adultos mayores) mediante un estudio encefalográfico en reposo, sujeto despierto con ojos cerrados durante 5 minutos, en el cual se les colocaron 19 electrodos de acuerdo con el sistema 10/20 internacional. El 10/20 se deriva de la distancia entre los puntos de referencia que se utilizan para marcar la posición de los electrodos, que corresponde al 10 % o al 20 % de la distancia entre puntos anatómicos clave en la cabeza, como el nasión (el punto en la parte superior de la nariz) y el inión (el punto en la parte posterior de la cabeza). El lóbulo cerebral se representa por letras (Tabla 4.1) y la ubicación del hemisferio se identifica mediante números, como se puede observar en la Figura 4.1.

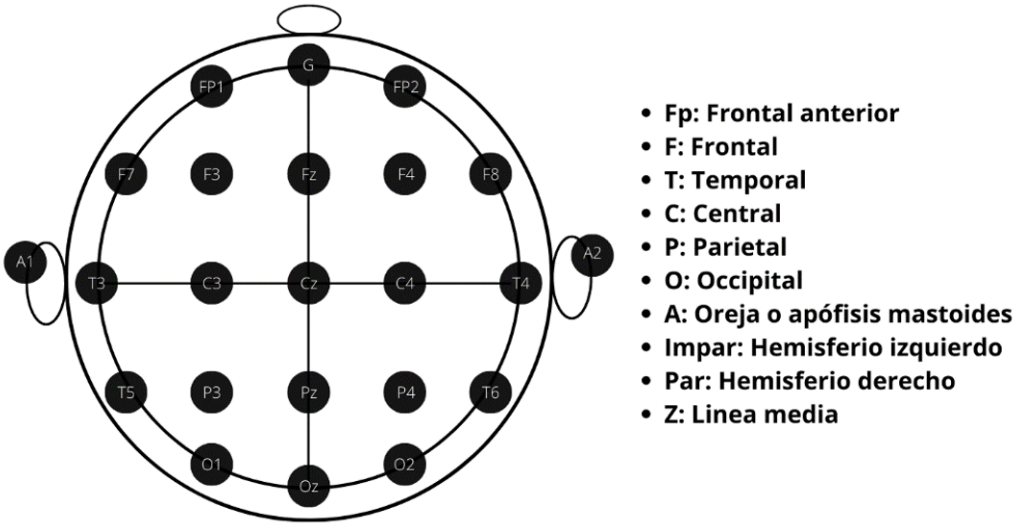


Figura 4.1: Sistema de posicionamiento del EEG.

Electrodo	Lóbulo
F	Frontal
T	Temporal
C	Central
P	Parietal
O	Occipital
FP	Frontal anterior

Tabla 4.1: Lóbulos cerebrales sobre los que se coloca el electrodo.

Se incluyeron 6 electrodos adicionales: 2 de Electrooculografía (EOG) para registrar movimientos oculares horizontales (LOG) y verticales (ROG), 2 en los lóbulos de las orejas, y 2 de Electromiografía (EMG) colocados en el músculo cuadrado del mentón para registrar la actividad muscular. Estos electrodos se muestran en la Figuras 4.1 y 4.2.
En relación con la distribución topográfica, según lo indicado en la Figura 4.1, las áreas cerebrales G y Oz no fueron equipados con electrodos, conforme al protocolo de Polisomnografía (PSG) establecido por los especialistas.

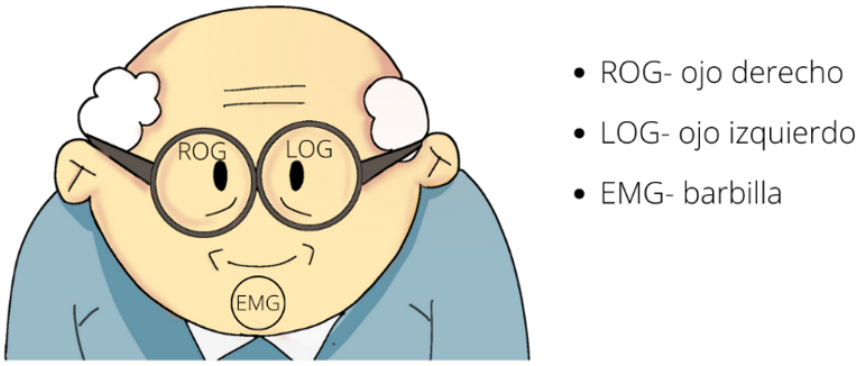


Figura 4.2: Electrodos en ojos y barbilla usados como marcadores de movimientos oculares y actividad muscular en el EEG.

Se registraron las ondas cerebrales de cada participante y se dividieron en dos grupos. El primer grupo, que actuó como control, estaba compuesto por 10 individuos. El segundo grupo consistía en 8 participantes diagnosticados con DCL mediante los exámenes Neuropsi y Mini Mental State Examination. La interpretación de los resultados de estos exámenes estuvo a cargo de expertos, incluyendo a la Psicóloga Génesis Vázquez Tagle Gallegos de la Universidad Autónoma del Estado de Hidalgo (UAEH) y la Dra. Alejandra Rosales Lagarde, de las Cátedras CONACyT, adscrita al Instituto Nacional de Psiquiatría Ramón de la Fuente Muñiz. Se utilizó un equipo MEDICID-5 equipado con 26 amplificadores para llevar a cabo el registro EEG.

Se procesó cada registro con el objetivo de eliminar las áreas afectadas por movimientos, ya sean voluntarios o involuntarios por parte del sujeto. Esta acción permitió la posterior aplicación de un filtro de banda, que posibilitó el paso de frecuencias comprendidas entre 0.5 Hz y 40 Hz, mientras que atenuaba todas las demás.

Los filtros se establecieron entre 0.1-100 Hz para el EEG, 10-70 Hz para el EMG y 0.3-15 Hz para el EOG. Se incorporó un filtro Notch para eliminar la frecuencia de 60 Hz, asociada al ruido generado por la corriente eléctrica, con el fin de alcanzar un nivel superior de estabilidad en los datos. La impedancia se mantuvo por debajo de 10 kΩ. Los datos fueron digitalizados a frecuencias de muestreo de 256 Hz, 200 Hz y 512 Hz, respectivamente, mediante un convertidor A/D de 16 bits. Posteriormente, se almacenaron en el sistema MEDICID-5 con el fin de facilitar su extracción posterior en formato .txt.

4.2. Método matemático

4.2.1. Juego del Caos Circular

El Juego del Caos, concebido por Barnsley, tiene como objetivo la creación de fractales mediante valores aleatorios, como el reconocido Triángulo de Sierpinski **[3]**. Para este juego de cero jugadores, Barnsley propone un algoritmo en el que se dispone un conjunto de m puntos, denominados vértices de la figura geométrica:

$$\text{Conjunto de vértices} : v_0, v_1, \ldots, v_{m-1} \tag{4.1}$$

donde m es un número predeterminado.
Partiendo de un punto al azar x_0 en el plano, el juego crea una sucesión de n puntos x_1, x_2, …, x_n a partir del conjunto de vértices.
Asimismo, seleccionamos n valores al azar entre 0 y $m-1$ que llamaremos r_i, para $i = 0, 1, \ldots, n-1$, para construir la sucesión de puntos de la siguiente manera:

$$\begin{aligned} x_1 &= \frac{v_{r_1} + x_0}{2} \\ x_2 &= \frac{v_{r_2} + x_1}{2} \\ &\vdots \\ x_i &= \frac{v_{r_i} + x_{i-1}}{2} \\ &\vdots \\ x_n &= \frac{v_{r_n} + x_{n-1}}{2} \end{aligned} \tag{4.2}$$

Los números aleatorios se eligen siguiendo una distribución uniforme, y la sucesión resultante de puntos $(x_0, x_1, x_2, \ldots, x_n)$ se denomina órbita de este proceso. Si tras un gran número de puntos, estos se aproximan hacia una figura específica, esta figura se considerará su atractor. El juego original utiliza un parámetro $m = 3$ para generar el Triángulo de Sierpinski (Figura 4.3). No obstante, Tsuchiya et al. **[46]** optaron por emplear un polígono regular en lugar del triángulo mencionado anteriormente, lo que dio lugar a la creación de otros tipos de fractales. Estos juegos, conocidos como Juegos de Caos Circular o IFSOAC

(sistemas de funciones iteradas en un círculo), han demostrado ser eficaces en el análisis de Series de Tiempo **[32]**, las cuales pueden definirse como una sucesión de valores $S = (s_1, s_2, \ldots, s_n)$ representando observaciones de algún sistema a intervalos regulares de tiempo. En este procedimiento, en lugar de elegir r_i al azar; mediante una función f, transformamos los datos de la serie en índices, para encontrar el vértice con el que debemos promediar los puntos para la orbita. Esto se logra de la siguiente manera:

$$r_i = f(s_i) \tag{4.3}$$

$f(s_i)$ es un valor del conjunto $\{0, 1, 2, \ldots, m-1\}$ proporcional, o de alguna forma relacionado, al valor de s_i.

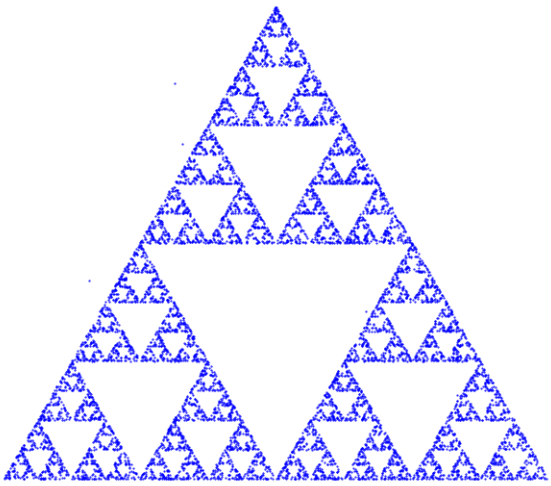
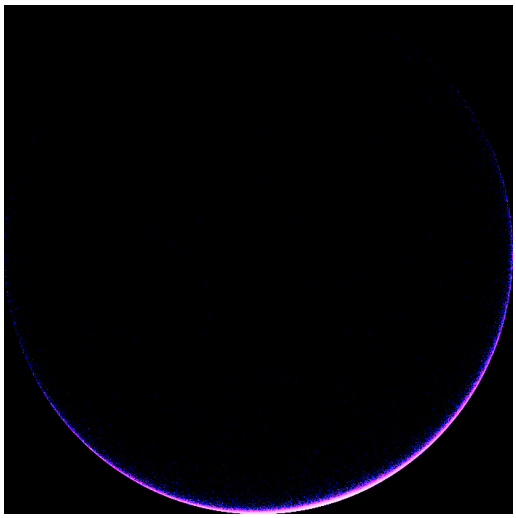
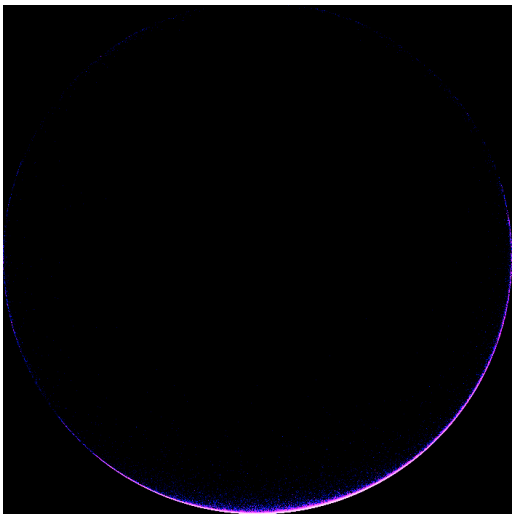


Figura 4.3: Triángulo de Sierpinski para 10000 puntos.

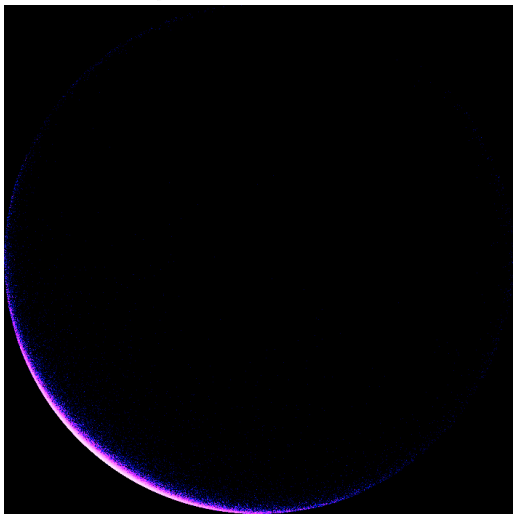
Dichos valores para r_i elegirán los vértices reflejando características de los valores de la serie de tiempo y cuando m es lo suficientemente grande, al menos tan grande como el número de valores distintos en la serie temporal, cada punto en la serie se representa en una posición angular dentro de un polígono regular de m lados. En nuestro caso optamos por utilizar una $m = 500$ con el fin de generar un atractor que adquiera una forma circular (Figura 4.4) **[9]**. Conforme m aumenta significativamente, la distribución de puntos proyectará una estructura que se asemejará a la de una rosquilla.



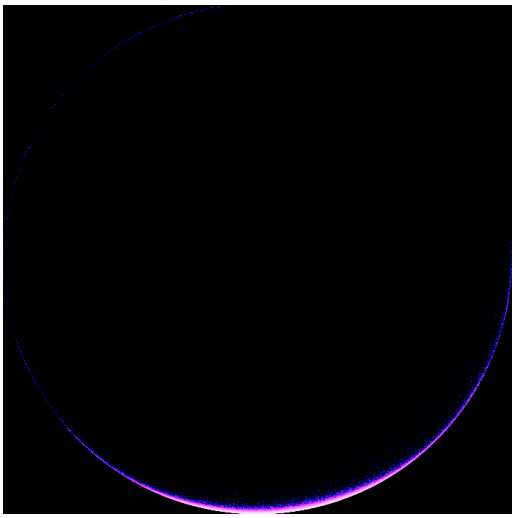
a Participante con DCL, lóbulo frontal 7.



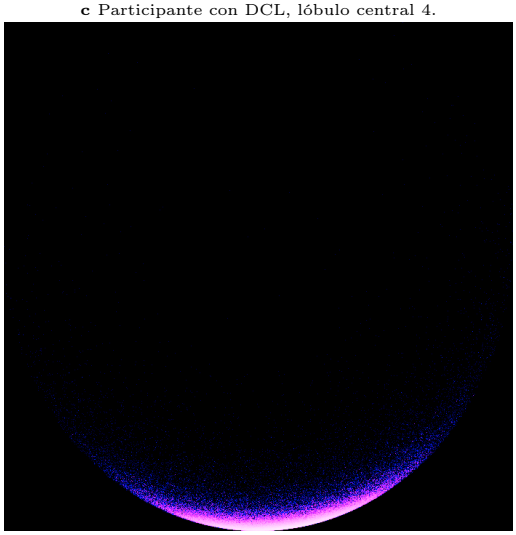
b Participante Control, lóbulo frontal 7.



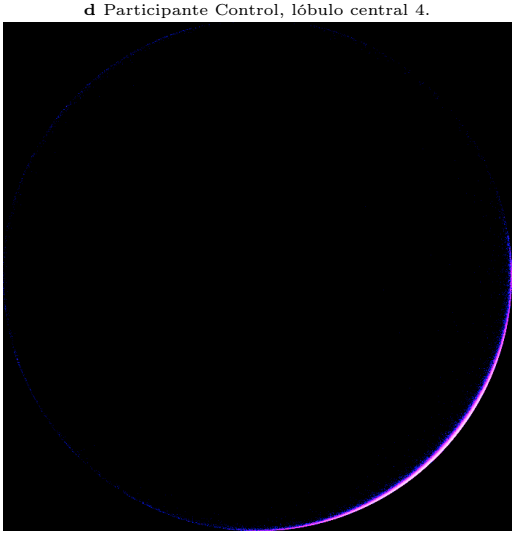
c Participante con DCL, lóbulo central 4.



d Participante Control, lóbulo central 4.



e Participante con DCL, lóbulo occipital 2.



f Participante Control, lóbulo occipital 2.

Figura 4.4: Comparación de imágenes fractales de dos participantes para diferentes áreas cerebrales.

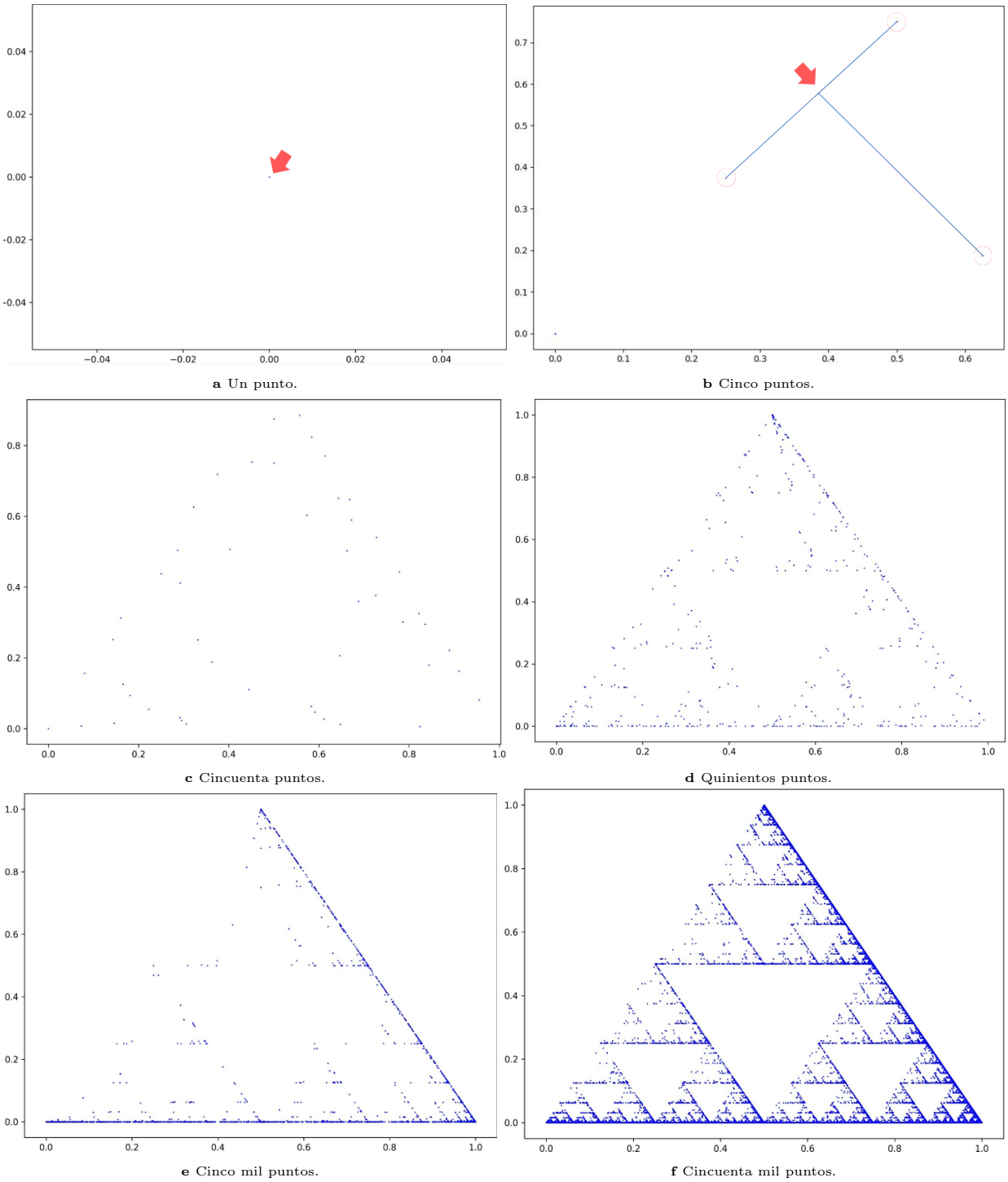
Finalmente, podemos definir la construcción del Juego del Caos para m vértices de la siguiente manera:

Definición 4.2.1. El Juego de Caos Circular para m vértices consiste en el conjunto $V = \{v_0, v_1, \ldots, v_{m-1}\}$ y un punto inicial al azar x_0; tras lo cual se construirán los valores x_i de una órbita con la fórmula:

$$x_i = \frac{v_{r_i} + x_{i-1}}{2}$$

donde $(r_i : i = 1, 2, 3, \ldots)$ es una sucesión de valores al azar en el conjunto $R = \{0, 1, 2, \ldots, m-1\}$ **[3]**

A continuación, en la Figura 4.5, se da un ejemplo de cómo se genera un Triángulo de Sierpinski a partir de una serie de tiempo.



a Un punto.

b Cinco puntos.

c Cincuenta puntos.

d Quinientos puntos.

e Cinco mil puntos.

f Cincuenta mil puntos.

Figura 4.5: Generación de un Triángulo de Sierpinski a partir de una serie de tiempo.

4.2.2. Dimensión fractal

En 1975, el matemático Benoit B. Mandelbrot introdujo el concepto de fractal en su obra *The fractal geometry of nature* **[26]**. El término se deriva del latín *fractus*, que significa roto, quebrado o partido. Se define como un conjunto de patrones que se repiten a diferentes escalas, poseen la propiedad de la autosimilitud, no son diferenciables (no se puede trazar una tangente con inclinación única a una curva fractal) y exhiben una dimensión fraccional **[44]**.
La dimensión fractal es un parámetro crucial para comprender la extensión métrica de un fractal y para compararlo con otros. Sin embargo, su cálculo puede realizarse de diversas maneras, entre las cuales se incluyen el conteo de cajas, la dimensión de brújula, la dimensión de auto-similaridad, entre otras. Todas estas metodologías son variantes especializadas de la dimensión fractal original propuesta por Mandelbrot, cuya inspiración se remonta al trabajo pionero de Hausdorff en 1919 **[32]**. La siguiente definición se utiliza para determinar si un conjunto A posee una dimensión fractal D:

Definición 4.2.2. Sea $A \in H(X)$ un subconjunto compacto no vacío de X, donde (X, d) es un espacio métrico. Para cada $\varepsilon > 0$, sea $N(A, \varepsilon)$ el menor número de bolas cerradas de radio $\varepsilon > 0$ necesarias para cubrir A. Si

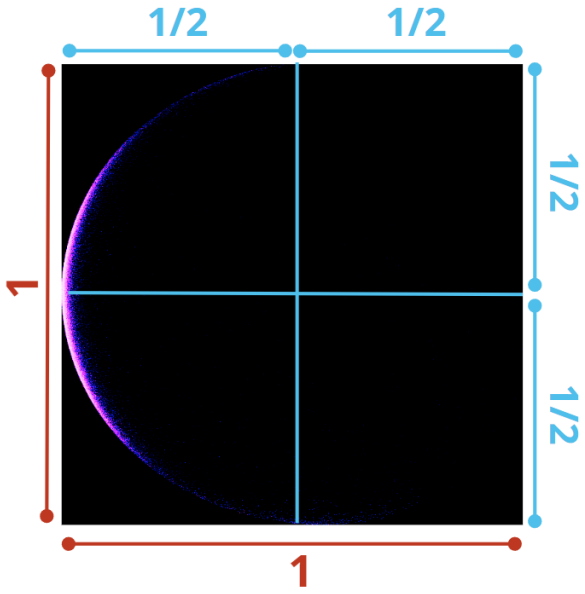
$$D = \lim_{\varepsilon \to 0} \left\{ \frac{\ln(N(A, \varepsilon))}{\ln\left(\frac{1}{\varepsilon}\right)} \right\}$$

si existe, entonces D es la dimensión fractal de A. También utilizaremos la notación $D = D(A)$ y diremos A tiene dimensión fractal D **[3]**.

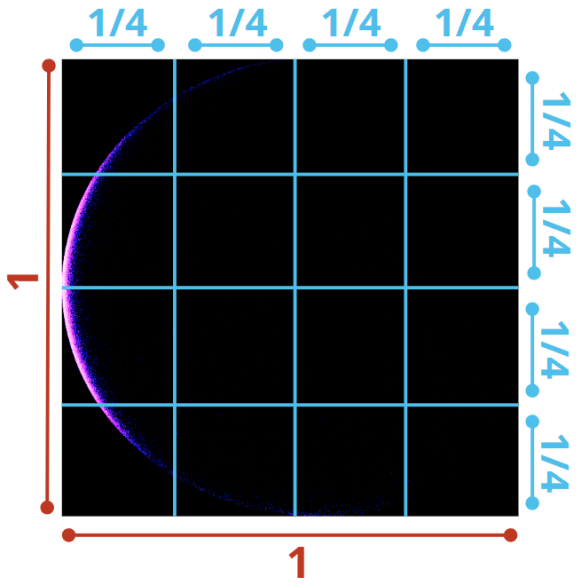
En la práctica, la dimensión fractal se calcula mediante el método de conteo de cajas, el cual goza de amplio reconocimiento en la comunidad científica debido a su versatilidad para aplicarse a una variedad de estructuras. Este método proporciona una medición sistemática y accesible, lo que lo hace especialmente útil para el estudio de fenómenos complejos en diferentes campos de la ciencia.
La figura generada a través de nuestro algoritmo se divide en niveles de cuadrícula, representados por el tamaño ε, que se reduce progresivamente con el objetivo de contar el número de cajas N que contienen al menos un punto del fractal. Este número N depende del tamaño ε, lo cual expresamos como $N(\varepsilon)$. Al modificar ε a tamaños progresivamente más pequeños y contar las cajas correspondientes $N(\varepsilon)$ (hasta llegar a un conteo cons-

tante), procedemos a crear la gráfica logarítmica de $\ln(N(\varepsilon))$ versus $\ln\left(\frac{1}{\varepsilon}\right)$. Ajustamos una línea recta a los puntos graficados y obtenemos su pendiente D_b, que resulta ser la dimensión fractal de conteo de cajas.

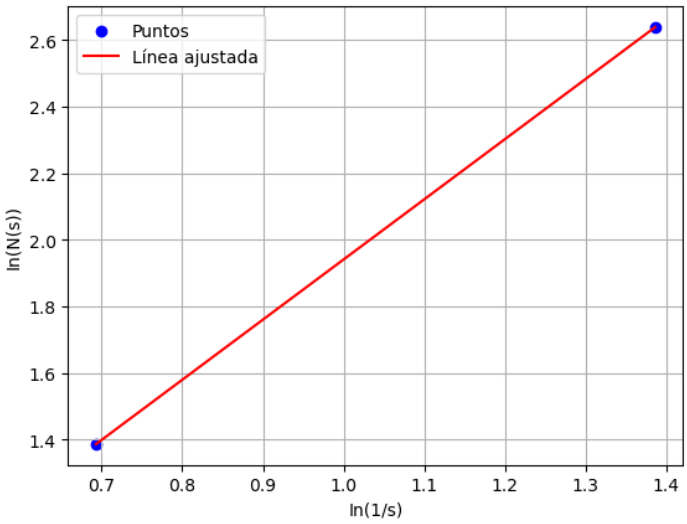
En la Figura 4.6 se muestra un ejemplo de lo mencionado anteriormente. A partir del fractal obtenido, se realiza una división en dos niveles de cuadrícula (con $\varepsilon = \frac{1}{2}$ y $\varepsilon = \frac{1}{4}$). Al realizar el conteo de cajas, encontramos una pendiente D_b aproximada de 1.81.



a $\varepsilon = \frac{1}{2}$ con $N(\varepsilon) = 4$.



b $\varepsilon = \frac{1}{4}$ con $N(\varepsilon) = 14$.



c Ajuste de una línea recta a los puntos, con $D_b \approx 1.81$.

Figura 4.6: Ejemplo de conteo de cajas utilizando una división en dos niveles de cuadrícula.

Es útil considerar una secuencia de cuadrículas en las cuales el tamaño se reduce a la mitad. Al realizar el conteo de cajas del fractal utilizando esta cuadrícula, obtenemos una secuencia de conteos $N(2^{-k})$, donde $k = 0, 1, 2, ...$
Finalmente, podemos establecer el teorema de conteo de cajas de la siguiente manera:

Teorema 4.2.1 (**Conteo de cajas**)**.** Sea $A \in H(\mathbb{R}^m)$, donde se utiliza la métrica Euclidiana. Se procede a cubrir $\mathbb{R}^m$ con cajas cuadradas cerradas de longitud lateral $\frac{1}{2^n}$, tal como se ilustra en la Figura 4.7 para $n = 2$ y $m = 2$. Definimos $N_n(A)$ como el número de cajas de longitud lateral $\frac{1}{2^n}$ que intersectan el atractor. Si

$$D = \lim_{n\to\infty} \left\{ \frac{\ln(N_n(A))}{\ln(2^n)} \right\}$$

entonces, A tiene dimensión fractal D **[3]**.

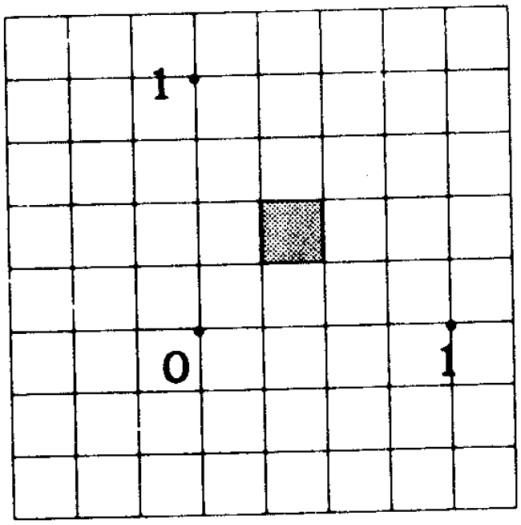


Figura 4.7: Las cajas cerradas de lado $\frac{1}{2^n}$ cubren el plano $\mathbb{R}^2$. En este caso, $n = 2$ **[3]**.

4.3. Método computacional

Una vez calculada la dimensión fractal de cada serie de tiempo de las áreas cerebrales de los participantes, se llevó a cabo una limpieza de datos. A partir de esta limpieza, se generó un archivo CSV que incluía dichos valores junto con la identificación de los adultos mayores (véase Tabla 4.2), categorizados por su edad, sexo, nivel educativo (en años), puntuación neuropsicológica (neuropsi) y grupo al que pertenecían (DCL o control). Estos datos se emplearon en el proceso de entrenamiento del clasificador IA.

Nombre	**Sexo**	**Edad**	**Escolaridad**	**Neuropsi**	**Grupo**
VC	F	59	12	107	Control
MJ	F	72	9	113	Control
JANA	F	78	5	102	Control
GH	M	65	9	107.5	Control
MG	F	61	9	114	Control
EM	F	50	22	117	Control
JALO	M	71	18	113	Control
MMA	F	61	13.5	106	Control
RAS	M	64	16	114	Control
GU	F	67	11	110	Control
CL	F	68	5	81	DCL
RL	F	63	9	90	DCL
RR	M	69	9	85	DCL
JG	M	65	11	87	DCL
AEFP	M	73	8	96	DCL
FG	F	71	9	83.5	DCL
PCM	M	71	9	111	DCL
LI	F	58	12	30	DCL

Tabla 4.2: Identificación de los participantes.

Para la codificación de los grupos y el sexo, se asignaron los números 1 y 0 para permitir que el clasificador distinguiera entre ambos grupos. Específicamente, se asignó el valor 1 a los participantes con DCL y género femenino, mientras que a los participantes del grupo control y género masculino se les asignó el valor 0.

El Neuropsi evalúa diversas funciones cognitivas y es un indicador del envejecimiento normal y patológico. En el participante PCM, a pesar de tener un puntaje Neuropsi de 111, es considerado con DCL por los especialistas debido a problemas subjetivos de memoria y una función cognitiva con tres desviaciones estándar.

4.3.1. Índice de desequilibrio

Se verificó la proporción entre las clases positivas y negativas en el conjunto de datos de entrenamiento mediante el uso del índice de desequilibrio, con el propósito de cuantificar y comparar la homogeneidad de la distribución de los datos disponibles.
El índice de desequilibrio se define como:

$$IR = \frac{N_{ma}}{N_{mi}} \tag{4.4}$$

donde N_{ma} representa el tamaño de muestra de la clase mayoritaria y N_{mi}, el de la clase minoritaria. Cuando $IR = 1$, el conjunto de datos está perfectamente equilibrado. Sin embargo, cuando $IR > 1$, indica un desequilibrio en el conjunto de datos, y cuanto mayor sea el valor de IR, mayor será la magnitud del desequilibrio. Se espera que los conjuntos de datos con un IR más alto sean más difíciles de clasificar **[49]**.
En nuestro caso, contamos con un total de 8 participantes con DCL y 10 en el grupo control, lo que nos proporcionó un índice de desequilibrio de:

$$IR = \frac{10}{8} = 1.25 \tag{4.5}$$

Aunque el índice de desequilibrio sea mayor que uno, no se considera significativo para la clasificación debido a que su magnitud no es considerablemente alta.
Posteriormente, se realizó un segundo cálculo tras ampliar los datos con el fin de mejorar el entrenamiento del clasificador, el cual se detalla en la sección 4.3.2. En este escenario, se incluyeron 48 registros de participantes del grupo de control y 40 de DCL.

$$IR = \frac{48}{40} = 1.2 \tag{4.6}$$

4.3.2. Incremento de datos

Como se mencionó previamente en el método biológico, para la clasificación de los datos se tomaron los primeros 5 minutos del registro EEG de cada área cerebral (Figura 4.8), los cuales representan la parte del estudio en reposo. Para recortar el segmento, se multiplicó la frecuencia por 300 segundos para obtener el número de puntos, determinando así el punto exacto donde se realizaría el corte en la señal.
Para aumentar los datos, se extrajeron segmentos de 1 minuto de los 5 minutos de registro EEG de cada área cerebral. Esto se logró multiplicando la frecuencia por 60 segundos.
A continuación, se detallan los puntos de corte para diferentes frecuencias de muestreo:

- Para 512 Hz, los puntos de corte fueron 153600 para 300 segundos y 30720 para 60 segundos.
- Para 256 Hz, los puntos de corte fueron 76800 para 300 segundos y 15360 para 60 segundos.
- Para 200 Hz, los puntos de corte fueron 60000 para 300 segundos y 12000 para 60 segundos.

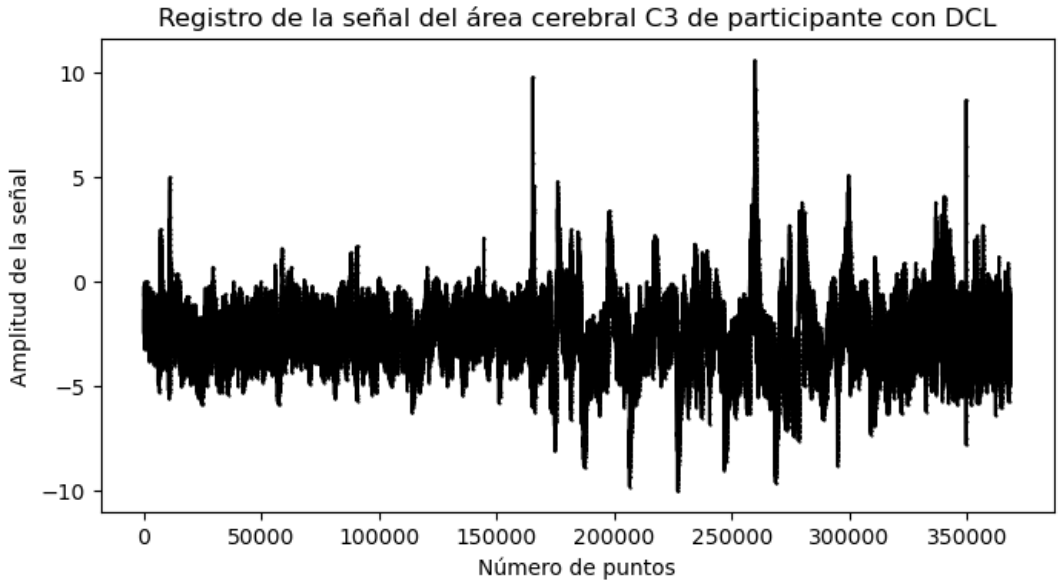


Figura 4.8: Gráfico de los datos obtenidos del área cerebral C3 en reposo de un adulto mayor con DCL. La amplitud de la señal se refiere al cambio de voltaje μV.

4.3.3. Selección del clasificador IA

Se seleccionaron cuatro clasificadores IA a partir de la revisión bibliográfica para el entrenamiento de los datos. Entre ellos se incluyen Extreme Gradient Boosting (XGBoost), Árbol de Decisión, Máquinas de Vectores de Soporte (SVM) y K-vecinos más cercanos (KNN) con un rango de 2 a 13 vecinos más cercanos. Cada uno de estos clasificadores fue evaluado exhaustivamente, lo que permitió seleccionar aquel que mejor se ajustaba a los datos disponibles. A continuación, se presenta una breve descripción del funcionamiento de cada uno.

1. **Extreme Gradient Boosting (XGBoost):** Algoritmo de aprendizaje automático que combina árboles de decisión de forma aditiva para mejorar la precisión de las predicciones. Cada árbol se construye de manera secuencial, corrigiendo las deficiencias del anterior. El algoritmo utiliza el método de gradient boosting durante el entrenamiento para optimizar el rendimiento del modelo, minimizando una función de pérdida diferenciable que evalúa el error del modelo y guía el proceso de aprendizaje hacia mejores predicciones (Figura 4.9) **[27]**.

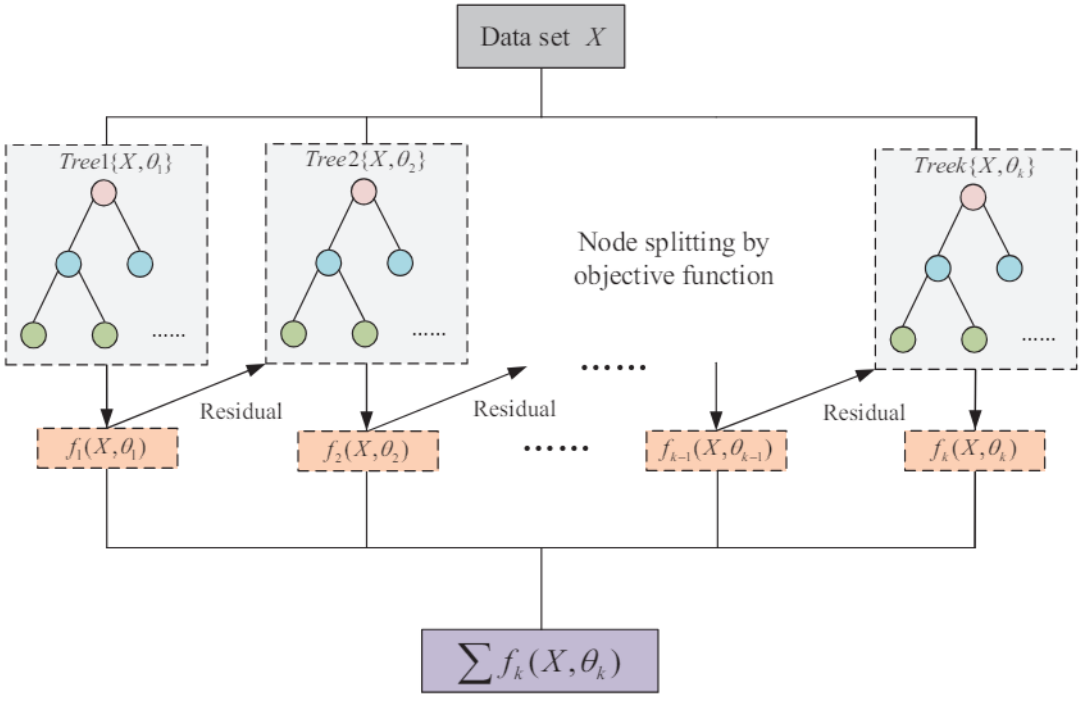


Figura 4.9: Diagrama de flujo XGBoos **[15]**.

2. **Árbol de decisión:** El algoritmo de árbol de decisión utiliza una estructura jerárquica de nodos para la predicción de la clase objetivo, basada en reglas de decisión derivadas de las características de entrada. Los nodos raíz clasifican las instancias con diferentes características, mientras que los nodos hoja representan la decisión o predicción final. En cada etapa, el árbol de decisión selecciona el nodo evaluando la mayor ganancia de información entre todos los atributos (Figura 4.10) **[41, 25]**.

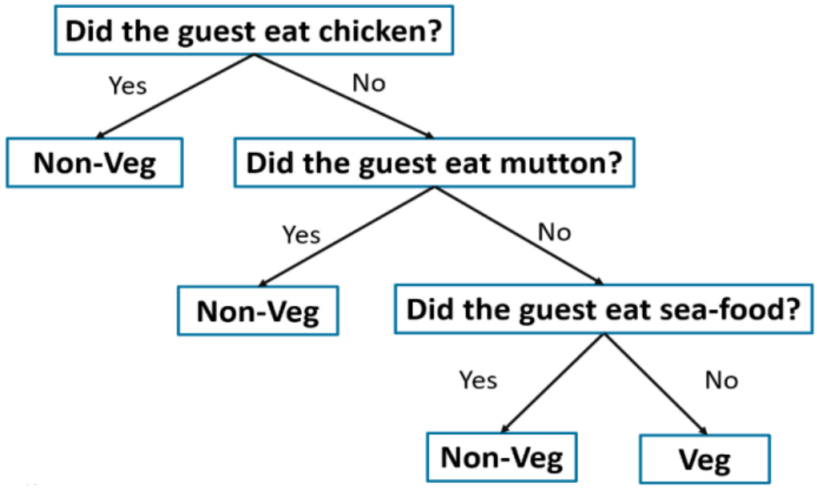


Figura 4.10: Ejemplo árbol de decisión **[25]**.

3. **Máquina de vectores de soporte (Support Vector Machine (SVM)):** Dada una muestra de entrenamiento con dos clases, el objetivo de una máquina de vectores de soporte es encontrar el hiperplano separador óptimo que maximice el margen entre las dos clases. Esto se logra al maximizar la separación entre los ejemplos positivos y negativos, minimizando así el error de clasificación (Figura 4.11). Para una mejor generalización, el hiperplano no debe estar cerca de los puntos de datos de la clase opuesta. Es fundamental seleccionar un hiperplano que esté lo más alejado posible de los puntos de datos de cada categoría. Los puntos que se encuentran más cerca del margen del clasificador se denominan vectores de soporte **[41, 40]**.

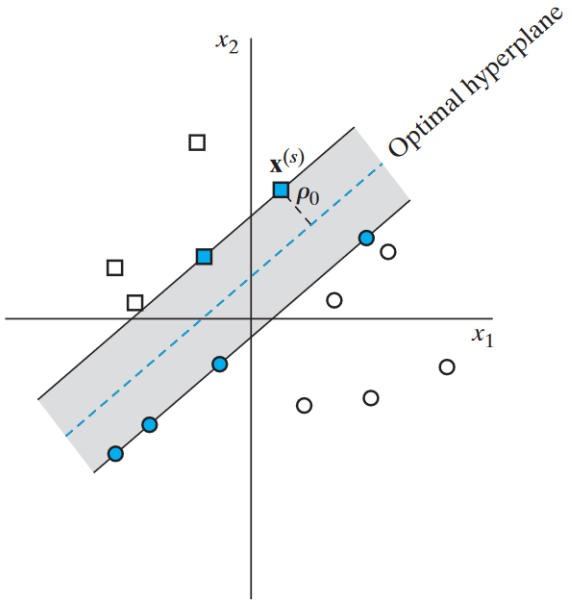


Figura 4.11: Ejemplo de hiperplano óptimo para patrones linealmente separables **[41, 40]**.

4. **K-vecinos más cercanos (K-Nearest Neighbors (KNN)):** El algoritmo encuentra los k puntos más cercanos en el conjunto de datos de entrenamiento al punto que se desea predecir, utilizando una métrica de distancia. Posteriormente, asigna la etiqueta de clase del nuevo punto de datos mediante votación mayoritaria entre sus k vecinos más cercanos (Figura 4.12).
   KNN se puede utilizar tanto para problemas de clasificación como de regresión. La elección adecuada de k es fundamental para lograr un buen equilibrio entre sobreajuste y subajuste **[25, 35]**.

   Igualmente, es fundamental elegir una métrica de distancia adecuada para las características del conjunto de datos. Entre las opciones disponibles se incluyen **[1, 8]**:

   - **Euclideana:** Conocida como norma L2, esta distancia es una extensión del Teorema de Pitágoras. Se calcula como la raíz cuadrada de la suma de los cuadrados de las diferencias entre los valores correspondientes en los vectores:

$$ED(x, y) = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2} \tag{4.7}$$

   - **Manhattan:** También conocida como norma L1, norma Taxicab, distancia rectilínea o distancia city block, esta métrica representa la suma de las diferencias absolutas entre los valores opuestos en los vectores:

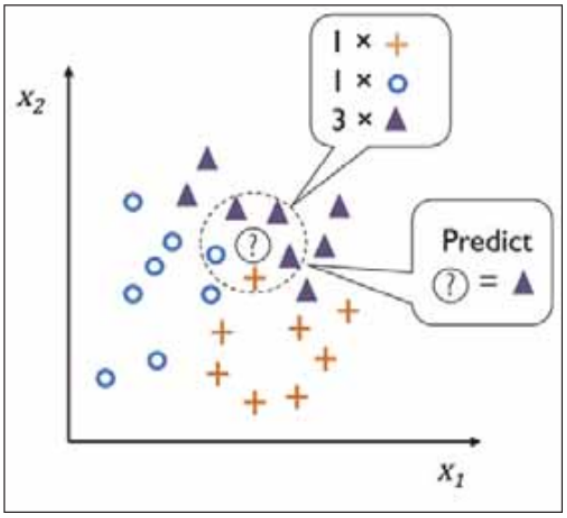


Figura 4.12: Ejemplo de KNN: El punto de datos se clasifica como "triángulo"según la votación mayoritaria entre sus 5 vecinos más cercanos **[35]**.

$$MD(x,y) = \sum_{i=1}^{n} |x_i - y_i| \tag{4.8}$$

Se utiliza para medir la distancia entre dos puntos en un espacio de coordenadas donde solo se permiten movimientos horizontales y verticales, similar a cómo se movería un taxi en una ciudad con calles dispuestas en una cuadrícula.

- **Chebychev:** Conocida como distancia de valor máximo, distancia de Lagrange y distancia de tablero de ajedrez. Esta métrica se define en un espacio vectorial, donde la distancia entre dos vectores o puntos con coordenadas estándar se determina como la mayor de sus diferencias a lo largo de cualquier dimensión de coordenada.

$$CD(x,y) = \max_{i} |x_i - y_i| \tag{4.9}$$

- **Coseno:** La distancia coseno, también conocida como distancia angular, mide el ángulo entre dos vectores. Esta distancia se obtiene restando la similitud del coseno de uno:

$$CosD(x,y) = 1 - \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}} \tag{4.10}$$

- **Minkowski:** La distancia de Minkowski es una generalización de las métricas Euclideana, Manhattan y Chebychev, correspondientes a diferentes valores de p. Se puede expresar de la siguiente manera:

$$D_{Mink}(x,y) = \sqrt[p]{\sum_{i=1}^{n} |x_i - y_i|^p} \tag{4.11}$$

  Se convierte en la distancia Euclidiana cuando $p = 2$, en la distancia Manhattan cuando $p = 1$, y en la distancia de Chebychev cuando $p = \infty$.

Es importante mencionar que la función de distancia $d(x, y)$ entre dos vectores x y y se considera una métrica si satisface las siguientes propiedades **[1, 48]**:

1. **No negatividad:** La distancia entre x e y siempre es un valor no negativo ($\geqslant 0$).

$$d(x,y) \geqslant 0 \tag{4.12}$$

2. **Identidad de los indiscernibles:** La distancia entre x e y es igual a 0 si y solo si x es igual a y.

$$d(x,y) = 0 \quad \text{si} \quad x = y \tag{4.13}$$

3. **Simetría:** La distancia entre x e y es igual a la distancia entre y e x.

$$d(x,y) = d(y,x) \tag{4.14}$$

4. **Desigualdad triangular:** Considerando la presencia de un tercer punto z, la distancia entre x e y siempre es menor o igual a la suma de la distancia entre x y z y la distancia entre y y z.

$$d(x,y) \leqslant d(x,z) \tag{4.15}$$

4.3.4. Validación cruzada

Para la evaluación del modelo se utilizó una técnica de validación cruzada. En esta técnica, el conjunto de datos etiquetados se divide aleatoriamente en varios subconjuntos. En cada iteración del proceso de validación cruzada, uno de estos subconjuntos se utiliza como conjunto de prueba o validación, mientras que los subconjuntos restantes se combinan para formar el conjunto de entrenamiento. La muestra de entrenamiento se utiliza para ajustar los parámetros del modelo en el clasificador, mientras que el conjunto de prueba se emplea para estimar el error de generalización **[40, 11]**.
Dentro de cada iteración de la validación cruzada, la muestra de entrenamiento puede dividirse a su vez en dos subconjuntos disjuntos **[40]**:

- Subconjunto de estimación para seleccionar el modelo.
- Subconjunto de validación interna para probar o validar el modelo dentro del conjunto de entrenamiento.

Para obtener un bajo error de generalización, se entrena el clasificador hasta alcanzar un mínimo en este error de validación **[11]**.
Debido a que se dispone de un conjunto de datos relativamente pequeño, se utilizó el método de validación cruzada dejando uno fuera, conocido en inglés como Leave One Out Cross Validation (LOOCV). En este método, se utilizan $N - 1$ ejemplos etiquetados disponibles para entrenar el modelo, y el modelo se valida probándolo en el ejemplo que se deja fuera. El experimento se repite un total de N veces, dejando fuera un ejemplo diferente para validación cada vez. Luego, el error cuadrático bajo validación se promedia sobre las N pruebas del experimento **[40]**.

4.3.5. Matriz de confusión

La matriz de confusión se utiliza para evaluar el rendimiento de un modelo de clasificación, permitiendo visualizar cómo el algoritmo clasifica las instancias de las clases. Esta matriz muestra el número de veces que las instancias de la clase A se clasifican como clase B, proporcionando una tabla con cuatro combinaciones de valores predichos y reales, como se ilustra en la Figura 4.13 **[35, 18]**. Los componentes principales de una matriz de confusión son **[35, 20, 7]**:

- **Verdaderos Positivos (VP):** Número de instancias que pertenecen a la clase positiva y que el modelo clasifica correctamente como positivas. En nuestro caso, serían aquellos participantes con DCL.
- **Falsos Positivos (FP):** Número de instancias que no pertenecen a la clase positiva, pero que el modelo clasifica incorrectamente como positivas.
- **Verdaderos Negativos (VN):** Número de instancias que no pertenecen a la clase positiva y que el modelo clasifica correctamente como negativas. Considerados como participantes control.
- **Falsos Negativos (FN):** Número de instancias que pertenecen a la clase positiva, pero que el modelo clasifica incorrectamente como negativas.

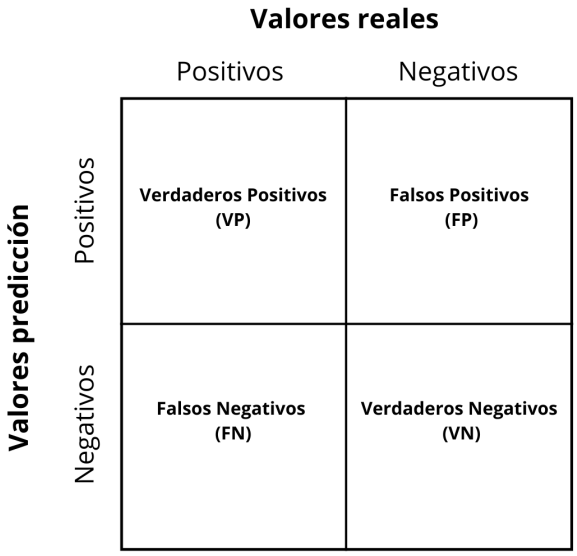


Figura 4.13: Matriz de confusión.

4.3.6. Métricas de evaluación

A partir de la matriz de confusión se obtienen las métricas de evaluación, las cuales nos permiten optimizar el algoritmo de clasificación y visualizar el rendimiento del modelo **[20]**. A pesar de que existen muchas métricas, entre las más utilizadas se encuentran **[20, 35, 41, 24]**:

- **Exactitud:** Mejor conocida en inglés como accuracy, determina la proporción de predicciones correctas del algoritmo y está definida como:

$$A = \frac{TP + TN}{\text{Número total de muestras}} \tag{4.16}$$

- **Precisión:** Mide la exactitud de las predicciones positivas entre el total de casos predichos como positivos. Está dada por:

$$P = \frac{TP}{TP + FP} \tag{4.17}$$

- **Sensibilidad:** Conocida en inglés como recall, mide la fracción de verdaderos positivos correctamente clasificados, definida como:

$$R = \frac{TP}{TP + FN} \tag{4.18}$$

- **F1-score:** Es el promedio ponderado de la precisión y la sensibilidad, calculado de la siguiente manera:

$$F1 = 2 \cdot \frac{P \cdot R}{P + R} \tag{4.19}$$

Para cuestiones de este trabajo, solo se hizo uso de la métrica de evaluación de exactitud.

Capítulo 5

Resultados y discusión

5.1. Exactitud

Se obtuvieron los siguientes porcentajes de exactitud para KNN de 2 a 13 vecinos mas cercanos y diferentes métricas (Tabla 5.1):

Algoritmo	**Métrica**		
	Euclidiana	**Manhattan**	**Coseno**
KNN (2 vecinos)	74 %	**77 %**	74 %
KNN (3 vecinos)	80 %	**83 %**	80 %
KNN (4 vecinos)	78 %	**81 %**	**81 %**
KNN (5 vecinos)	78 %	**81 %**	77 %
KNN (6 vecinos)	73 %	72 %	**75 %**
KNN (7 vecinos)	73 %	70 %	**78 %**
KNN (8 vecinos)	75 %	69 %	**78 %**
KNN (9 vecinos)	73 %	73 %	**75 %**
KNN (10 vecinos)	72 %	69 %	**76 %**
KNN (11 vecinos)	69 %	74 %	**76 %**
KNN (12 vecinos)	69 %	68 %	**75 %**
KNN (13 vecinos)	70 %	69 %	**73 %**

Tabla 5.1: Tasa de exactitud del algoritmo KNN con diversas métricas.

El KNN con $k = 3$ y una métrica de Manhattan obtuvo el mayor porcentaje de exactitud. Asimismo, se calcularon los porcentajes para los algoritmos de clasificación XGBoost, Árbol de Decisión y SVM. Estos fueron comparados con el mejor resultado obtenido por KNN, como se muestra en la Tabla 5.2..

Algoritmo	Exactitud
XGBoost	72 %
Árbol de decisión	70 %
SVM	56 %
KNN (3 vecinos - Manhattan)	**83 %**

Tabla 5.2: Exactitud para diferentes algoritmos de clasificación.

Entre los algoritmos entrenados, KNN con $k = 3$ y métrica de Manhattan mostró el mejor rendimiento, alcanzando una tasa de precisión del 83 %.

5.2. Estadística

Dado que disponemos de dos grupos independientes, el grupo DCL y el grupo de control, se optó por emplear la prueba t de Student para comparar las medias de las muestras. Para llevar a cabo el análisis estadístico, primero definimos la hipótesis nula (H_0) que establece que no hay diferencia de medias entre los dos grupos de la población, y la hipótesis alterna (H_a), que indica una diferencia significativa:

$$\begin{aligned} H_0 &: \mu_1 = \mu_2 \\ H_a &: \mu_1 \neq \mu_2 \end{aligned} \tag{5.1}$$

En nuestro caso, H_0 sugiere que no hay diferencia entre la dimensión fractal de los adultos mayores del grupo de control y aquellos con DCL.
Una vez que se han establecido las hipótesis, calculamos el valor p, el cual representa la probabilidad de observar los datos (o resultados aún más extremos) bajo la suposición de que H_0 es verdadera. Si el valor p es menor que el nivel de significancia ($\alpha = 0.05$), se rechaza H_0. En caso contrario, si el valor p es mayor que el nivel de significancia, no se rechaza H_0.
A partir de un programa desarrollado en Python, se obtuvo un valor $p = 0.011$, lo que indica que $p < 0.05$. Esto nos indica que los resultados son estadísticamente significativos y proporcionan evidencia a favor de H_a.
Si el valor p es superior al 5 %, se asume que ambos grupos tienen igual varianza. Sin embargo, en nuestro caso, esta suposición no se cumple. Por lo tanto, para abordar esta

situación, calculamos el estadístico t para muestras independientes con varianza desigual. En el código utilizado, la función *ttest_ ind* de la biblioteca *scipy.stats* calcula el estadístico t utilizando una fórmula específica para la prueba t de Welch, una versión modificada de la prueba t que es robusta a las diferencias en las varianzas de los grupos. Esto resulta en un valor estadístico t igual a 2.552.
Finalmente, en la Figura 5.1 se presenta la distribución t de Student con el valor t calculado previamente y un valor t crítico de 1.961.

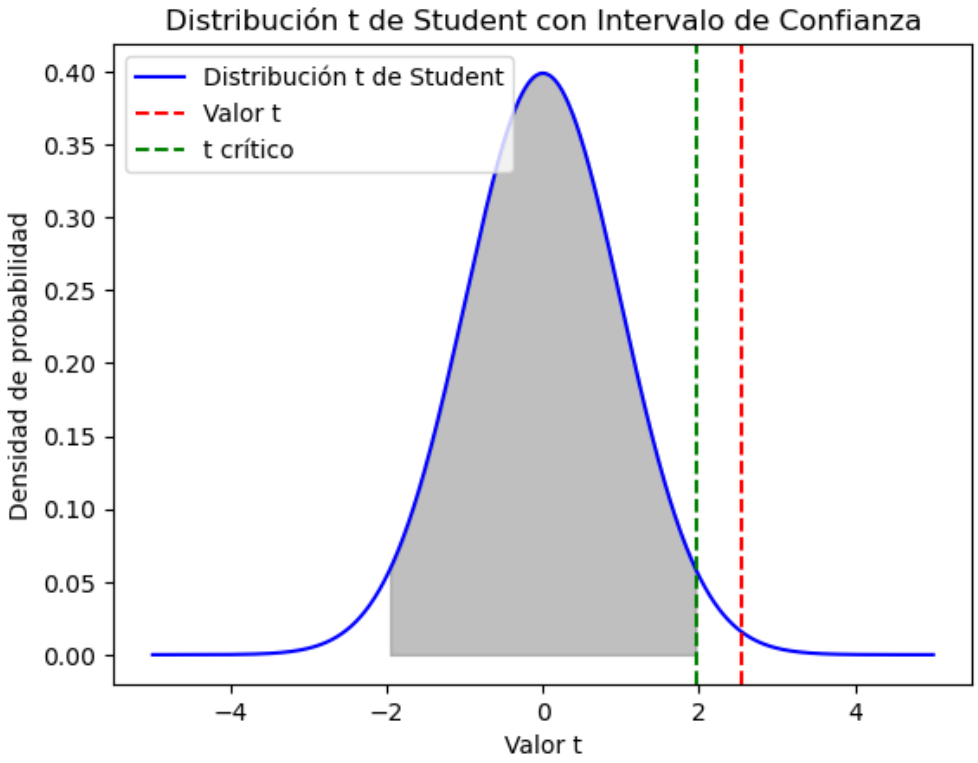


Figura 5.1: Distribución t de Student con intervalo de confianza.

Por otro lado, se realizó una prueba t de Student independiente para determinar si existen diferencias significativas en las medias de las áreas cerebrales entre los grupos de control y DCL. Los resultados de estas pruebas se detallan en la Tabla 5.3, utilizando 86 grados de libertad dentro de los grupos.

Área cerebral	Media		Desviación estándar		Estadística T	Valor P
	Control	DCL	Control	DCL		
C3	1.425	1.407	0.069	0.064	1.255	0.213
C4	1.447	1.416	0.067	0.071	2.128	**0.036**
CZ	1.422	1.414	0.058	0.064	0.609	0.544
F3	1.424	1.362	0.076	0.081	0.544	**0.0**
F4	1.415	1.374	0.083	0.086	2.262	**0.026**
F7	1.401	1.346	0.104	0.075	2.799	**0.006**
F8	1.394	1.389	0.084	0.084	0.287	0.775
FP1	1.336	1.302	0.092	0.112	1.55	0.125
FP2	1.329	1.303	0.102	0.121	1.112	0.269
FZ	1.392	1.365	0.076	0.085	1.596	0.114
O1	1.42	1.448	0.064	0.059	-2.055	**0.043**
O2	1.436	1.459	0.049	0.053	-2.165	**0.033**
P3	1.43	1.433	0.048	0.048	-0.299	0.766
P4	1.434	1.443	0.052	0.058	-0.832	0.408
PZ	1.426	1.428	0.052	0.054	-0.157	0.876
T3	1.442	1.434	0.075	0.088	0.461	0.646
T4	1.456	1.442	0.064	0.073	0.969	0.335
T5	1.439	1.441	0.06	0.074	-0.179	0.858
T6	1.439	1.471	0.058	0.045	-2.882	**0.005**
LOG	1.241	1.222	0.073	0.075	1.219	0.226
ROG	1.249	1.256	0.084	0.081	-0.42	0.675

Tabla 5.3: Medias y desviaciones estándar de las dimensiones fractales y sus comparaciones mediante pruebas t de Student para cada área cerebral y las actividades eléctricas oculares.

Los valores de C4, F3, F4, F7, O1, O2 y T6 rechazan la hipótesis nula, indicando así la presencia de diferencias significativas entre los grupos.

Capítulo 6

Conclusiones y trabajo futuro

Los resultados de este estudio indican que es posible detectar DCL mediante el análisis de imágenes fractales en adultos mayores, ya que se evidencian diferencias significativas entre los participantes control y aquellos con DCL. La selección del clasificador de IA implica un proceso de evaluación y selección de características entre diferentes opciones para determinar cuál cumple mejor con los requisitos específicos del problema. Esta selección debe basarse en un examen exhaustivo de la naturaleza de los datos considerados.
La evaluación de clasificadores de IA para la detección de DCL utilizando patrones fractales de EEG en adultos mayores en reposo reveló que el clasificador KNN con tres vecinos más cercanos y la métrica de Manhattan mostró la mayor efectividad, logrando una tasa de exactitud del 83 %. El análisis estadístico reveló que la dimensión fractal fue capaz de detectar alteraciones significativas en regiones cerebrales específicas, incluidas las regiones frontal y temporal, que están asociadas con los procesos de toma de decisiones y memoria. Además, la disociación entre las regiones anteriores y posteriores del cerebro fue de considerable importancia. El DCL parece alterar la dimensión fractal de las regiones frontales bilaterales y la región central derecha, mientras que preserva las regiones posteriores.
Como trabajo futuro, se propone la realización de una interfaz que permita a los usuarios realizar los recortes de cada segmento, obtener su dimensión fractal y realizar la clasificación de los datos de forma sencilla.

Bibliografía

[1] Haneen Arafat Abu Alfeilat, Ahmad B. A. Hassanat, Omar Lasassmeh, Ahmad S. Tarawneh, Mahmoud Bashir Alhasanat, Hamzeh S. Eyal Salman, and V. B. Surya Prasath. Effects of distance measure choice on k-nearest neighbor classifier performance: A review. *Big Data*, 7(4):221–248, 12 2019. doi:10.1089/big.2018.0175.

[2] A. Accardo, M. Affinito, M. Carrozzi, and F. Bouquet. Use of the fractal dimension for the analysis of electroencephalographic time series. *Biological Cybernetics*, 77(5):339–350, 11 1997. doi:10.1007/s004220050394.

[3] Michael F. Barnsley and Hawley Rising. *Fractals Everywhere*. Morgan Kaufmann, 1993. 568 pages.

[4] Cristina Alexandra Benavides Caro. Deterioro cognitivo en el adulto mayor. *Revista Mexicana de Anestesiología*, 40(2):107–112, 2017.

[5] Jose Bernal, Kaisar Kushibar, Daniel S. Asfaw, Sergi Valverde, Arnau Oliver, Robert Martí, and Xavier Lladó. Deep convolutional neural networks for brain image analysis on magnetic resonance imaging: a review. *Artificial Intelligence in Medicine*, 95:64–81, 04 2019. URL: https://linkinghub.elsevier.com/retrieve/pii/S0933365716305206, doi:10.1016/j.artmed.2018.08.008.

[6] Katarzyna Blinowska and Piotr Durka. Electroencephalography (EEG). 04 2006. doi:10.1002/9780471740360.ebs0418.

[7] Ricardo Borja Robalino, Antonio Monleon Getino, and Jose Benedé. Estandarización de métricas de rendimiento para clasificadores machine y deep learning. 02 2020.

[8] Kittipong Chomboon, Pasapitch Chujai, Pongsakorn Teerarassammee, Kittisak Kerdprasop, and Nittaya Kerdprasop. *An Empirical Study of Distance Metrics for k-Nearest Neighbor Algorithm*. 01 2015. 285 pages. doi:10.12792/iciae2015.051.

[9] Felipe Humberto Contreras Alcalá. fhca/attractors, 11 2022. Accessed: 2024. URL: https://github.com/fhca/attractors.

[10] Helton Hugo de Carvalho, Robson Luiz Moreno, Tales Cleber Pimenta, Paulo C. Crepaldi, and Evaldo Cintra. A heart disease recognition embedded system with fuzzy cluster algorithm. *Computer Methods and Programs in Biomedicine*, 110(3):447–454, 06 2013. URL: https://www.sciencedirect.com/science/article/pii/S0169260713000084, doi:10.1016/j.cmpb.2013.01.005.

[11] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. John Wiley & Sons, 11 2000. 679 pages.

[12] Alberto Freire Pérez. Métodos de cribaje del deterioro cognitivo leve en atención primaria. *Revista Española de Geriatría y Gerontología*, 52:15–19, 06 2017. URL: https://linkinghub.elsevier.com/retrieve/pii/S0211139X1830074X, doi:10.1016/S0211-139X(18)30074-X.

[13] Valeria García Muñoz. El color del ruido en el sueño MOR de adultos mayores con y sin deterioro cognitivo. Tesis de licenciatura, 06 2018. 83 pages. URL: https://www.uaeh.edu.mx/docencia/Tesis/icbi/licenciatura/documentos/2018/tesis-valeria-garcia-munoz.pdf.

[14] Ary L. Goldberger, Luis A. N. Amaral, Jeffrey M. Hausdorff, Plamen Ch. Ivanov, C.-K. Peng, and H. Eugene Stanley. Fractal dynamics in physiology: Alterations with disease and aging. *Proceedings of the National Academy of Sciences*, 99:2466–2472, 02 2002. URL: https://pnas.org/doi/full/10.1073/pnas.012579499, doi:10.1073/pnas.012579499.

[15] Rui Guo, Zhiqian Zhao, Tao Wang, Guangheng Liu, Jingyi Zhao, and Dianrong Gao. Degradation state recognition of piston pump based on iceemdan and xgboost. *Applied Sciences*, 10:6593, 09 2020. doi:10.3390/app10186593.

[16] Luis Miguel Gutiérrez Robledo, María del Carmen García Peña, Paloma Arlet Roa Rojas, and Adrián Martínez Ruiz. *La enfermedad de Alzheimer y otras demencias como problema nacional de salud.* Intersistemas, primera edition, 2017. 127 pages. URL: https://www.gob.mx/inger/documentos/la-enfermedad-de-alzheimer-y-otras-demencias-como-problema-nacional-de-salud.

[17] José Gutiérrez Rodríguez and Germán Guzmán Gutiérrez. Definición y prevalencia del deterioro cognitivo leve. *Revista Española de Geriatría y Gerontología*, 52:3–6, 06 2017. URL: https://linkinghub.elsevier.com/retrieve/pii/S0211139X18300726, doi:10.1016/S0211-139X(18)30072-6.

[18] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 09 2019. 851 pages.

[19] Caroline M Hagerhall, Thorbjörn Laike, Richard P Taylor, Marianne Küller, Rikard Küller, and Theodore P Martin. Investigations of human EEG response to viewing fractal patterns. *Perception*, 37(10):1488–1494, 10 2008. URL: http://journals.sagepub.com/doi/10.1068/p5918, doi:10.1068/p5918.

[20] Mohammad Hossin and Sulaiman M.N. A review on evaluation metrics for data classification evaluations. *International Journal of Data Mining & Knowledge Management Process*, 5:01–11, 03 2015. doi:10.5121/ijdkp.2015.5201.

[21] Anna K. Jerebko, Ronald M. Summers, James D. Malley, Marek Franaszek, and C. Daniel Johnson. Computer-assisted detection of colonic polyps with CT colonography using neural networks and binary classification trees. *Medical Physics*, 30(1):52–60, 2003. URL: https://onlinelibrary.wiley.com/doi/abs/10.1118/1.1528178, doi:10.1118/1.1528178.

[22] Simarjeet Kaur, Jimmy Singla, Lewis Nkenyereye, Sudan Jha, Deepak Prashar, Gyanendra Prasad Joshi, Shaker El-Sappagh, Md. Saiful Islam, and S. M. Riazul Islam. Medical diagnostic systems using artificial intelligence (AI) algorithms: Principles and perspectives. *IEEE Access*, 8:228049–228069, 2020. URL: https://ieeexplore.ieee.org/abstract/document/9279211, doi:10.1109/ACCESS.2020.3042273.

[23] Igor Kononenko. Machine learning for medical diagnosis: history, state of the art and perspective. *Artificial Intelligence in Medicine*, 23(1):89–109, 08 2001. URL: https://www.sciencedirect.com/science/article/pii/S093336570100077X, doi:10.1016/S0933-3657(01)00077-X.

[24] Yixuan Li and Zixuan Chen. Performance evaluation of machine learning methods for breast cancer prediction. *Applied and Computational Mathematics*, 7(4):212–216,

10 2018. URL: https://www.sciencepg.com/article/10.11648/j.acm.20180704.15, doi:10.11648/j.acm.20180704.15.

[25] Batta Mahesh. Machine learning algorithms - a review. pages 381–386, 01 2019. doi:10.21275/ART20203995.

[26] Benoit B. Mandelbrot. *The Fractal Geometry of Nature*. Henry Holt and Company, 1983. 504 pages.

[27] Rory Mitchell and Eibe Frank. Accelerating the XGBoost algorithm using GPU computing. *PeerJ Computer Science*, 3:127, 2017. URL: https://peerj.com/articles/cs-127, doi:10.7717/peerj-cs.127.

[28] Mercedes Montenegro Peña, Pedro Montejo Carrasco, Marcos Llanero Luque, and Ana Isabel Reinoso García. Evaluación y diagnóstico del deterioro cognitivo leve. *Rev. logop. foniatr. audiol. (Ed. impr.)*, 32(2):47–56, 2012. URL: https://pesquisa.bvsalud.org/portal/resource/pt/ibc-100322.

[29] Hamidreza Namazi, Erfan Aghasian, and Tirdad Seifi Ala. Fractal-based classification of electroencephalography (EEG) signals in healthy adolescents and adolescents with symptoms of schizophrenia. *Technology and Health Care: Official Journal of the European Society for Engineering and Medicine*, 27(3):233–241, 2019. doi:10.3233/THC-181497.

[30] Leonardo Palacios Sanchez. Breve historia de la electroencefalografía. *Acta Neurológica Colombiana*, 18(2):104–107, 2002.

[31] Ulises Paredes-Hernández, Patricia Pliego-Pastrana, Enrique Vázquez-Mendoza, Consuelo Morgado-Valle, Luis Beltran-Parrazal, Arturo Criollo-Perez, and Erika Elizabeth Rodriguez-Torres. Fractal and multifractal characterization of in vitro respiratory recordings of the pre-bötzinger complex. *Brain Multiphysics*, 2:100026, 01 2021. URL: https://www.sciencedirect.com/science/article/pii/S266652202100006X, doi:10.1016/j.brain.2021.100026.

[32] Heinz Otto Peitgen, Hartmut Jürgens, and Dietmar Saupe. *Chaos and Fractals: New Frontiers of Science*. Springer Science & Business Media, 02 2004. 926 pages.

[33] Montri Phothisonothai and Masahiro Nakagawa. Fractal-based EEG data analysis of body parts movement imagery tasks. *The Journal of Physiological Sciences*,

57(4):217–226, 2007. URL: http://www.jstage.jst.go.jp/article/physiolsci/57/4/57_4_217/_article, doi:10.2170/physiolsci.RP006307.

[34] Salvador Quiroz González, Erika Elizabeth Rodríguez Torres, Bertha Segura Alegría, Javier Pereira Venegas, Rosa Estela Lopez Gomez, and Ismael Jiménez Estrada. Detrended fluctuation analysis of compound action potentials re-corded in the cutaneous nerves of diabetic rats. *Chaos, Solitons & Fractals*, 83:223–233, 02 2016. URL: https://www.sciencedirect.com/science/article/pii/S0960077915004294, doi:10.1016/j.chaos.2015.12.011.

[35] Sebastian Raschka and Vahid Mirjalili. *Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow*. Packt, second edition, 09 2017. 622 pages.

[36] Erika Rodriguez Torres, Alejandra Rosales Lagarde, Carlos Fernando Chávez Vega, José Luis Ocaña Garrido, Yair Alejandro Pardo Rosales, and Rodrigo Silva Mota. Análisis fractal del electroencefalograma durante la vigilia en reposo de adultos mayores hidalguenses y deterioro cognitivo. *Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI*, 7(14):10–13, 01 2020. URL: https://repository.uaeh.edu.mx/revistas/index.php/icbi/article/view/4334, doi:10.29057/icbi.v7i14.4334.

[37] Erika Elizabeth Rodríguez Torres, Valeria García Muñoz, and Jorge Viveros Rogel. Las estructuras fractales en las fibras musculares. *Universitarios Potosinos*, 13(204):44, 2016. URL: https://www.uaeh.edu.mx/campus/icbi/investigacion/matematicas/curriculums/documentos/erika/1.pdf.

[38] Erika Elizabeth Rodríguez Torres, Alejandra Rosales Lagarde, and Brenda Fernanda Noguez Ruiz. Detección del deterioro cognitivo en adultos mayores en reposo por medio de técnicas fractales. In *Ciencia y Transdisciplinariedad*. Tirant lo Blanch, 1ª edition, 11 2023. 506 pages. URL: https://editorial.tirant.com/mex/libro/ciencia-y-transdisciplinariedad-maria-del-rocio-hernandez-pozo-9788419588029.

[39] Alejandra Rosales Lagarde, Erika E. Rodriguez Torres, Benjamín A. Itzá Ortiz, Pedro Miramontes, Génesis Vázquez Tagle, Julio C. Enciso Alva, Valeria García Muñoz, Lourdes Cubero Rego, José E. Pineda Sánchez, Claudia I. Martínez Alcalá, and Jose S. Lopez Noguerola. The color of noise and weak stationarity at the NREM

to REM sleep transition in mild cognitive impaired subjects. *Frontiers in Psychology*, 9:1205, 07 2018. URL: https://www.frontiersin.org/articles/10.3389/fpsyg.2018.01205.

[40] Haykin Simon. *Neural networks and learning machines*. Prentice Hall, 3rd edition, 2009. 906 pages.

[41] Deepti Sisodia and Dilip Singh Sisodia. Prediction of diabetes using classification algorithms. *Procedia Computer Science*, 132:1578–1585, 01 2018. URL: https://www.sciencedirect.com/science/article/pii/S1877050918308548, doi:10.1016/j.procs.2018.05.122.

[42] Tarigoppula Sriram, M. Rao, G Narayana, T. Vital, and Kaladhar SVGK Dowluru. Intelligent parkinson disease prediction using machine learning algorithms. *IJEIT*, 3:212–215, 07 2013.

[43] Nima Tajbakhsh, Jae Y. Shin, Suryakanth R. Gurudu, R. Todd Hurst, Christopher B. Kendall, Michael B. Gotway, and Jianming Liang. Convolutional neural networks for medical image analysis: Full training or fine tuning. *IEEE Transactions on Medical Imaging*, 35(5):1299–1312, 05 2016. URL: http://ieeexplore.ieee.org/document/7426826/, doi:10.1109/TMI.2016.2535302.

[44] Vicente Talanquer. *Fractus, fracta, fractal: fractales, de laberintos y espejos*. Fondo de Cultura Económica, 2002. 108 pages.

[45] Robert Tibshirani, Jerome Friedman, and Trevor Hastie. *The Elements of Statistical: Learning Data Mining, Inference, and Prediction*. Springer, second edition, 01 2016. 767 pages.

[46] Takashi Tsuchiya. Circular chaos game representation of 1-d chaos and its relation to the complex weierstrass function. *International Journal of Bifurcation and Chaos*, 09(10):2069–2080, 10 1999. URL: https://www.worldscientific.com/doi/abs/10.1142/S0218127499001504, doi:10.1142/S0218127499001504.

[47] Turker Tuncer, Sengul Dogan, and Abdulhamit Subasi. A new fractal pattern feature generation function based emotion recognition method using EEG. *Chaos, Solitons & Fractals*, 144:110671, 03 2021. URL: https://www.sciencedirect.com/science/article/pii/S0960077921000242, doi:10.1016/j.chaos.2021.110671.

[48] Kilian Q. Weinberger and Lawrence K. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(9):207–244, 2009. URL: http://jmlr.org/papers/v10/weinberger09a.html.

[49] Rui Zhu, Yiwen Guo, and Jing-Hao Xue. Adjusting the imbalance ratio by the dimensionality of imbalanced data. *Pattern Recognition Letters*, 133:217–223, 05 2020. URL: https://www.sciencedirect.com/science/article/pii/S0167865520300829, doi:10.1016/j.patrec.2020.03.004.

[50] Cristina Zurique Sánchez, Miguel Oswaldo Cadena Sanabria, Marina Zurique Sánchez, Paul Anthony Camacho López, Marina Sánchez Sanabria, Santiago Hernández Hernández, Karen Velásquez Vanegas, and Andrea Ustate Valera. Prevalencia de demencia en adultos mayores de américa latina: revisión sistemática. *Revista Española de Geriatría y Gerontología*, 54(6):346–355, 11 2019. URL: https://linkinghub.elsevier.com/retrieve/pii/S0211139X19300113, doi:10.1016/j.regg.2018.12.007.

Apéndice A

Programas

A.1. Cálculo de dimensión fractal

El presente código recorre una estructura de carpetas en busca de archivos de registros EEG, procesa estos datos para calcular dimensiones fractales, organiza los resultados en un DataFrame y, finalmente, guarda estos resultados en un archivo CSV. El programa $ifsoac\_2$ y el código original adaptado y utilizado para este trabajo pueden visualizarse en **[9]**.

```
import os
import numpy as np, pandas as pd
from glob import glob
from ifsoac_2 import *

def dimension_fractal(orb):
    """(orb.real, orb.imag)"""
    scales = np.logspace(0.01, 10, num=16, endpoint=False, base=2)
    ns = []
    for escala in scales:
        h, _, _ = np.histogram2d(orb[0], orb[1], bins=(int(escala), int(escala)))
        n = np.sum(h > 0)
        ns.append(n)
        if n >= orb.shape[1]:
            break
    coeffs = np.polyfit(np.log2(scales[:len(ns)]), np.log2(ns), 1)
    return coeffs[0]

root_folder = "C:/Users/brend/Documents/Recortes/"

#Carpeta raíz contiene las carpetas "DCL" y "control"
subfolders = ["DCL", "Control"]

results = []
```

```
25
26 for subfolder in subfolders:
27     subfolder_path = os.path.join(root_folder, subfolder)
28     label = '1' if subfolder == "DCL" else '0'
29     if os.path.exists(subfolder_path):
30         for folder in os.listdir(subfolder_path):
31             folder_path = os.path.join(subfolder_path, folder)
32             if os.path.isdir(folder_path):
33                 for file_path in glob(os.path.join(folder_path, "*.txt")):
34                     #Verificar el tamaño del archivo antes de intentar leerlo
35                     if os.path.getsize(file_path) == 0:
36                         print(f"Archivo vacío: {file_path}. Ignorando...")
37                         continue
38
39                     #Separar el nombre de la carpeta por el guion bajo
40                     folder_name = folder.split('_')
41
42                     #Parte antes del guion bajo
43                     folder_name_before = folder_name[0] if len(folder_name) > 0 else''
44
45                     #Parte después del guion bajo
46                     folder_name_after = folder_name[1] if len(folder_name) > 1 else''
47
48                     #Obtener el nombre del archivo sin la extensión
49                     file_name = os.path.splitext(os.path.basename(file_path))[0]
50
51                     #Separar nombre por guion bajo y obtener último elemento
52                     file_number = file_name.split('_')[-1]
53                     data = pd.read_csv(file_path, header=None).to_numpy()[5000:158600]
54                     op = {"rotate":0, "cmap_ds":"CET_C1", "ventana_ds":1200, "cols_ds"
55                     :4}
56                     img_points = np.array(Ifsoac(data, op).jDC()).T
57                     fractal_dimension = dimension_fractal(img_points)
58                     results.append([label, folder_name_before, folder_name_after,
59                     file_number, fractal_dimension])
60
61 #Convertir los resultados en un DataFrame de Pandas
62 df = pd.DataFrame(results, columns=['Grupo', 'Participante', 'Lobulo', 'Minuto', '
       Dimensión Fractal'])
63 print(df)
64
65
66 #Pivotar la tabla para tener cada lóbulo en una columna con sus dimensiones fractales
67 df_pivoted = df.pivot_table(index=['Grupo', 'Participante', 'Minuto'], columns='Lobulo
       ', values='Dimensión Fractal', aggfunc='mean').reset_index()
68 print(df_pivoted)
69
70
71 #Guardar el DataFrame pivotado en un archivo CSV
72 df_pivoted.to_csv('DM.csv', index=False)
73 print("Archivo CSV guardado exitosamente.")
```

Listing A.1: Procesamiento y análisis de datos EEG para cálculo de dimensión fractal.

A.2. Modelos de clasificación con Validación Cruzada Leave-One-Out

El código que se muestra a continuación implementa los algoritmos de clasificación XGBoost, Árbol de Decisión, SVM y KNN, utilizando la técnica de Validación Cruzada Leave-One-Out. El script carga datos desde un archivo CSV, entrena los modelos, realiza predicciones sobre el conjunto de prueba y calcula tanto la matriz de confusión como el accuracy de cada modelo.

```
import numpy as np
import xgboost as xgb
import pandas as pd
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
from sklearn.model_selection import LeaveOneOut
from xgboost import XGBClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn import svm
from sklearn.neighbors import KNeighborsClassifier

#Importa la ruta del archivo
p3 = "C:\\Users\\brend\\Documents\\tesis\\programas\\DM.csv"
df_f3 = pd.read_csv(p3)

#Visualización de los datos
df_f3

#Visualización de los nombres de las columnas
column_names = df_f3.columns.tolist()
print(column_names)

X = df_f3.drop(columns='Grupo')
X = X.drop(columns='Participante')
X = X.drop(columns='Minuto')
y = df_f3['Grupo']

#XGBoost
def classify_with_xgboost(X_data, y_labels, pos, neg):
    num_samples = len(X_data)
    CM = []
    TP = 0
    TN = 0
    FP = 0
    FN = 0

    incorrect_indices = []

    for i in range(num_samples):
```

```
        #Leave one out
        X_train = X_data.drop(i)
        y_train = y_labels.drop(i)
        X_test = X_data.iloc[i].to_numpy()
        y_test = y_labels[i]

        #Entrenar XGBoost
        model = xgb.XGBClassifier()
        model.fit(X_train, y_train)

        #Predecir en datos de prueba
        y_pred = model.predict(X_test.reshape(1, -1))

        #Verifica si la predicción es incorrecta
        if y_pred != y_test:
               incorrect_indices.append(i)

        #Calcular matriz de confusión
        if y_test == pos:
            if y_pred == pos:
                TP += 1
            else:
                FN += 1
        else:
            if y_pred == neg:
                TN += 1
            else:
                FP += 1

    CM = np.array([[TP, FP], [FN, TN]])

    #Calcular accuracy
    total = np.sum(CM)
    accuracy = (TP + TN) / total

    return CM, incorrect_indices, accuracy


#Árbol de decisión
def classify_with_decision_tree_classifier(X_data, y_labels, pos, neg):
    num_samples = len(X_data)
    CM = []
    TP = 0
    TN = 0
    FP = 0
    FN = 0

    incorrect_indices = []

    for i in range(num_samples):

        #Leave one out
        X_train = X_data.drop(i)
```

```
        y_train = y_labels.drop(i)
        X_test = X_data.iloc[i].to_numpy()
        y_test = y_labels[i]

        #Entrenar árbol de decisión
        model = DecisionTreeClassifier()
        model.fit(X_train, y_train)

        #Predecir en datos de prueba
        y_pred = model.predict(X_test.reshape(1, -1))

        #Verifica si la predicción es incorrecta
        if y_pred!= y_test:
               incorrect_indices.append(i)

        #Calcular matriz de confusión
        if y_test == pos:
            if y_pred == pos:
                TP += 1
            else:
                FN += 1
        else:
            if y_pred == neg:
                TN += 1
            else:
                FP += 1

    CM = np.array([[TP, FP], [FN, TN]])

    #Calcular accuracy
    total = np.sum(CM)
    accuracy = (TP + TN) / total

    return CM, incorrect_indices, accuracy


#SVM
def classify_with_svm(X_data, y_labels, pos, neg):
    num_samples = len(X_data)
    CM = []
    TP = 0
    TN = 0
    FP = 0
    FN = 0

    incorrect_indices = []

    for i in range(num_samples):

        #Leave one out
        X_train = X_data.drop(i)
        y_train = y_labels.drop(i)
        X_test = X_data.iloc[i].to_numpy()
```

```
145         y_test = y_labels[i]
146 
147         #Entrenar SVM
148         model = svm.SVC(kernel='linear') # Linear Kernel
149         model.fit(X_train, y_train)
150 
151         #Predecir en datos de prueba
152         y_pred = model.predict(X_test.reshape(1, -1))
153 
154         #Verifica si la predicción es incorrecta
155         if y_pred!= y_test:
156                incorrect_indices.append(i)
157 
158         #Calcular matriz de confusión
159         if y_test == pos:
160             if y_pred == pos:
161                 TP += 1
162             else:
163                 FN += 1
164         else:
165             if y_pred == neg:
166                 TN += 1
167             else:
168                 FP += 1
169 
170     CM = np.array([[TP, FP], [FN, TN]])
171 
172     #Calcular accuracy
173     total = np.sum(CM)
174     accuracy = (TP + TN) / total
175 
176     return CM, incorrect_indices, accuracy
177 
178 
179 #KNN
180 def classify_with_knn(X_data, y_labels, pos, neg, k, metrica):
181     num_samples = len(X_data)
182     CM = []
183     TP = 0
184     TN = 0
185     FP = 0
186     FN = 0
187 
188     incorrect_indices = []
189 
190     for i in range(num_samples):
191 
192         #Leave one out
193         X_train = X_data.drop(i)
194         y_train = y_labels.drop(i)
195         X_test = X_data.iloc[i].to_numpy()
196         y_test = y_labels[i]
197 
```

```
        #Entrenar KNN
        model = KNeighborsClassifier(n_neighbors=k,  metric=metrica)
        model.fit(X_train, y_train)

        #Predecir en datos de prueba
        y_pred = model.predict(X_test.reshape(1, -1))

        #Verifica si la predicción es incorrecta
        if y_pred!= y_test:
               incorrect_indices.append(i)

        #Calcular matriz de confusión
        if y_test == pos:
            if y_pred == pos:
                TP += 1
            else:
                FN += 1
        else:
            if y_pred == neg:
                TN += 1
            else:
                FP += 1

    CM = np.array([[TP, FP], [FN, TN]])

    #Calcular accuracy
    total = np.sum(CM)
    accuracy = (TP + TN) / total

    return CM, incorrect_indices, accuracy


#Imprime resultados para XGBoost
CM, incorrect_indices, accuracy = classify_with_xgboost(X, y, 1, 0)
print("Confusion Matrix:")
print(CM)
print("Incorrect Predictions Indices:")
print(incorrect_indices)


# Redondear la precisión a dos decimales
print("Accuracy:", round(accuracy, 2))
print()


#Imprime resultados para árbol de decisión
import warnings

#Ignorar todas las advertencias del usuario
warnings.filterwarnings("ignore", category=UserWarning)
warnings.filterwarnings("ignore", message=".*mode.*", category=FutureWarning)
CM, incorrect_indices, accuracy = classify_with_decision_tree_classifier(X, y, 1, 0)
print("Confusion Matrix:")
```

```
print(CM)
print("Incorrect Predictions Indices:")
print(incorrect_indices)

#Redondear la precisión a dos decimales
print("Accuracy:", round(accuracy, 2))
print()


#Imprime resultados para SVM
import warnings

#Ignorar todas las advertencias del usuario
warnings.filterwarnings("ignore", category=UserWarning)
warnings.filterwarnings("ignore", message=".*mode.*", category=FutureWarning)
CM, incorrect_indices, accuracy = classify_with_svm(X, y, 1, 0)
print("Confusion Matrix:")
print(CM)
print("Incorrect Predictions Indices:")
print(incorrect_indices)

#Redondear la precisión a dos decimales
print("Accuracy:", round(accuracy, 2))
print()


#Imprime resultados para KNN
import warnings

#Ignorar todas las advertencias del usuario
warnings.filterwarnings("ignore", category=UserWarning)
warnings.filterwarnings("ignore", message=".*mode.*", category=FutureWarning)
neighbors = [2,3,4,5,6,7,8,9,10,11,12,13]

#Metricas que se pueden utilizar: euclidean, minkowski, cityblock, cosine, l1, l2,
    manhattan, nan_euclidean
for k in neighbors:
    print(f"Neighbor: {k}")
    CM, incorrect_indices, accuracy = classify_with_knn(X, y, 1, 0, k, metrica='
    cityblock')
    print("Confusion Matrix:")
    print(CM)
    print("Incorrect Predictions Indices:")
    print(incorrect_indices)

    #Redondear la precisión a dos decimales
    print("Accuracy:", round(accuracy, 2))
    print()
```

Listing A.2: Código en Python que implementa múltiples algoritmos de clasificación con LOOCV.

A.3. Procesamiento y segmentación de registros EEG

Para incrementar los datos, se desarrolló el siguiente código. Este programa recorre una carpeta principal y sus subcarpetas en busca de archivos de registros EEG, lee los datos de estos archivos, los divide en segmentos de un minuto y guarda estos segmentos en archivos separados. Además, verifica que un archivo específico contenga la cantidad esperada de datos.

```
import numpy as np
import pandas as pd
import os

#Programa para recortar los primeros 5 min del EEG
#Definir la carpeta principal donde se buscarán los archivos
p1 = "C:\\Users\\brend\\Documents\\Datos"

#Carpeta específica dentro de la carpeta raíz
carpeta_deseada = "Control"

#Extensión de archivo deseada (primeras 2 letras del archivo)
extension_deseada = "EM"

#Definir la carpeta donde se guardarán los recortes
carpeta_salida = "C:\\Users\\brend\\Documents\\Recortes\\Control"

#Verificar si la carpeta de salida existe, si no, crearla
if not os.path.exists(carpeta_salida):
    os.makedirs(carpeta_salida)

#Lista para almacenar las rutas de los archivos
rutas_archivos = []

#Recorre todas las carpetas y subcarpetas dentro de la ruta principal
for carpeta_raiz, carpetas, archivos in os.walk(p1):

    #Verifica si la carpeta actual es la carpeta deseada
    if carpeta_deseada in carpetas:

                #Obtiene la ruta completa de la carpeta deseada
                carpeta_deseada_ruta = os.path.join(carpeta_raiz, carpeta_deseada)

                 #Recorre todas las subcarpetas dentro de la carpeta deseada
                for subdir, _, files in os.walk(carpeta_deseada_ruta):

                     #Recorre los archivos dentro de la subcarpeta
                     for archivo in files:
                         #Verifica si las 3 letras del archivo coinciden con las dadas
                          if archivo[:2].upper() == extension_deseada.upper():
```

```
                            #Obtiene la ruta completa del archivo deseado
                              ruta_completa_archivo= os.path.join(subdir, archivo)
                              rutas_archivos.append(ruta_completa_archivo)

#Si no encontraron archivos, imprime un mensaje informativo
if not rutas_archivos:
     print("No se encontraron archivos con la extension deseada.")

else:

      #Recorre todas las rutas de archivos encontradas
      for ruta_archivo in rutas_archivos:

          #Abre el archivo en modo de lectura
          with open(ruta_archivo, "r") as file:

                #Lee todas las líneas del archivo y las guarda en una lista
                 lines = file.readlines()

                #Elimina los espacios en blanco y crea una lista de valores
                 values = [line.strip() for line in lines]

                #Convierte cada valor en un número flotante y guarda los resultados
                 values = [float(x) for x in values]

          #Lista para almacenar los datos de cada minuto (frecuencia*segundo)
          min1 = values[:256*60]
          min2 = values[256*60:256*60*2]
          min3 = values[256*60*2:256*60*3]
          min4 = values[256*60*3:256*60*4]
          min5 = values[256*60*4:256*60*5]

          #Crea una carpeta para almacenar los recortes de este archivo
          carpeta_archivo = os.path.join(carpeta_salida, os.path.basename(ruta_archivo
          ))
          os.makedirs(carpeta_archivo, exist_ok=True)

          #Guardar los recortes de cada minuto en archivos individuales
          for i, minuto_data in enumerate([min1, min2, min3, min4, min5], start=1):
               with open(os.path.join(carpeta_archivo, f"minuto_{i}.txt"), "w") as f:
                    for dato in minuto_data:
                         f.write(f"{dato}\n")

print("Recortes guardados exitosamente en la carpeta de salida.")


#Verificar que los archivos contengan el mismo numero de datos
#Definir la carpeta donde se guardarán los recortes
c1 = "C:\\Users\\brend\\Documents\\Recortes\\Control\\EMVIGOS2_C3.txt\\minuto_5.txt"

#Verificar si el archivo existe en la ruta especificada
if os.path.exists(c1):

```

```
    #Inicializar el contador total
    conteo_total = 0

    #Leer el archivo línea por línea
    with open(c1, "r") as f:

        #Iterar sobre cada línea del archivo
        for linea in f:

            #Dividir la línea en palabras o tokens
            # Esto asume que las palabras están separadas por espacios en blanco
            palabras = linea.split()

            #Contar la cantidad de palabras en la línea
            conteo_linea = len(palabras)

            #Agregar el conteo de la línea al conteo total
            conteo_total += conteo_linea

    #Imprimir el conteo total de datos en el archivo
    print("El conteo total de datos en el archivo es:", conteo_total)
```

Listing A.3: Código para procesamiento y segmentación de registros EEG.

A.4. Análisis comparativo con prueba t de Student

El programa analiza los datos de dimensión fractal de un archivo CSV, eliminando columnas no relevantes y dividiéndolos en dos grupos (control y DCL). Realiza un test t de Student para comparar los grupos, calculando y mostrando la t-estadística y el valor p redondeados. Además, calcula los grados de libertad y el valor crítico de t para un nivel de significancia del 5 %. Se visualiza la distribución t de Student, destacando la región crítica y el valor t calculado. Finalmente, imprime el número de valores en cada grupo, proporcionando una visión completa del análisis estadístico realizado.

```
import numpy as np
import pandas as pd
import seaborn as sns
from scipy.stats import ttest_ind, t
import matplotlib.pyplot as plt

p3 = "C:\\Users\\brend\\Documents\\tesis\\programas\\DM.csv"
df_f3 = pd.read_csv(p3)
```

```
df_f3

#Eliminar las columnas 'Participante' y 'Minuto'
df_f3 = df_f3.drop(columns=['Participante', 'Minuto', 'C3', 'C4', 'CZ', 'LOG', 'O1','O2','P3','P4', 'PZ', 'ROG', 'T3', 'T4', 'T5', 'T6'])

#Obtener todas las columnas que no sean 'Grupo'
columnas_interes = df_f3.columns[df_f3.columns != 'Grupo']

#Dividir los datos en dos grupos: control y DCL
grupo_control = df_f3[df_f3['Grupo'] == 0][columnas_interes].values.flatten()
grupo_DCL = df_f3[df_f3['Grupo'] == 1][columnas_interes].values.flatten()

#Realizar el test t de Student
t_statistic, p_value = ttest_ind(grupo_control, grupo_DCL)

#Redondear los valores a dos decimales
t_statistic = round(t_statistic, 3)
p_value = round(p_value, 3)

print("t-statistic:", t_statistic)
print("p-value:", p_value)

#Calcular los grados de libertad
n_control = len(grupo_control)
n_DCL = len(grupo_DCL)
df = n_control + n_DCL - 2

#Definir el nivel de significancia
alpha = 0.05

#Calcular los cuantiles t para el intervalo de confianza
t_critical = t.ppf(1 - alpha / 2, df)

#Crear la figura y los ejes
fig, ax = plt.subplots()

#Graficar la distribución t de Student
x = np.linspace(-5, 5, 1000)
y = t.pdf(x, df)
ax.plot(x, y, label='Student´s t-distribution', color='blue')

#Resaltar la región crítica de dos colas
ax.fill_between(x, 0, y, where=(x >= -t_critical) & (x <= t_critical), color='gray', alpha=0.5)

#Añadir etiquetas y título
ax.set_xlabel('T-value')
ax.set_ylabel('Probability density')

#Añadir la línea vertical para el valor t-statistic
ax.axvline(t_statistic, color='red', linestyle='--', label='T-value')

```

```
#Añadir la línea vertical para el valor t crítico
ax.axvline(t_critical, color='green', linestyle='--', label='Critical t-value')

#Añadir leyenda
ax.legend()

#Mostrar figura
plt.show()

#Redondear los valores a dos decimales
t = round(t_critical, 3)
t

#Obtener el número de valores en cada grupo
num_valores_control = len(grupo_control)
num_valores_DCL = len(grupo_DCL)

print("Número de valores en el grupo control:", num_valores_control)
print("Número de valores en el grupo DCL:", num_valores_DCL)
```

Listing A.4: Análisis estadístico comparativo de las dimensiones fractales entre grupo control y DCL utilizando la prueba t de Student.

A diferencia del código anterior, este realiza la comparación de la dimensión fractal entre dos grupos (control y DCL) para cada área cerebral. Para cada área, se ejecuta una prueba t de Student, se calculan los grados de libertad y se muestran los resultados, que incluyen la estadística t, el valor p, los grados de libertad, las medias y las desviaciones estándar de ambos grupos. Además, se interpretan los resultados según un nivel de significancia del 5 %, indicando si hay diferencias entre los grupos.

```

import pandas as pd
from scipy.stats import ttest_ind

#Cargar los datos
p3 = "C:\\Users\\brend\\Documents\\tesis\\programas\\DM.csv"
df_f3 = pd.read_csv(p3)

#Eliminar las columnas 'Participante' y 'Minuto'
df_f3 = df_f3.drop(columns=['Participante', 'Minuto'])
df_f3

#Dividir los datos en dos grupos: control y DCL
grupo_control = df_f3[df_f3['Grupo'] == 0]
grupo_dcl = df_f3[df_f3['Grupo'] == 1]

#Obtener los nombres de los lóbulos cerebrales
```

```
lobulos = df_f3.columns[1:]  # Suponiendo que las columnas son los lóbulos cerebrales

#Realizar prueba t de Student para cada lóbulo cerebral
for lóbulo in lobulos:

    #Realizar prueba t para el lóbulo cerebral actual
    resultado_ttest = ttest_ind(grupo_control[lóbulo], grupo_dcl[lóbulo])

     #Calcular los grados de libertad
    n1 = len(grupo_control[lóbulo])
    n2 = len(grupo_dcl[lóbulo])
    grados_libertad = n1 + n2 - 2

    #Imprimir los resultados
    print(f"\nPrueba t de Student para el lóbulo cerebral {lóbulo}:")
    print("Estadística t:", round(resultado_ttest.statistic, 3))
    print("Valor p:", round(resultado_ttest.pvalue, 3))
    print("Grados de libertad:", grados_libertad)

    #Calcular la media de cada grupo
    media_control = grupo_control[lóbulo].mean()
    media_dcl = grupo_dcl[lóbulo].mean()
    print("Media del grupo control:", round(media_control, 3))
    print("Media del grupo DCL:", round(media_dcl, 3))

    #Calcular la desviación estándar de cada grupo
    std_dev_control = grupo_control[lóbulo].std()
    std_dev_dcl = grupo_dcl[lóbulo].std()
    print("Desviación estándar del grupo control:", round(std_dev_control, 3))
    print("Desviación estándar del grupo DCL:", round(std_dev_dcl, 3))

    #Interpretación de los resultados
    alpha = 0.05
    if resultado_ttest.pvalue < alpha:
        print("Se rechaza la hipótesis nula. Hay diferencias significativas entre los
    grupos.")
    else:
        print("No se rechaza la hipótesis nula. No hay diferencias significativas
    entre los grupos.")
```

Listing A.5: Análisis estadístico comparativo de la dimensión fractal en cada área cerebral entre los grupos control y DCL utilizando la prueba t de Student.