Escuela Superior Politécnica del Litoral

Facultad de Ingeniería en Electricidad y Computación

Reconocimiento de patrones relacionados a la enfermedad de Parkinson usando sensores inerciales e Inteligencia Artificial

TECH-399

Proyecto Integrador

Previo la obtención del Título de:

Ingeniero en Ciencias de la Computación

Presentado por:
Ricardo Duval Molina Coronel
Andres Ulises Moscoso Esquivel

Guayaquil - Ecuador
Año: 2024

Dedicatoria

Este trabajo se lo dedico a mis padres, que me han acompañado durante este camino, y me han brindado su amor incodicional. Se lo dedicó también a mi familia y mentores por sus consejos y guías.

Ricardo Duval Molina Coronel

Dedico este proyecto a mi Familia, que incondicionalmente me han apoyado sin fin a lo largo de mi carrera y este proyecto. Además, a mis amigos, siempre ayudándome con lo que necesite y alentándome hacia adelante. También a los profesores y mentores que siempre creyeron en mí y enseñaron con pasión. Es gracias a todo este apoyo que he logrado estar donde me encuentro, y ha sido una parte sumamente importante de mi futuro.

Andrés Ulises Moscoso Esquivel

Agradecimientos

Mi más sincero agradecimiento a mis padres Emerita Coronel y Duval Molina, a mi hermana Valeria Molina, y el resto de mi familia por haberme acompañado en esta etapa de mi vida. Gracias también a Alexandra Galvan por haber estado presente y apoyándome durante tantos días de trabajo. Gracias también a mis tutores y cliente de este proyecto por su compromiso. Finalmente, gracias a mis amigos como Jared Castillo por ayudarme con herramientas y ayuda para que sea más llevadera esta etapa, y por su puesto a mi compañero Andrés Moscoso por el apoyo en el desarrollo de este trabajo.

Ricardo Duval Molina Coronel

Agradecimientos

Quiero agradecer al Dr. Enrique Pelaez, PhD, el Dr. Francis Loayza, PhD, y al Dr. Edwin Valarezo, PhD. Por su apoyo y seguimiento a lo largo del proyecto y por proveer lo necesario para realizarlo. También al personal de la universidad ICESI que amablemente ayudaron con los datos utilizados. Y A mi compañero Ricardo Molina, por su excelente apoyo y esfuerzo tanto en este proyecto como en varios otros a lo largo de la carrera en la que hemos colaborado.

Andrés Ulises Moscoso Esquivel

Declaración Expresa

Declaración Expresa

Nosotros Andrés Ulises Moscoso Esquivel y Ricardo Duval Molina Coronel acordamos y reconocemos que:

La titularidad de los derechos patrimoniales de autor (derechos de autor) del proyecto de graduación corresponderá al autor o autores, sin perjuicio de lo cual la ESPOL recibe en este acto una licencia gratuita de plazo indefinido para el uso no comercial y comercial de la obra con facultad de sublicenciar, incluyendo la autorización para su divulgación, así como para la creación y uso de obras derivadas. En el caso de usos comerciales se respetará el porcentaje de participación en beneficios que corresponda a favor del autor o autores.

La titularidad total y exclusiva sobre los derechos patrimoniales de patente de invención, modelo de utilidad, diseño industrial, secreto industrial, software o información no divulgada que corresponda o pueda corresponder respecto de cualquier investigación, desarrollo tecnológico o invención realizada por nosotros durante el desarrollo del proyecto de graduación, pertenecerán de forma total, exclusiva e indivisible a la ESPOL, sin perjuicio del porcentaje que nos corresponda de los beneficios económicos que la ESPOL reciba por la explotación de nuestra innovación, de ser el caso.

En los casos donde la Oficina de Transferencia de Resultados de Investigación (OTRI) de la ESPOL comunique a los autores que existe una innovación potencialmente patentable sobre los resultados del proyecto de graduación, no se realizará publicación o divulgación alguna, sin la autorización expresa y previa de la ESPOL.

Guayaquil, 10 de octubre del 2024.

Ricardo Duval Molina
Coronel

Andrés Ulises Moscoso
Esquivel

Recibido MJ Oct 10/2024

Evaluadores

Firmado electrónicamente por:
JAVIER ALEJANDRO
TIBAU BENITEZ

Javier Alejandro Tibau Benitez, Ph.D.

Profesor de Materia

Firmado electrónicamente por:
COLON ENRIQUE
PELAEZ JARRIN

Colon Enrique Peláez Jarrin, Ph.D.

Tutor de proyecto

RESUMEN

La Enfermedad de Parkinson (EP) es una enfermedad actualmente incurable, que afecta a 1.5% de personas a partir de los 65 años. La detección temprana de la enfermedad permitiría mejorar la calidad de vida de los pacientes y su familia, ya que permitiría tomar acciones preventivas más oportunas que una detección en una etapa avanzada. Uno de los síntomas que suelen aparecer en etapas tempranas de la EP son los patrones de marcha modificados, que alteran el movimiento de brazos y piernas. Basado en esto, se propone el análisis de estos movimientos mediante técnicas de Inteligencia Artificial que permitan capturar patrones para clasificar pacientes con EP de sujetos sanos. Para el desarrollo de esta solución se usaron datos tomados con sensores IMU durante la marcha de sujetos sanos y pacientes con EP, ubicados en las muñecas y espina base de los sujetos, de la Universidad ICESI de Colombia. Se extrajeron características, escalogramas y señales temporales de estos datos, que fueron las entradas para modelos de ML Shallow y Deep Learning, de los cuales, el modelo Red MLP mediante el método de extracción de características, dio el mejor resultado, teniendo un 95% de exactitud. Se integró este modelo en una aplicación Web para uso médico, donde puedan ser subidos archivos de señales IMU de los pacientes y predecir la probabilidad de tener EP según el modelo.

Palabras Clave: Enfermedad de Parkinson, Machine Learning, Deep Learning, IMU, Marcha.

Abstract

Parkinson's disease (PD) is a currently incurable disease that affects 1.5% of people over the age of 65. Early detection of the disease would improve the quality of life of patients and their families, as it would allow for more timely preventive actions than detection at an advanced stage. One of the symptoms that usually appear in the early stages of PD are modified walking patterns, which alter the movement of arms and legs. Based on this, the analysis of these movements using Artificial Intelligence techniques is proposed to capture patterns to classify patients with PD from healthy subjects. For the development of this solution, data taken with IMU sensors during the walking of healthy subjects and patients with PD, located on the wrists and base of the spine of the subjects, from the ICESI University of Colombia, were used. Features, scalograms and temporal signals were extracted from this data, which were the inputs for ML Shallow and Deep Learning models, of which, the Red MLP model using the feature extraction method gave the best result, having 95% accuracy. This model was integrated into a Web application for medical use, where IMU signal files of patients can be uploaded and the probability of having PD can be predicted according to the model.

Keywords: Parkinson's disease, Machine Learning, Deep Learning, IMU, Walking.

Índice general

Abreviaturas

CNN Convolutional Neural Network

DL Deep Learning

EP Enfermedad de Parkinson

ESPOL Escuela Superior Politécnica del Litoral

FIEC Facultad de Ingeniería en Electricidad y Computación

GBM Gradient Boosting Machine

HI Hiposmia Idiopática

IA Inteligencia Artificial

IMU Inertial Management Unit

KNN K-Nearest Neighbots

LNB Laboratorio de Bioingeniería y Neuroimagen

LSTM Long-Short Term Memory

MDS-UPDRS Movement Disorder Society-Unified Parkinson's Disease Rating Scale

ML Machine Learning

MLP Multi Layer Perceptron

PCA Análisis de Componentes Principales

PSD Densidad de Potencia

SVM Support Vector Machine

TST Time-Series Transformer

Simbología

s	Segundo
g	Aceleración de la gravedad de terrestre
m	Metro

Índice de figuras

Índice de tablas

Capítulo 1

1. Introducción

1.1 Descripción del Problema

La Enfermedad de Parkinson (EP) es una enfermedad actualmente incurable [3], el segundo trastorno neurodegenerativo más común [2, 3] y tiene una prevalencia poblacional de aproximadamente el 1,5 % a los 65 años [1]. La EP se manifiesta con síntomas motores y no motores tales como trastornos de sueño, del ánimo, de la marcha, entre otros; mismos que afectan la calidad de vida del paciente y la de su familia [17]. Debido a esto es de especial interés una detección temprana de la enfermedad, ya que el diagnóstico de una enfermedad neurodegenerativa potencial podría ayudar a mejorar sus condiciones de vida y la de sus familiares como el cuidado, la planificación familiar, la decisión sobre la jubilación, la preparación psicológica entre otros factores [1], así como retrasar el progreso de la enfermedad lo más rápido posible mediante protocolos de tratamiento oportunos [2].

Algunos síntomas que se pueden presentar de forma temprana en los pacientes con EP son los patrones de marcha modificados, que alteran los movimientos en brazos y piernas [3]. Aunque la marcha en los pacientes con EP se ve condicionada principalmente por cambios en la cinemática de extremidades inferiores [3], se han descrito cambios motores sutiles en el balanceo de los brazos y en el movimiento del tronco [3], incluso durante primeras etapas de la enfermedad [3], que pueden ser relevantes para el diagnóstico y seguimiento [3]. Estos cambios sutiles en la marcha durante las primeras etapas de la enfermedad pueden llegar a ser imperceptibles para el ser humano [3], por lo que se usan las técnicas que nos provee actualmente la Inteligencia Artificial (IA) para poder analizar patrones en la aceleración lineal y angular de las extremidades superiores y el tronco de los pacientes durante la marcha. Además de esto, el uso de los dispositivos de lectura de estos movimientos conocidos como Inertial Management Unit (IMU) son una alternativa de bajo costo y no invasiva para los pacientes [10].

El cliente al que va dirigido el trabajo es el **Laboratorio de Bioingeniería y Neuroimagen (LNB)** de la **Escuela Superior Politécnica del Litoral (ESPOL) /** Laboratorio enfocado en la investigación principalmente sobre trastornos neurológicos.

1.2 Justificación del Problema

La patología de la EP puede iniciar hasta una década antes de que lo síntomas sean suficientemente severos para permitir un diagnóstico basado en criterios actuales, dificultando el tratamiento oportuno de los pacientes y creando una necesidad para el diagnóstico temprano de esta enfermedad, ya que la identificación temprana puede tener considerables efectos en la calidad de vida del paciente y su familia [15]. Existen otros trastornos que comparten síntomas motores similares a la EP [18]. A causa de esto el diagnóstico preciso de esta enfermedad causa dificultades [18].

Los cambios motores se suelen evaluar mediante escalas clínicas como la “Escala de Calificación Unificada de la Enfermedad de Parkinson de la Sociedad de Trastornos del Movimiento” (MDS-UPDRS Parte III) [3], este enfoque es altamente subjetivo [3]. En las últimas décadas, se han utilizado múltiples dispositivos tecnológicos para cuantificar los cambios en el control motor [3], algunas de estas tecnologías incluyen IMU [4]. Estudios recientes [1, 11, 14, 20] han demostrado que los IMU, miden el movimiento corporal con alta precisión y pueden detectar anomalías en la actividad muscular que no son visibles a simple vista [4]. La detección temprana de estos cambios motores mediante estos dispositivos es de especial interés debido a que es una alternativa de bajo costo [1] y pueden ofrecer una detección de la enfermedad más temprana, precisa, menos subjetiva y cuantificable [2].

Aunque existen algoritmos de aprendizaje automático (ML) y aprendizaje profundo (DL) que analicen los datos capturados durante la marcha en extremidades superiores mediante IMU

de pacientes con EP [2, 11, 14], estas son limitadas en número y alcance [3], no tomando en cuenta características de procesamiento o diseño que mejorarían el desempeño de los algoritmos al momento de la predicción [3].

1.3 Objetivos

1.3.1 Objetivo general

Desarrollar un prototipo funcional de un modelo basado en IA capaz de identificar patrones asociados a los síntomas motores de la EP a partir del análisis de señales electrofisiológicas y datos electromiográficos de sujetos sanos y pacientes enfermos, para ayudar a los personales médicos en el diagnóstico preciso y temprano de la enfermedad.

1.3.2 Objetivos específicos

1. Recolectar y preprocesar datos electromiográficos y señales electrofisiológicas provenientes de IMU, asegurando que los datos sean de alta calidad para su análisis para hacer uso de estos en el entrenamiento del modelo.
2. Implementar y entrenar un modelo de aprendizaje automático capaz de identificar patrones motores relacionados con la enfermedad, a partir de los datos preprocesados para clasificar personas con EP de personas sanas.
3. Evaluar la efectividad del modelo predictivo utilizando métricas de rendimiento, para validar su capacidad para predecir personas con EP.
4. Desarrollar un aplicativo web para visualizar los resultados de la predicción del modelo de IA al subir un archivo de señales de IMU.

1.4 Marco teórico

1.4.1 Enfermedad de Parkinson (EP)

La EP es un trastorno neurológico progresivo caracterizado por una gran cantidad de características motoras y no motoras que pueden afectar la función en un grado variable [7]. La base anatomopatológica de la EP se caracteriza por la pérdida progresiva de neuronas dopaminérgicas de la sustancia negra pars compacta (SNpc) del mesencéfalo [19], así como la presencia de inclusiones intracelulares llamadas cuerpos de Lewy [19].

No existe una prueba definitiva para el diagnóstico de la EP [7], la enfermedad debe diagnosticarse basándose en criterios clínicos [7]. El temblor en reposo, la bradicinesia, la rigidez y la pérdida de los reflejos posturales se consideran generalmente los signos cardinales de la EP [7]. La presencia y la presentación específica de estas características se utilizan para diferenciar la EP de los trastornos parkinsonianos relacionados [7].

Entre el 70% y 90% de los pacientes con EP presentan síntomas motores [19]. Entre los síntomas y signos motores, los cardinales (bradicinesia, temblor en reposo y rigidez) se atribuyen principalmente a la pérdida de neuronas dopaminérgicas, pero los que involucran postura, equilibrio y marcha son en gran medida secundarios a la degeneración de vías no dopaminérgicas y contribuyen significativamente al deterioro y la discapacidad en pacientes con EP avanzada [8].

Existen varios signos motores cardinales que suceden en la EP, los cuales se describen a continuación [8]:

- Bradicinesia: Lentitud del movimiento voluntario y/o movimiento continuo.

- Temblor en reposo: Temblor asimétrico de amplitud moderada de 4 a 6 Hz, que suele afectar el pulgar.
- Rigidez: Aumento del tono muscular percibido durante el examen por el movimiento pasivo del segmento afectado.

Existen otros síntomas y signos motores que suceden en etapas tempranas y avanzadas de la enfermedad [8]:

- Inestabilidad postural: Ajuste postural con deterioro, debido a disminución o pérdida de los reflejos posturales.
- Acinesia: Reducción, retraso o falta de movimientos voluntarios o naturales.
- Hipocinesia: Amplitud de movimiento reducida, especialmente con movimientos repetitivos.
- Hipofonía: Volumen de voz reducido.
- Micrografía: Letra pequeña que se vuelve progresivamente más pequeña y menos legible.

1.4.2 Inertial Management Unit (IMU)

Las unidades de medición inercial (IMU) tienen una popularidad duradera en una variedad de aplicaciones industriales, desde sistemas de navegación hasta sistemas de robótica. Su uso en la práctica clínica ahora se está volviendo más común, gracias a la miniaturización y la capacidad de integrar funciones computacionales y de soporte de decisiones [9].

Una IMU suele estar compuesta por un acelerómetro y un giroscopio. En ocasiones, se puede combinar con un magnetómetro para que funcione como un sistema de navegación inercial, cada uno de estos componentes mide información en tres ejes — X, Y, y Z —, lo que permite capturar movimientos y orientaciones espaciales en 3D [10].

Las IMU son una opción prometedora como sensores portátiles para el seguimiento del movimiento en muchos aspectos. Una de las ventajas es que las IMU no tienen ninguna fuente. Dado que un acelerómetro y un giroscopio miden las aceleraciones lineales y las velocidades angulares respectivamente, que están relacionadas con el movimiento de los objetos donde están fijados los sensores, una IMU es un enfoque completamente autónomo [10]. Suelen ser usados también debido a su bajo costo y adquisición rápida de datos [11].

El análisis de la marcha de los pacientes basado en IMU es una herramienta útil para la evaluación , la detección y el diagnóstico de la EP , debido a su capacidad para capturar un amplio espectro de características de la marcha inducidas por deficiencias motoras [9].

1.4.3 Soluciones implementadas

En [2] trataron el análisis del balanceo del brazo de los pacientes mediante el uso de IMU. Hicieron uso de dos pulseras en las dos muñecas con IMU del modelo Meta Motion Rectangle de Mbientlab para medición, mientras los pacientes hacían la prueba TUG (Timed Up and Go) usada recurrentemente para detección de disfunciones motoras. Cada IMU posee un giroscopio de 3 ejes, un acelerómetro de 3 ejes y un magnetómetro de 3 ejes. A través de algoritmos como el de fusión de sensores de Bosch, obtuvieron parámetros de la marcha. Realizaron un re-muestreo de estas señales mediante la transformada de Wavelet con el fin de obtener matrices para entrenar una red CNN para una única señal, o una CNN multicanal para realizar una predicción más robusta y precisa tomando en cuenta todas las señales de los IMU

directamente en la red. Mediante estas técnicas clasificaron si los sujetos tenían disfunciones motoras relacionadas con EP o los sujetos se encontraban sanos.

En [11] desarrollaron un modelo de red neuronal capaz de distinguir una marcha con EP de la de una persona mayor sana, al mismo tiempo que se diferenció EP en etapa temprana de EP en etapa avanzada con alta precisión. Aplicaron el sistema APDM OPAL [12] con IMU portátiles para registrar los datos cinemáticos de cada sujeto, se colocaron cinco IMU en las pantorrillas, antebrazos y cintura de cada sujeto. Durante el período de recolección de datos, los sujetos caminaban hacia adelante y hacia atrás, a lo largo de una línea recta de diez metros de longitud. Cada IMU registró las aceleraciones en los 3 ejes y las velocidades angulares en 3 ejes con una frecuencia de muestreo de 128 Hz. Propusieron un modelo de red neuronal CNN para identificar EP y clasificar sus etapas. El modelo estaba compuesto por dos submodelos: submodelo I y submodelo II. El submodelo I estima si los datos de entrada de IMU son EP en etapa avanzada, mientras que el submodelo II clasifica los datos de entrada como EP en etapa temprana o No EP si los datos de IMU no son EP en etapa avanzada. La red consta de la capa de entrada, tres capas convolucionales, la capa de aplanamiento, tres capas completamente conectadas y la capa de salida.

En [14] utilizaron un dispositivo inercial portátil, llamado SensHand V1, para adquirir datos de movimiento de las extremidades superiores (muñeca y 3 falanges distales) durante la realización de seis tareas seleccionadas por MDS-UPDRS III. Esto lo realizaron con el fin de clasificar pacientes con EP, sujetos sanos y pacientes con hiposmia idiopática (HI), este último debido a que es una condición que ocurre previo a la EP. Realizaron múltiples análisis comparando tres algoritmos de aprendizaje supervisado, Support Vector Machine (SVM), Random Forest (RF) y Naive Bayes, en tres conjuntos de datos diferentes.

En [3] utilizaron un sistema de sensores portátil basado en la nube que incorpora acelerómetros triaxiales. Desarrollaron una herramienta de diagnóstico asistido de bajo costo para su uso en la cuantificación de la cinemática del balanceo del brazo de pacientes con EP. El grupo con EP mostró reducciones significativas ($p < 0,05$) en los valores RMS del balanceo del brazo, mayor asimetría del balanceo del brazo entre otras características. Por lo tanto, el sistema portátil proporciona un medio confiable para respaldar la práctica clínica en la evaluación de la EP.

En [5] crearon una aplicación móvil con Android, que permite conectarse a cuatro Simblee BLE con IMU para obtener los datos que toman de los sujetos y sean recibidos en una computadora. La aplicación permite ver las series de tiempo de los ejes de los acelerómetros en tiempo real. Configuraron estos dispositivos con algoritmos específicos para tomar los datos de estos acelerómetros en distintas frecuencias para pacientes con EP.

Capítulo 2

2. Metodología.

La metodología aplicada se compone de 5 etapas de forma secuencial. Estas etapas son las siguientes: obtención de datos, preprocesamiento de las señales, generación de entradas para entrenamiento de los modelos mediante distintos tipos de métodos, selección del mejor modelo para integración con la aplicación web y desarrollo de aplicación web. Estas etapas se pueden observar en el diagrama mostrado en la (**Figura 1**).

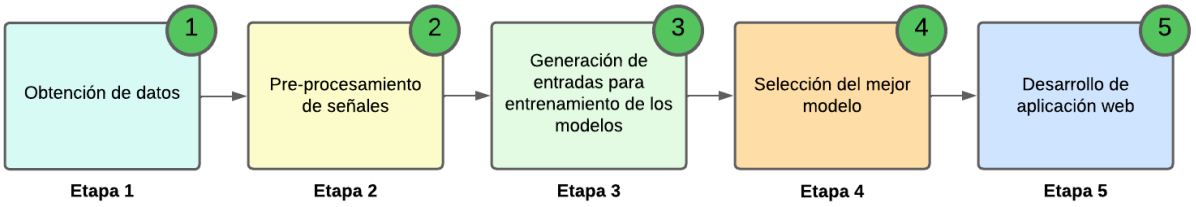

Figura 1. Etapas de la metodología aplicada (Autoría propia)

2.1 Obtención de datos

Se utilizó 2 conjuntos de datos provenientes de la Universidad ICESI (Cali, Colombia). Un primer conjunto de 18 archivos denominado S1, que fueron tomados de sujetos sanos que fueron utilizados como datos de control, o ejemplos de la clase negativa, y un segundo conjunto denominado S2 de 263 archivos, con diagnóstico positivo de la EP o que pertenecen a la clase positiva, el cual fue balanceado con la clase negativa eligiendo aleatoriamente 18 archivos de este conjunto. Ambos conjuntos de datos fueron alimentados con lecturas tomadas mediante 3 sensores IMU ubicados en cada sujeto, uno en la muñeca derecha del paciente, otro en la muñeca izquierda y el tercero en la espina base del paciente. Cada uno de estos 3 sensores capturaron información en forma de señales desde los 3 ejes — X, Y, y Z de un acelerómetro, y de los 3 ejes — X, Y, y Z de un giroscopio.

La ubicación de estos sensores IMU para la toma de datos se muestra en la (**Figura 2**). Los triángulos representan la ubicación los sensores IMU, y el sistema de referencia.

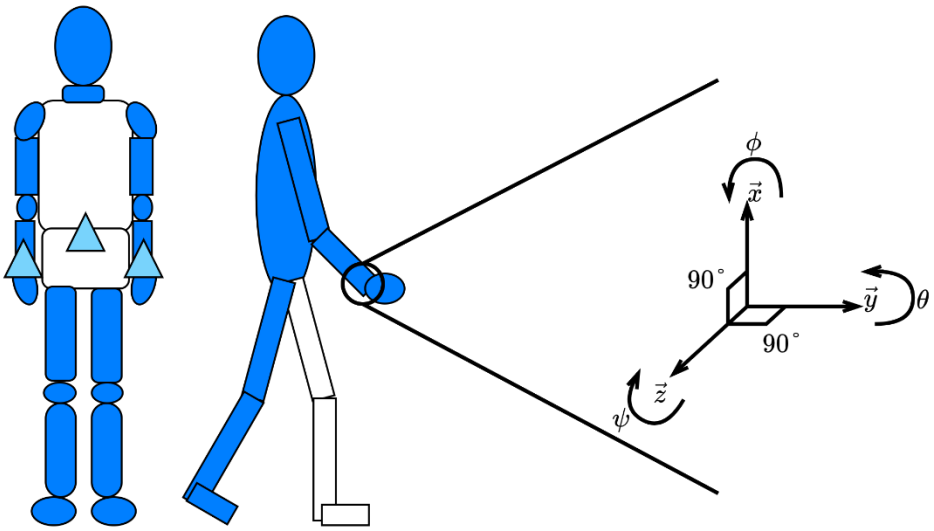

Figura 2. Ubicación de sensores en sujetos con su sistema de referencia (Universidad ICESI)

Cada uno de los sujetos en el experimento realizó caminatas en línea recta, dando giros en ciertos tramos, para luego nuevamente continuar con la caminata en línea recta, pero en sentido contrario.

Estos datos fueron preprocesados por la Universidad ICESI antes de su uso para este proyecto. El preprocesamiento realizado fue el siguiente:

En el caso de los acelerómetros de los sensores IMU, los valores se registraron en unidades de *g*, donde *1 g* corresponde a la aceleración debido a la gravedad terrestre. Para transformar estas lecturas en m/s², fijaron el sensor de manera que uno de sus ejes (por ejemplo, el eje y) estuviera alineado con la dirección de la gravedad, en esta posición el sensor debería registrar exactamente *1 g* en el eje alineado con la gravedad, mientras que los otros ejes deberían mostrar aproximadamente *0 g* si no hay movimiento adicional. El valor que reporta el sensor en ese eje

cuando está fijo en la dirección de la gravedad se utiliza como referencia para definir cuánto es *1 g* en las unidades del sensor, esto permite compensar posibles variaciones en las lecturas causadas por el hardware del sensor. Una vez que obtuvieron qué valor representa *1 g* en las unidades del sensor, determinaron un factor de conversión, este factor de conversión resultó de la división de la aceleración de la gravedad en m/s^2 entre el valor en unidades del acelerómetro para *1 g*, el factor de conversión resultante está estimado en $f = 9.8/4130\ [m/s^2]$. Luego, todas las lecturas en los 3 ejes del acelerómetro en unidades de *g* se multiplican por este factor de conversión para obtener los valores en m/s^2.

En el caso de los valores de velocidad angular registrados por el giroscopio, utilizaron el método de *Euler* sugerido por el fabricante de los sensores, el cual consiste en corregir los valores de la velocidad angular en base a un conjunto de derivadas, el enlace a la guía del fabricante donde detalla este método se muestra en [ANEXO 2]. Esto transforma los valores de la velocidad angular de los 3 ejes del giroscopio en *dps (Degrees Per Second).*

Adicional a esto, procesaron los datos mediante el algoritmo de interpolación Piecewise Cubic Hermite Interpolating Polynomial (PCHIP) [32], el cual es un método de interpolación que genera una curva suave que pasa por un conjunto de puntos dados, preservando la forma de los datos originales [32], esto lo realizaron para suavizar los datos de las señales. Finalmente, aplicaron un filtro paso bajo Butterworth [28] con una frecuencia de corte en 7.5 Hz a cada una de las señales del giroscopio y acelerómetro.

2.2 Preprocesamiento de las señales

Para el preprocesamiento se utilizó la siguiente serie de pasos enumerados a continuación:

1. Se realizó la normalización de todas las señales con respecto a la media.
2. En las señales se eliminó los segmentos correspondientes a los giros mediante su identificación en los gráficos temporales de los archivos; para eso, utilizamos principalmente las señales del giroscopio para su detección, de esta manera enfocamos la solución a reconocer los patrones relacionados a la marcha.
3. Se realizó la interpolación de las señales con la finalidad de que tengan una frecuencia de muestreo estable a 50 Hz, esto debido a que las señales en los archivos tenían variaciones entre 50 y 51 Hz, lo cual podría afectar al rendimiento de los modelos.
4. Se dividieron los datos de cada archivo en ventanas uniformes de 2000 milisegundos, es decir que cada ventana posee 100 lecturas, esto con la finalidad de enfocar el reconocimiento de los patrones en segmentos más pequeños de la marcha, y tener una mayor cantidad de datos para el entrenamiento de los modelos. Se obtuvieron en total 1895 ventanas para el entrenamiento. De las cuales se separaron 238 ventanas para prueba y 141 para validación.

2.2.1 *Extracción de características*

Luego del preprocesamiento de las señales se extrajo características de las señales de cada ventana, a continuación se unieron las ventanas de los 3 sensores tomando como base al rango de tiempo en la que se encontraban; es decir, por ejemplo, las ventanas de 0 – 2000 ms de

los 3 sensores se unieron en una única entrada de 3 ventanas, finalmente, se seleccionaron las características más relevantes de estas entradas.

Para la extracción de características se utilizó 2 métodos. El primero está basado en el procedimiento propuesto por Aayesha Zia en [28]. Este método consiste en: por cada ventana, las 6 señales del acelerómetro y giroscopio son descompuestas en sub-bandas utilizando la descomposición de la señal a través de la Transformada de Wavelet Discreta (DWT). Se utilizó Daubachies 4 (db4) [28] como la función wavelet y se generaron coeficientes por cada señal, los cuales corresponden a sub-bandas específicas de la señal. Cada conjunto de coeficientes representa la información contenida en una sub-banda, con los coeficientes de detalle en las bandas altas y los coeficientes de aproximación en las bandas bajas [28]. Mediante esta descomposición se obtuvo 4 sub-bandas por cada una de las 6 señales. De cada señal se extrajo 2 tipos de características, características temporales y espectrales. De acuerdo a la estrategia propuesta se extrajo 3 características de tipo temporales por cada una de las 4 sub-bandas de cada una de las 6 señales, que son: la amplitud mínima, la amplitud media y la desviación estándar de la amplitud; así mismo, se extrajo 3 características de tipo espectrales por cada una de las 4 sub-bandas de cada una de las 6 señales, que son: la densidad de potencia máxima, la densidad de potencia mínima y la varianza de la densidad de potencia. Con este método en total se obtuvo 144 características por cada ventana. Uniendo las ventanas de los 3 sensores se obtuvo 432 características que se utilizaron como entrada a los modelos.

El segundo método se basó en extraer características de cada ventana por medio de la librería TSFresh [22], la cual está disponible en Python y permite extraer una variedad de características estadísticas, matemáticas y transformaciones de las series temporales [22]. Con

este método se extrajo 3323 características por cada ventana, lo cual produjo un total de 9969 características tomando en cuenta los 3 sensores.

Para ambos métodos se extrajo las características más relevantes mediante el Análisis de Correlación respecto a la variable objetivo [23], y conservando las características que poseían una correlación > 0.2 o < -0.2, luego se extrajo las características más relevantes por medio del Análisis de Componentes Principales (PCA) [24], y preservando el 98% de la información. Para el primer método resultaron 69 características, mientras que para el segundo 720.

2.2.2 Generación de Escalogramas

También, a partir de las señales preprocesadas, se generó Escalogramas de las señales de cada ventana mediante la transformada Wavelet de tipo Complex Morlet [25], la cual está disponible en la librería PYWT de Python y Matplotlib; al final cada imagen fue redimensionada a un tamaño de 224 x 224 píxeles.

2.2.3 Señales preprocesadas

Se calculó la matriz de correlación entre las señales de los 6 ejes (3 ejes del giroscopio y 3 ejes del acelerómetro), usando datos de 3 de los archivos del conjunto de datos S1 y 3 de los archivos del conjunto de datos S2, para identificar si hay ejes fuertemente correlacionados. No se encontraron ejes con una correlación > 0.3 o < -0.3, por lo que se conservó las señales de los 6 ejes de las ventanas.

2.3 Evaluación y selección de arquitecturas

En esta etapa se realizaron pruebas y comparación con los mejores modelos de predicción determinados, los cuales usan como entrada las características extraídas de las ventanas y los escalogramas generados. Como resultado de cada modelo se obtiene la probabilidad de

clasificación de un sujeto con EP o sin EP. Para la selección del mejor modelo de diagnóstico se compararon 5 métricas de desempeño principales: una matriz de confusión la cual nos permite identificar verdaderos y falsos positivos y negativos. Además, se usó la curva ROC, la cual es una representación de la razón o proporción de verdaderos positivos frente a la razón o proporción de falsos positivos, además se usaron las métricas de exactitud, precisión, y sensibilidad descritos en la sección 2.4.11.

Las características extraídas mediante los métodos DWT y TSFresh son de tipo tabular, por lo que se optó por usar modelos de clasificación basados en ML poco profundos, o tipo shallow, tales como: Support Vector Machines (SVM) , XGBoost (GBM), Multi Layer Perceptron (MLP), Random Forests (RF) o Logistic Regression (LR), K-Nearest-Neighbors (KNN) y Decision Trees (DT), los cuales se describen en las siguientes secciones.

Mientras que, para analizar los escalogramas obtenidos, se utilizó un modelo VGG-16 con transfer learning, heredando los pesos de las capas convolucionales del modelo previamente entrenado por ImageNet [26], congelando las primeras 2 capas convolucionales y generando únicamente la capa de clasificación al final de la red.

Para el análisis de las señales únicamente preprocesadas en los 6 ejes, es decir, sin métodos de extracción de características o generación de escalogramas, se utilizó una red Long Short-Term Memory (LSTM) y un Time-Series Transformer (TST).

2.3.1 Multi Layer Perceptron (MLP)

Un Perceptrón Multicapa (MLP) es una arquitectura de red neuronal artificial compuesta por múltiples capas de neuronas organizadas de forma secuencial. Esta estructura permite a la red aprender y modelar relaciones no lineales complejas en los datos [30].

Arquitectura de una MLP:

1. **Capa de Entrada**: Esta capa recibe las características o variables de entrada del conjunto de datos. Cada una de las neuronas en esta capa corresponde a una característica específica [30].

2. **Capas Ocultas**: Una o varias capas situadas entre la capa de entrada y la capa de salida. Cada una de las neuronas en estas capas aplica una función de activación a una combinación lineal de las salidas de la capa anterior (esta puede ser la capa de entrada o una capa oculta), esto permite a la red capturar patrones complejos en los datos [30].

3. **Capa de Salida**: Genera la respuesta final de la red. El número de neuronas en esta capa y la función de activación utilizada dependen de la del tipo de problema [30].

En la **(Figura 3)** se puede ver un ejemplo de una arquitectura de una red MLP.

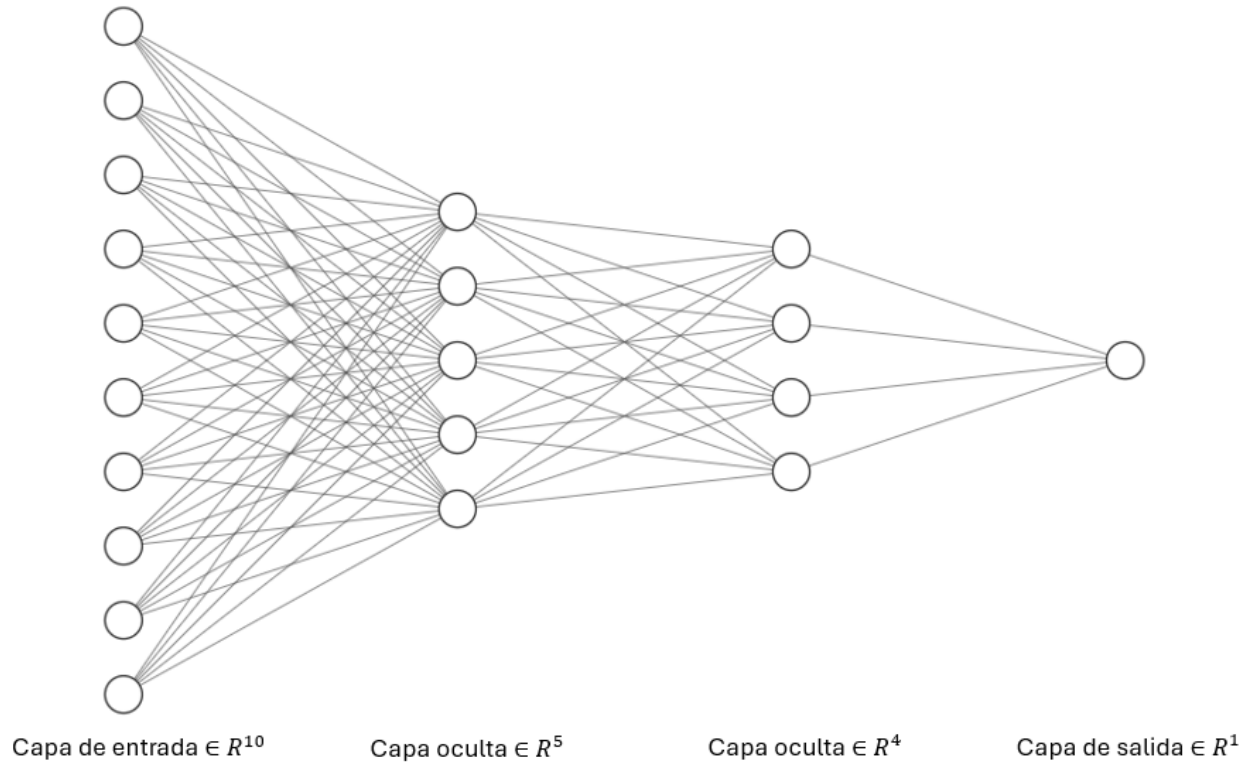

Figura 3. Ejemplo de arquitectura de una red MLP con: una capa de entrada, una capa de salida y dos capas ocultas (Autoría propia)

2.3.2 Support Vector Machine

Las Máquinas de Vectores de Soporte describen un tipo de algoritmo de aprendizaje supervisado centrado en clasificación y regresión, estas organizan los datos en un espacio de *n* dimensiones, y busca un hiperplano que los divida de la mejor manera en sus respectivas clases [37]. La cantidad de dimensiones depende de la cantidad de características utilizadas, por lo que es una buena opción para casos como el nuestro, donde se tiene una alta cantidad de características [37]. (Ver **Figura 4**).

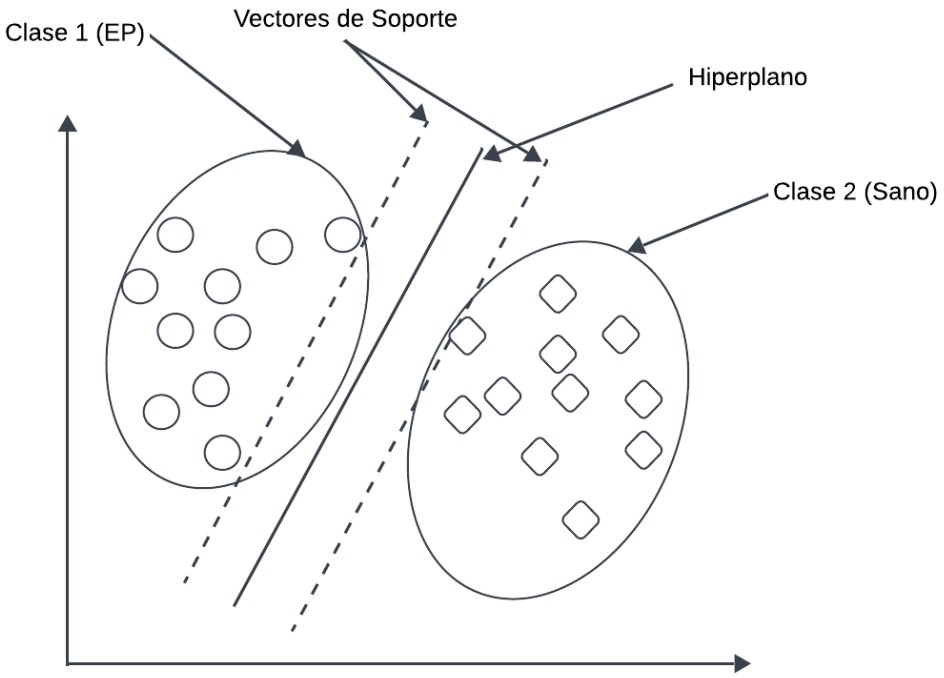

Figura 4. La SVM busca organizar los datos en varias dimensiones de manera que se vean separados por el hiperplano. (Autoría propia)

En la práctica, una SVM tiene varios hiperparámetros que rigen su comportamiento, como el parámetro C, que da un tipo de regularización para balancear la generalización del modelo [37], Gamma, que indica la influencia que se le da a cada valor individual, y el Kernel, que es la función utilizada para transformar los datos en el espacio [37].

En nuestro caso, utilizamos un C de 1.0, un Gamma ‘RBF’, y un Kernel lineal.

2.3.3 Decision Tree

Los árboles de decisión son modelos predictivos que funcionan representando diferentes decisiones y las consecuencias que tendrían de manera jerárquica [40], creando la estructura de un árbol, donde los nodos internos corresponden a divisiones basadas en un atributo, y en el caso de clasificación, las hojas corresponden a la clase correspondiente, utilizando diferentes medidas para tomar la decisión que minimice la pérdida en el modelo [40] (Ver **Figura 5**).

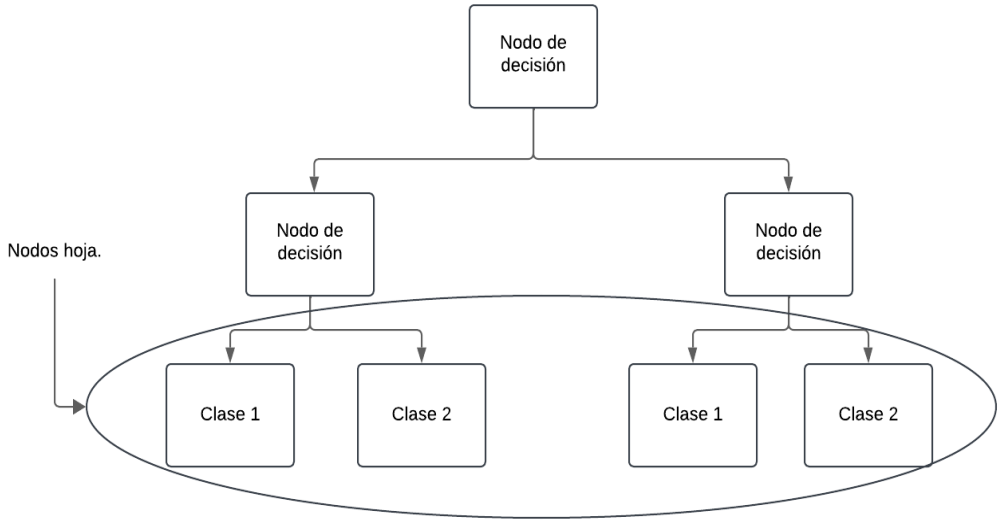

Figura 5. El árbol navega los nodos internos hasta llegar a un nodo hoja, donde decide la clase a elegir. (Autoría propia)

Los hiperparámetros que rigen su comportamiento son los siguientes:

- min_samples_split, que indica el número mínimo de muestras que se requiere para que un nodo se divida [41]. En nuestro caso, lo ajustamos en 10.
- min_samples_leaf, que indica el mínimo número de muestras que una hoja debe tener, así controlando las hojas terminales [41]. En nuestro caso, lo ajustamos a 5.

- max_depth, que indica la profundidad de los árboles y ayuda a minimizar sobreajuste [41]. Lo establecimos en 20 en base a nuestro análisis.
- criterion, que indica la función que se utilizará para identificar la perdida [41]. En nuestro caso ‘entropy’ dio los mejores resultados.

2.3.4 Random Forest

Random Forest es una técnica de aprendizaje supervisado que funciona combinando múltiples árboles de decisión con un muestreo aleatorio de características, donde la idea es que cada árbol sea independiente, votando entre sí para encontrar el mejor resultado final posible, esto le proporciona una mayor resistencia al sobreajuste que los Decision Trees, dado que los árboles son entrenados con sets aleatorios de ejemplos. [40]

Para su entrenamiento, los siguientes hiperparámetros fueron significativos:

- n_estimators, que indica la cantidad de árboles a generar en el bosque. [40] Obtuvimos el mejor resultado con un valor de 100.
- min_samples_split, que indica el número mínimo de muestras que se requiere para que un nodo se divida. [40] En nuestro caso, resultó en 2.
- min_samples_leaf, que indica el mínimo numero de muestras que una hoja debe tener, así controlando las hojas terminales. [40] Para nosotros, 1 resultó el tamaño ideal.
- max_depth, que indica la profundidad de los árboles y ayuda a minimizar sobreajuste. [40] Resultó en ‘none’ en nuestro análisis, es decir, no se limitaba.

2.3.5 Regresión Logística

La Regresión Logística es un modelo predictivo que tiene una gran utilidad en tareas de clasificación, especialmente si la variable de salida, como la nuestra, es binaria. [42] El modelo utiliza la función Logística (o Sigmoide) para transformar una combinación de las características de nuestros datos en una probabilidad. [42]

Los parámetros que optimizamos para este modelo fueron los siguientes:

- C, que controla la fuerza de regularización aplicada al modelo. [42] 10 en nuestro caso.
- penalty, que indica el tipo de regularización a utiliza. [42] L2, para nosotros.
- solver, que indica el método de ajuste del modelo. [42] 'liblinear' para nosotros.

2.3.6 XGBoost

XGBoost es una biblioteca de aprendizaje supervisado basado en la técnica de Gradient Boosting, similar Random Forest, utiliza varios árboles de decisión, pero con un enfoque distinto, ya que, en lugar de hacerlo de manera aleatoria, los genera de manera secuencial, permitiendo que cada uno corrija los errores del anterior [43]. Otra diferencia que tiene con Random Forest, es que su función de perdida incorpora una regularización que controla la complejidad de los árboles y por lo tanto permite la "pre-poda" de las ramas de los árboles [43]. Además, incluye varios métodos de generación aleatoria y muestreo, que permiten minimizar el overfitting y dar un entrenamiento más acelerado. [43]

Esto se presenta en la práctica en varios hiperparámetros, para nuestro modelo evaluado se optimizaron los siguientes :

- learning_rate (o shrinkage), que controla el tamaño de los pasos de actualización. [43] En nuestro caso, un valor de 0.1.
- gamma**,** que regula la profundidad de los árboles al establecer la reducción mínima de pérdida necesaria para dividir un nodo. [43] Para nosotros un valor de 0.
- max_depth**,** que limita la profundidad de los árboles. [43]. Resultando en nuestro caso, 3.
- colsample_bytree, que controla la fracción de características (o columnas) que se seleccionan de manera aleatoria para entrenar cada árbol en el ensamble. [43] Para nuestro modelo, 1.0.
- n_estimators, que especifica el número total de árboles que se construirán en el modelo. [43] Resultó en 200 para nosotros en base a nuestro análisis.

2.3.7 K-Nearest Neighbors

K-NN es un algoritmo de clasificación que es conocido por ser más simple y versátil para tareas de clasificación, siendo adecuado para manejar relaciones complejas y no lineales en datos de alta dimensionalidad. Este funciona por medio de situar los datos en un plano, basado en sus características, y medir la distancia a los demás, decidiendo su clase basado en los datos que se encuentren más cercanos [44]. Se puede visualizar un ejemplo de clasificación en la (**Figura 6**).

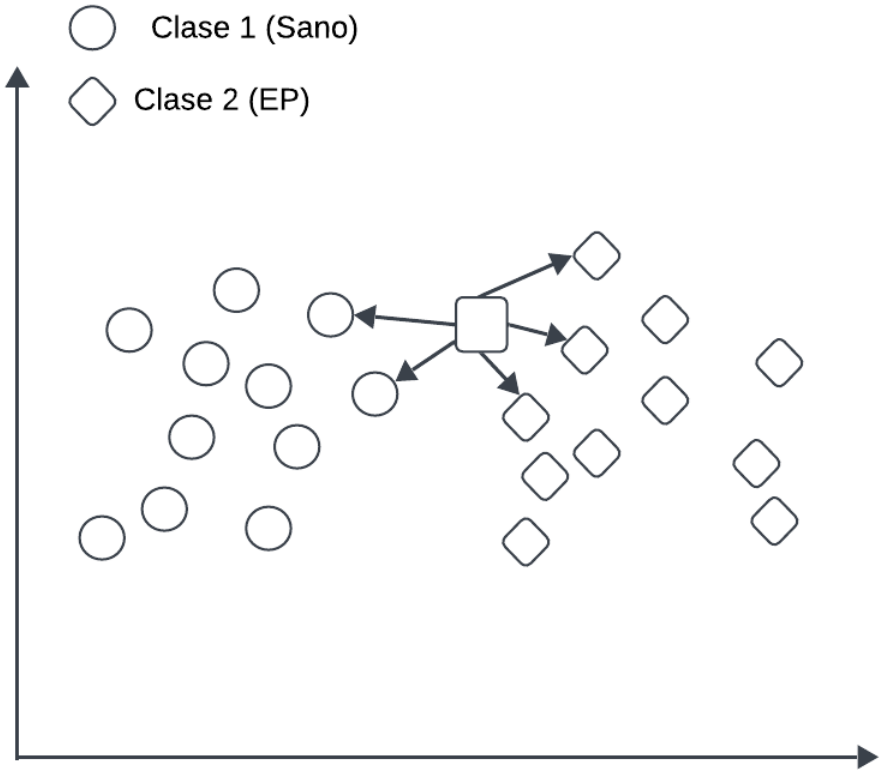

Figura 6. Ejemplo donde el ejemplo a clasificar encuentra a sus 5 vecinos más cercanos, la mayoría siendo de Clase 2, por lo que será clasificado a esta misma clase. (Autoría propia.)

En nuestro caso, los hiperparámetros relevantes fueron:

- algorithm, que indica cual algoritmo se utilizará para encontrar los vecinos más cercanos. [44] En este caso, ‘auto’.
- p, que indica el tipo de distancia a utilizar, L1 (Manhattan), o L2 (Euclidiana). [44] Para nosotros, fue L1
- weights, que define la ponderación de los vecinos al momento de decidir la clasificación, es decir, cuales vecinos son considerados más. [44] Para nosotros, ‘distance’, es decir, que mientras más cercanos los vecinos, mayor ponderación tendrán.

- n_neighbors, que indica la cantidad de vecinos más cercanos que serán considerados al clasificar. [44]. Para nosotros resultó 9.

2.3.8 VGG-16 con Transfer-learning

La VGG-16 es una red neuronal convolucional profunda. Es conocida por su arquitectura simple pero potente y se ha utilizado ampliamente en visión por computadora [33].

Cuando se combina con transfer learning, la VGG-16 se puede adaptar fácilmente a nuevas tareas sin necesidad de entrenar toda la red desde cero, esto ahorra tiempo y recursos computacionales [33]. Se puede visualizar la arquitectura de una VGG-16 en la (**Figura 7**).

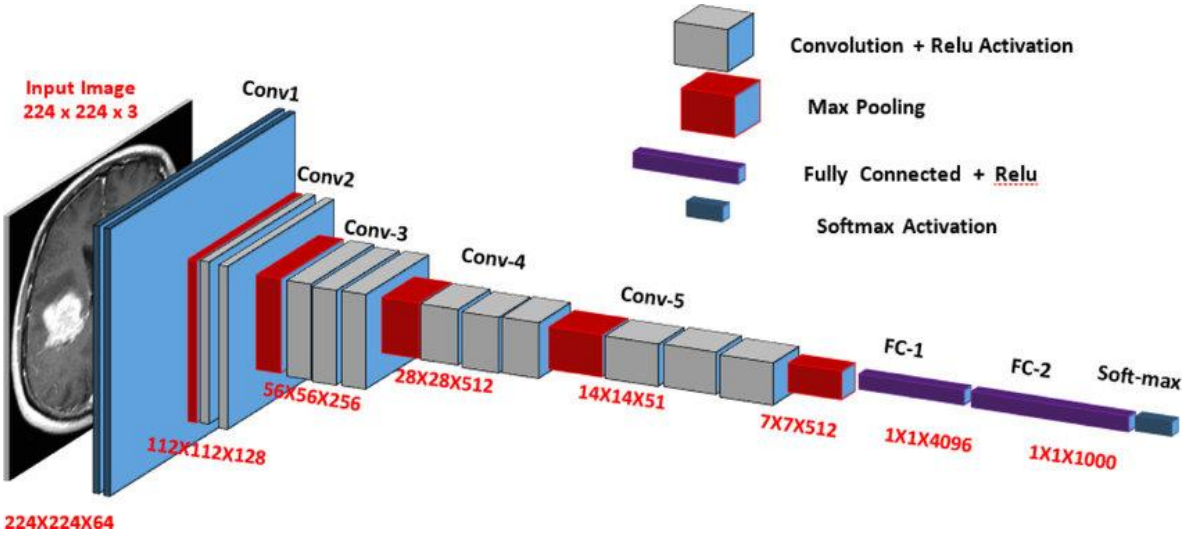

Figura 7. Arquitectura de una VGG-16 (Fuente: [33])

2.3.9 Long Short-Term Memory

Una red Long Short-Term Memory (LSTM) es un tipo de red neuronal recurrente (RNN) especialmente diseñada para trabajar con secuencias de datos, como series temporales, texto o audio [34]. Las LSTMs son capaces de aprender dependencias a largo plazo en los datos, resolviendo el problema del "desvanecimiento del gradiente" que afecta a las RNNs estándar [34]. En la (**Figura 8**) se puede ver el diagrama de una celda de una red LSTM.

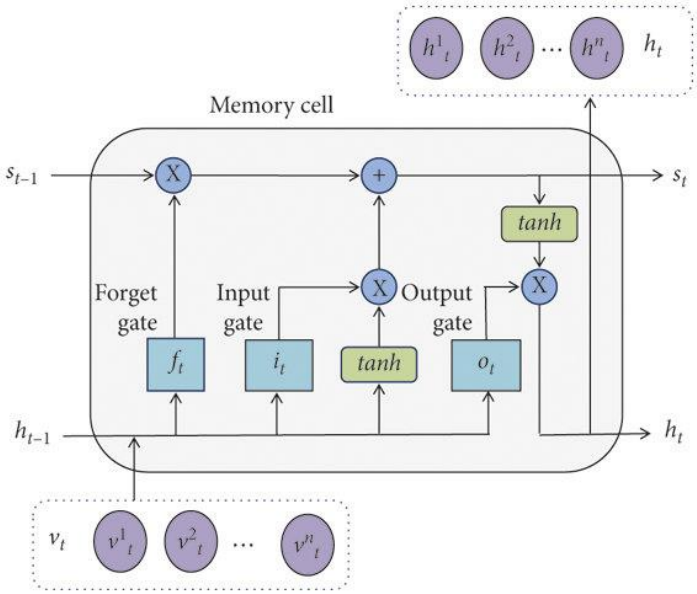

Figura 8. Diagrama de la representación de una celda de una red LSTM con sus 3 puertas: puerta de olvido, entrada y salida (Fuente: [35])

2.3.10 Time-Series Transformer (TST)

Un Transformer diseñado específicamente para datos de series temporales. Su arquitectura incluye mecanismos de atención optimizados para procesar secuencias de datos numéricos. Esta captura dependencias a largo plazo entre los puntos de la señal [36]. Es robusto

para manejar entradas multivariadas (múltiples características por paso temporal), como en nuestro caso en el cual existen 6 señales por paso temporal [36] (Ver **Figura 9**).

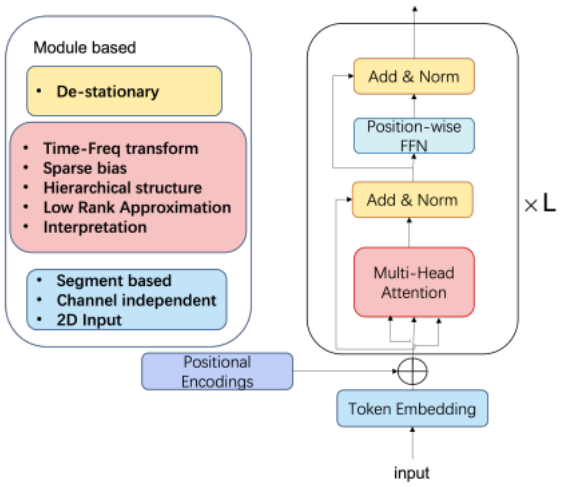

Figura 9. Categorización de variante de transformer a nivel módulo, para la predicción basada en series de tiempo. (Fuente: [36])

2.3.11 Generación de métricas de evaluación

Para evaluar los modelos anteriormente descritos y realizar una comparación del mejor modelo, el cual se utilizará en la aplicación a desarrollar, se obtuvo las siguientes métricas, tal como se describen en la sección 2.4.1.

En las ecuaciones a continuación, cada variable representa:

- TP: Verdadero positivo.

- TN: Verdadero negativo.

- FP: Falso positivo.

- FN: Falso negativo.

Matriz de confusión: Esta es una representación matricial de los resultados de cada una de las predicciones, lo que permite analizar falsos y verdaderos positivos y negativos. Como se describe en [29], esta matriz muestra la cantidad de los verdaderos positivos, falsos positivos, verdaderos negativos y falsos negativos. Para obtener la Exactitud [29] de cada uno de los modelos, se evaluó los resultados usando la **Ecuación (2.1)**, esta ecuación se encuentra en términos de las clases positivas y negativas y mide la proporción de predicciones correctas realizadas por el modelo con respecto al total de predicciones.

$$Exactitud = \frac{TF + TN}{TF + TN + FN + FP} \qquad (2.1)$$

Area Under the Curve - Receiver Operating Characteristic: El AUC-ROC es una métrica utilizada para evaluar el rendimiento de modelos de clasificación binaria [30]**.**

ROC (Receiver Operating Characteristic): Esta es una curva que representa la relación entre la tasa de verdaderos positivos (TPR, sensibilidad o recall) y la tasa de falsos positivos (FPR) [30].

AUC (Area Under the Curve):

- Es el área bajo la curva ROC. Representa la capacidad general del modelo para distinguir entre clases positivas y negativas [30].
- El valor del AUC varía entre 0 y 1 [30]:
  - **1**: Perfecto. El modelo clasifica correctamente todas las clases.
  - **0.5**: No mejor que un clasificador aleatorio.
  - **< 0.5**: El modelo está clasificando peor que aleatoriamente.

Ejes de la curva ROC:

1. **Eje Y (TPR, sensibilidad o recall)**: Para obtener la TPR de cada uno de los modelos se utilizó la **Ecuación (2.2)**. La TPR mide la proporción de instancias positivas correctamente clasificadas con respecto al total de instancias positivas reales en el conjunto de datos de prueba [30].

$$TPR = \frac{TP}{FN + TP} \quad (2.2)$$

2. **Eje X (FPR)**: Para obtener la FPR de cada uno de los modelos se usa la **Ecuación (2.3)**. La FPR mide qué tan propenso es un modelo a clasificar erróneamente instancias negativas como positivas [30].

$$FPR = \frac{FP}{FP + TN} \quad (2.3)$$

Precisión : Para obtener la Precisión de cada uno de los modelos se usa la **Ecuación (2.4)**. La Precisión mide qué proporción de las predicciones positivas realizadas por el modelo son realmente correctas [30].

$$Precision = \frac{TP}{TP + FP} \quad (2.4)$$

2.4 Desarrollo de prototipo

En esta sección se describe el funcionamiento del producto final, los actores y la arquitectura del sistema.

2.4.1 Actores del sistema

En base a la problemática a resolver, donde se analizó en conjunto con el cliente y el tutor de este proyecto mediante el análisis de prototipos de bajo nivel e ideas de solución, se establecieron 2 tipos de roles:

2.4.1.1 Actor Médico. Este actor representa a los médicos o especialistas que utilizaran la aplicación web, para en base a datos de sensores inerciales de un paciente, poder realizar un diagnóstico de la EP. Este actor puede realizar las siguientes acciones en el sistema:

- Cargar archivos de señales de sensores IMU de un sujeto.
- Previsualizar las señales de los datos cargados.
- Ejecutar procesos para el preprocesamiento de los datos antes de la predicción.
- Ejecutar el proceso de predicción con los datos de señales cargados.
- Visualizar los resultados de la predicción con su probabilidad y el diagnóstico en base a la predicción.
- Mostrar observaciones en base a los resultados obtenidos.
- Enviar observaciones sobre la predicción realizada.

2.4.1.2 Actor Administrador. Este actor es el encargado de supervisar el sistema y su correcto funcionamiento, la administración de usuarios, archivos subidos y el modelo integrado en el sistema. Posee las siguientes funcionalidades en el sistema:

- Administrar usuarios con rol de médico y otros administradores.
- Visualizar el historial de predicciones realizadas con su información y los archivos cargados.
- Descargar los archivos cargados, usados para predicciones anteriores.

2.4.2 Diseño del prototipo

2.4.2.1 Tecnologías a utilizar. El sistema está compuesto por un backend y un frontend, los cuales se describen a continuación:

El frontend esta desarrollado con React.js, el cual es un framework de código abierto de JavaScript que se utiliza para crear interfaces de usuario web y nativas, y el backend mediante Django, el cual es un Framework escrito en Python que posee funcionalidades ya integradas sobre administración de usuarios, se usará una base de datos SQLite para manejo de información no volátil, esta base de datos viene integrada con el Framework Django.

2.4.2.2 Diagramas de la arquitectura. En esta sección se describen los diagramas que explican los requerimientos funcionales y arquitectura del sistema.

Se realizó un diagrama de casos de uso que explica las funcionalidades que pueden realizar los dos actores del sistema descritos en la sección 2.5.1 (Ver **Figura 10**).

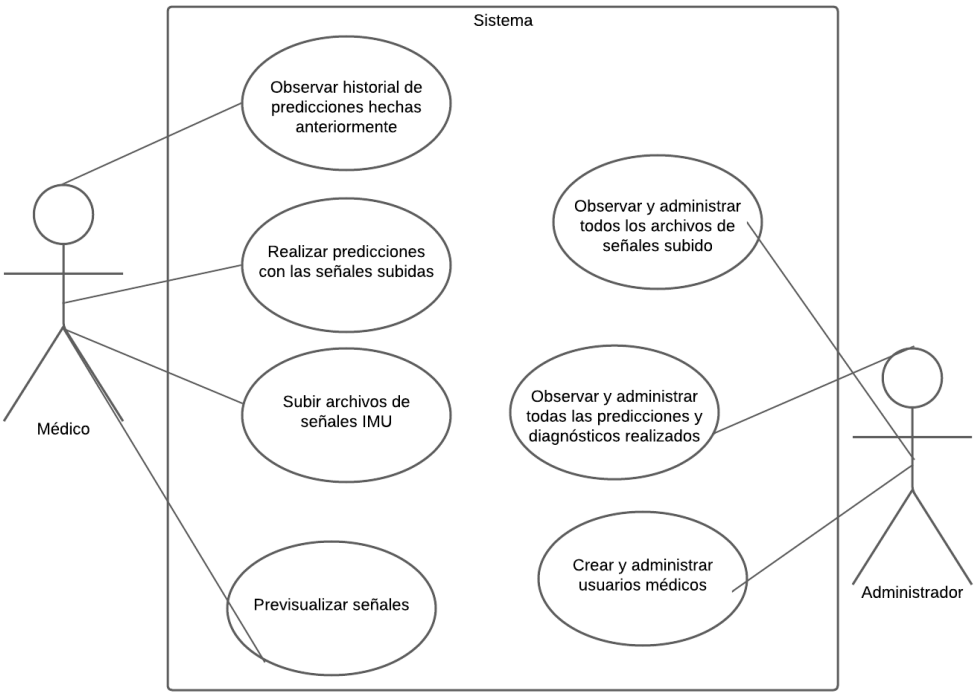

Figura 10. Diagrama de casos de uso del sistema (Autoría propia)

Se realizaron diagramas de 4+1 [39] vistas explicando el funcionamiento del sistema para el usuario médico.

Vista Lógica: En este diagrama se visualiza el funcionamiento lógico del programa, y cómo los componentes grandes interactuan (Ver **Figura 11**).

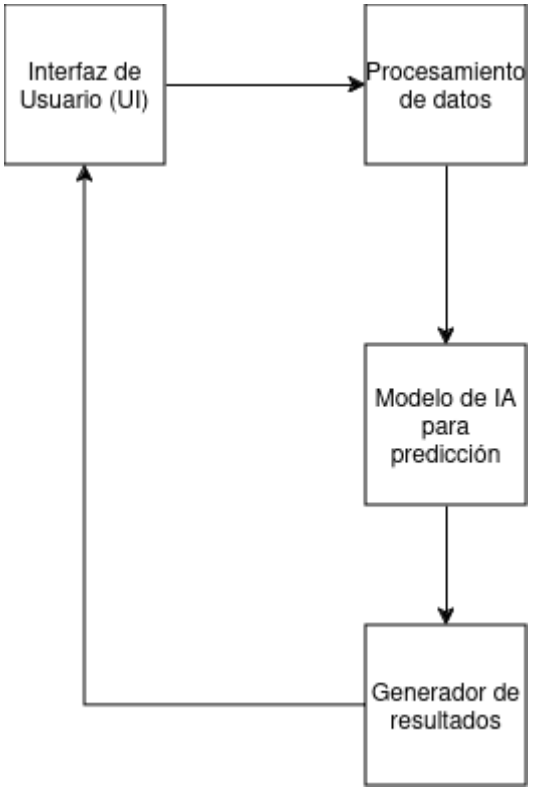

Figura 11. Vista lógica del sistema (Autoría propia)

Vista de procesos: En este diagrama se puede ver una secuencia de los procesos que tienen lugar en el uso de la aplicación (Ver **Figura 12**).

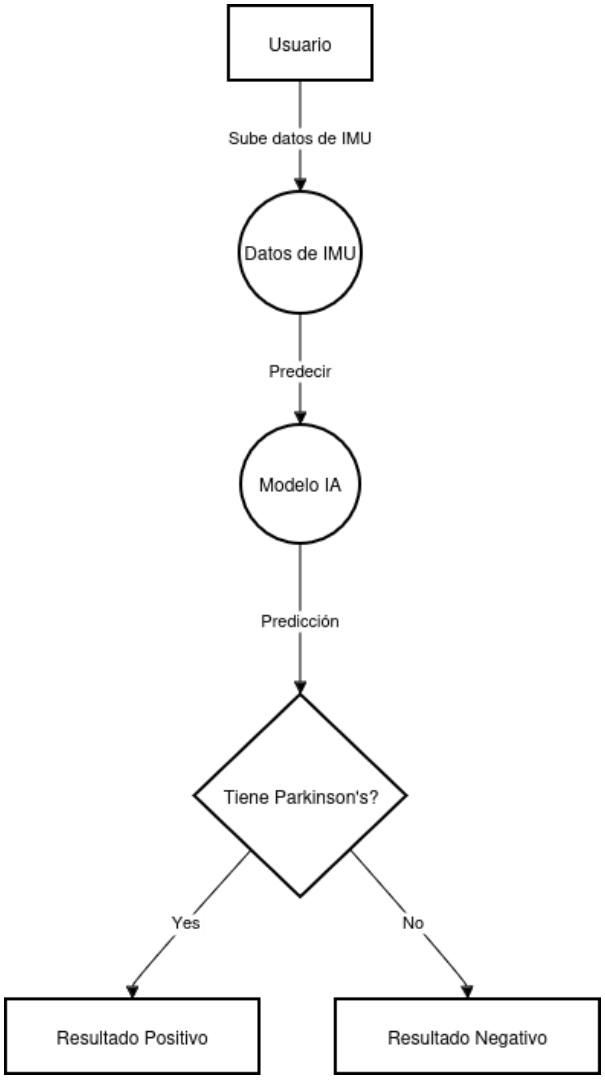

Figura 12. Vista de procesos del sistema (Autoría propia)

Vista de desarrollo: En este diagrama se puede ver desde una vista de desarrollo los componentes a ser realizados y la clasificación de los mismos (Ver **Figura 13**).

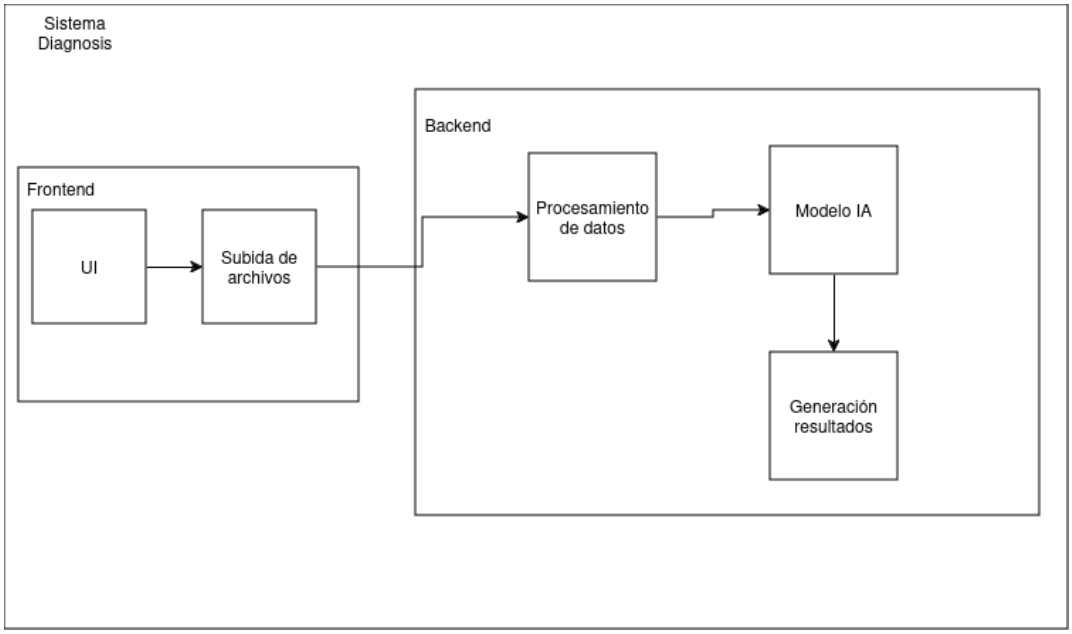

Figura 13. Vista de desarrollo del sistema (Autoría propia)

Vista Física: En este diagrama se pueden visualizar los componentes físicos del sistema y su interacción (Ver **Figura 14**).

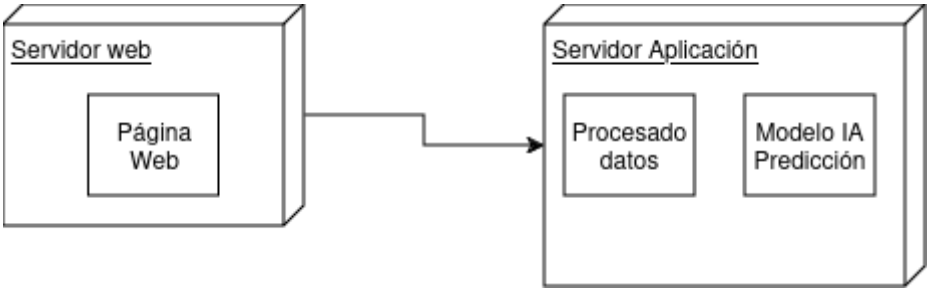

Figura 14. Diagrama de despliegue del sistema (Autoría propia)

2.4.3 Prototipo diseñado

En esta etapa se presenta el prototipo diseñado en la plataforma Figma [38], el cuál consta de los dos tipos de usuarios (médico y administrador) con sus funcionalidades y especificaciones descritas en la sección 2.5.1. Este prototipo muestra las funcionalidades principales que el sistema desarrollado tendrá.

El enlace web a este prototipo desplegado en Figma se muestra en [ANEXO 1].

Las pantallas principales del sistema se pueden observar en (**Figura 15**) y (**Figura 16**).

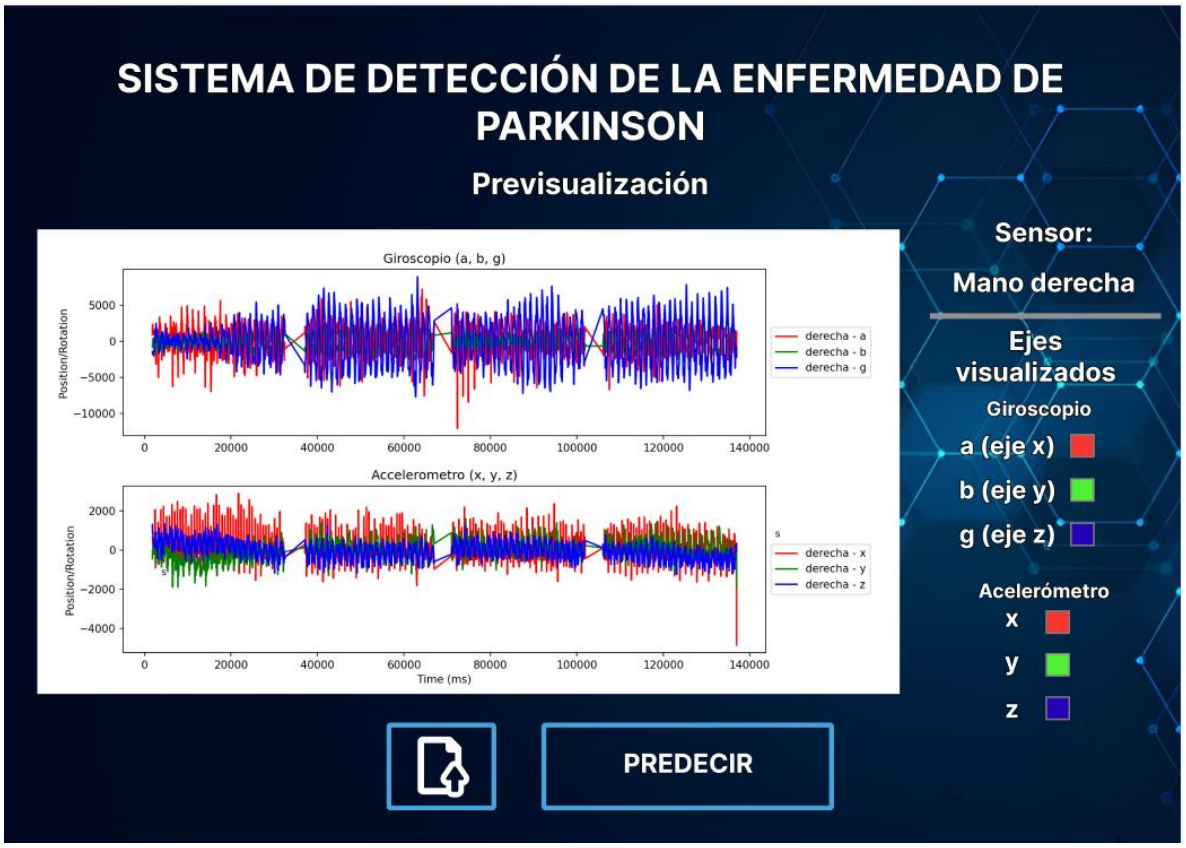

Figura 15. Archivos de señales IMU subidos por el usuario médico, previsualizados en la aplicación (Autoría propia)

Figura 16. Descripción de la predicción realizada por el usuario médico en la aplicación

(Autoría propia)

Capítulo 3

3. Resultados y análisis

En esta sección se detalla los resultados obtenidos en la evaluación de los modelos descritos en la sección 2.4 con su respectivo análisis.

3.1 Entornos de desarrollo

Los experimentos llevados a cabo se realizaron en 2 entornos diferentes descritos a continuación:

El primer entorno fue utilizando el IDE PyCharm Community Edition 2023.1.2. Los experimentos realizados en este entorno fueron realizados en una computadora con CPU Intel Core I5 vPro inside y 8GB de RAM, sin GPU. En este entorno se realizó la implementación, entrenamiento y pruebas del modelo de Red MLP con los 2 métodos de extracción de características descritos en la sección 2.3.1

El segundo entorno fue utilizando Jupyter Notebook, en un entorno virtual con la supercomputadora de CEDIA, donde se utilizó 16 cores de CPU, 16 GB de RAM, y un GPU Nvidia A100 Mig 1g.5GB. En este entorno se realizó la implementación, entrenamiento, y pruebas de los modelos SVM, KNN, GBM, Decision Tree, Random Forest, Regresión Logística, Transformer, LSTM.

Las librerías utilizadas en Python, para el entrenamiento e implementación de los modelos fueron Keras 3.8.0, Tensorflow 2.18 y Scikit-Learn

3.2 Datos de entrenamiento para los modelos

Se generaron 3 tipos de entrada para el entrenamiento de los modelos explicados en la sección 2.4.5: mediante métodos de extracción de características, la generación de imágenes

de escalogramas, y las señales temporales con un preprocesamiento previo de ICESI y normalización.

3.2.1 Métodos de extracción de características

Para el entrenamiento de los modelos ML Shallow (Red MLP, Random Forest, Decision Tree, SVM, Regresión Logística, XGBoost y K-NN) descritos en la sección 2.4, se utilizaron las características extraídas de las señales de los datos.

Se usaron 2 métodos diferentes para la extracción de características como se explicó en la sección 2.3.1, el primer método extrajo características de las subbandas de cada señal a través de una DWT, y el segundo método extrajo características de las señales a través de la librería TSFresh; luego, se seleccionaron las características más relevantes generadas a través de los métodos de selección de características (Ver **Tabla 1**).

Tabla 1: Cantidad de características extraídas mediante cada método.

Método	Datos
DWT	Se extrajeron un total de 432 características a través del método, mediante los métodos de correlación y PCA para seleccionar las características más relevantes para predicción resultando en un total de 69 componentes finales.
TSFresh	Se extrajeron un total de 9969 características a través del método, mediante los métodos de correlación y PCA para seleccionar las características más relevantes para predicción resultaron un total de 720 componentes.

3.2.2 Generación de imágenes de escalogramas

Para el entrenamiento de la VGG-16 con Transfer Learning, se usó los escalogramas generados de las señales descritas en la sección 2.3.2, resultando cada entrada en imágenes RGB de 224x224 píxeles.

3.2.3 Señales temporales

Para el entrenamiento de los modelos LSTM y TS Transformer se utilizó las series temporales de las señales con el preprocesamiento hecho de ICESI y su normalización, tal como se describe en la sección 2.3.3. Resultando cada entrada en matrices de 18x100, donde cada columna es una lectura en cada uno de los 6 ejes de los 3 sensores en un tiempo específico, y al ser entradas de ventanas de 2 segundos con un frecuencia de muestreo uniforme de 20ms cada lectura, se obtuvo un total de 100 lecturas por cada ventana.

3.3 Experimentos efectuados

3.3.1 Resultados de modelos entrenados con los métodos de extracción de características

Para el entrenamiento y la implementación de los modelos ML Shallow, se realizó una búsqueda iterativa de los hiperparámetros de las diferentes arquitecturas, manteniendo la combinación que produjo mejores resultados en la evaluación de cada modelo, los mejores resultados obtenidos se muestran en la (**Tabla 2**).

Tabla 2: Resultados de evaluación de modelos de ML Shallow

MODELO	MÉTODO	EXACTITUD
SVM	DWT (Selección de características)	0.9088
Random Forest	DWT (Selección de características)	0.9509
MLP	DWT (Selección de características)	0.9523
Decision Tree	DWT (Selección de características)	0.8675
Logistic Regression	DWT (Selección de características)	0.6384
XGBoost	DWT (Selección de características)	0.9505
K-NN	DWT (Selección de características)	0.7302

SVM	TSFresh (Selección de características)	1
Random Forest	TSFresh (Selección de características)	1
MLP	TSFresh (Selección de características)	1
Decision Tree	TSFresh (Selección de características)	1
Logistic Regression	TSFresh (Selección de características)	1
XGBoost	TSFresh (Selección de características)	1
K-NN	TSFresh (Selección de características)	1

Se puede visualizar en la (**Tabla 2**), que mediante el método de extracción de características de TSFresh, los 7 modelos de ML Shallow exhiben una exactitud muy alta en la evaluación, esto podría ser un indicio de sobreajuste en los modelos, posiblemente por una elevada cantidad de características.

Mediante el método de DWT, se puede ver que el rendimiento de los modelos varía, siendo el de mayor exactitud la red MLP con un 95.23% de predicciones correctas y el de menor rendimiento, el modelo de Regresión Logística con un 63.84% de predicciones correctas.

3.3.2 Resultados de modelo entrenado con escalogramas de los datos

Para la prueba del modelo VGG-16 con Transfer Learning, en la que se utilizaron los escalogramas de las señales, los resultados se pueden visualizar en la (**Tabla 3**).

Estos resultados fueron obtenidos guardando el modelo con la menor cantidad de pérdida con los datos de validación durante el entrenamiento.

Tabla 3: Resultados de modelo VGG-16 con Transfer Learning

Arquitectura	**Exactitud**	**Pérdida**
VGG-16 con Transfer Learning	0.7016	0.8675

3.3.3 Resultados de modelos entrenados con las series temporales de las señales

Para la evaluación de arquitecturas en las que se utilizaron series temporales, se usaron las librerías Keras y Tensorflow. Las arquitecturas seleccionadas, tal como se revisaron en la sección 2, fueron: LSTM y TS Transformer.

En este caso, se utilizaron ventanas de 2 segundos para la evaluación, con 350 épocas para cada arquitectura, y se hicieron pruebas también empleando Early Stopping [45], para detener el entrenamiento una vez que la pérdida deja de disminuir en cierta cantidad de épocas; sin embargo, al entrenar el modelo con las épocas completas, los resultados se vieron similares (Ver **Tabla 4**).

Tabla 4: Resultados de modelos entrenados con las señales temporales de los datos

Arquitectura	**Épocas**	**Ventana de Tiempo**	**Exactitud**	**Pérdida**
LSTM	350	2s	0.9102	0.4025
TS Transformer			0.8405	0.3236

3.4 Resultados finales

Con estos resultados, y considerando que todos los modelos fueron probados con varios conjuntos de datos que no estuvieron presentes en su entrenamiento, el modelo que mejor resultados ofreció, y por lo tanto fue elegido como modelo para el sistema, es la red MLP con características extraídas mediante el método DWT, para una evaluación más profunda, generamos una matriz de confusión y extrajimos las métricas de Precisión, Recall, y AUC-ROC del modelo:

La matriz de confusión obtenida con los datos de prueba se puede ver en la **(Figura 17**). Como se describió en la sección 2.2, hay un total de 238 datos de prueba, dividido en 107 para la clase 0 (Sano), y 131 para la clase 1 (EP).

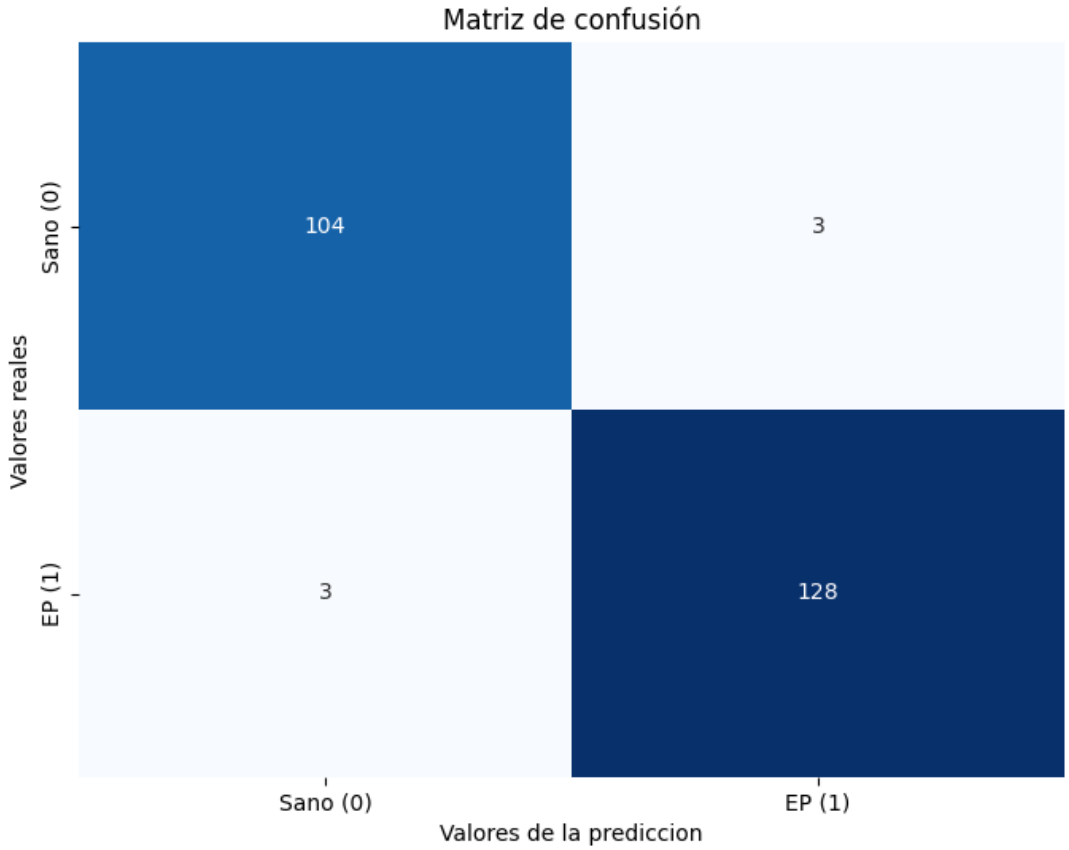

Figura 17. Matriz de confusión del modelo MLP con extracción de características mediante DWT

El valor de las métricas obtenidas se pueden ver en la (**Tabla 5**)

Tabla 5: Resultados de modelos entrenados con las señales temporales de los datos

Modelo/Métrica	**Precisión**	**Recall**	**AUC-ROC**
MLP	0.9616	0.9726	0.9606

Capítulo 4

4. Resultados y análisis

Para desarrollar este proyecto, se utilizaron varios archivos de señales inerciales capturadas y provistas por la universidad ICESI de Cali, Colombia, que incluían datos de sensores IMU ubicados en las dos muñecas y espina base durante la marcha de sujetos tanto sanos, como pacientes con EP. Estos datos fueron analizados, preprocesados, y utilizados para el entrenamiento y pruebas posteriores de los 10 modelos de ML Shallow y DL mencionados en este trabajo.

En cuanto al preprocesamiento, los archivos fueron analizados, normalizados, y recortados para remover ruido, luego se dividieron los archivos de datos en ventanas para aumentar así la cantidad de datos disponibles. Adicionalmente a las señales temporales de las ventanas, se extrajeron características de estos datos, y escalogramas para la evaluación de estos métodos con sus respectivas arquitecturas.

Para este proyecto, los modelos de aprendizaje tipo Shallow implementados fueron XGBoost, Decision Tree, Random Forest, Support Vector Machine (SVM), Regresión Logística, K-Nearest Neighbors (KNN) y Multi Layer Perceptron. Para los de tipo Deep Learning, se utilizaron las arquitecturas Time Series Transformer, Long Short-Term Memory (LSTM), y Convolutional Neural Network (CNN). Posterior al entrenamiento, se evaluaron todos los modelos mencionados mediante un conjunto de métricas y los datos de prueba reservados, y se encontró al que tuviera el mayor desempeño.

Con el mejor modelo encontrado, se desarrolló una aplicación web, capaz de recibir archivos de señales inerciales, preprocesarlos, y utilizar el modelo de IA para

detectar la enfermedad, y permitir que un doctor realice un diagnóstico con la ayuda de la herramienta.

4.1 Conclusiones

- Se generaron 3 tipos de entradas para los modelos de IA, se generaron entradas a través de dos métodos para extraer características de las señales (el método por DWT y el método por TSFresh), un método para generación de escalogramas de las señales y la entrada de las señales temporales preprocesadas, obteniendo que el método de extracción de características de las señales por DWT tuvo el mejor desempeño en los modelos evaluados.
- Se realizó un análisis comparativo del desempeño entre distintos modelos tipo Shallow, así como el desempeño de los distintos modelos tipo DL, obteniendo el modelo con mayor desempeño el de tipo ML Shallow, la red MLP, y el modelo con mayor desempeño DL, el modelo LSTM.
- Se desarrolló una aplicación web para uso médico, en la que se encuentra integrado el modelo con el mejor desempeño y el proceso para generar las entradas al modelo a partir de las señales de sensores inerciales ubicados en las muñecas y la espina base de los pacientes, para así poder detectar si una persona padece o no EP en base a los patrones en la marcha. Esta aplicación posee dos tipos de usuarios, un usuario médico y un administrador, donde el usuario médico posee acceso a la interfaz para realizar la predicción en base a un archivo de señales IMU que desee subir, y el usuario administrador tiene acceso a los archivos de señales subidos a la

base de datos para predicción y a la creación, modificación y eliminación de usuarios y registros en la base de datos.

4.2 Recomendaciones

- Se recomienda que el sistema sea usado bajo la supervisión de un especialista encargado, tomando en cuenta que los resultados obtenidos durante la predicción, sean considerados junto con otros exámenes clínicos y pruebas diagnósticas estándar, para un diagnóstico preciso de la enfermedad.
- Los datos de los sensores IMU que se ingresen al sistema para realizar la predicción, se recomiendan ser capturados únicamente en marchas en línea recta de los pacientes, en las mismas unidades que se encontraban los datos de entrenamiento utilizados, para una predicción mas precisa.
- Para trabajos futuros, se puede ampliar la cantidad de datos usados para predicción, que contemple diferentes edades, demografía, géneros o etnias de las personas. Así mismo se podría incrementar el número de clases de clasificación del modelo, basado en alguna escala estándar, como Hoehn y Yahr, para así tener información más exacta sobre el estado de la enfermedad en un paciente.

Referencias

[1] R. B. Postuma and J. Montplaisir, “Predicting Parkinson’s disease – why, when, and how?,” *Parkinsonism & Related Disorders*, vol. 15, pp. S105–S109, 2009, doi: https://doi.org/10.1016/S1353-8020(09)70793-X.

[2] T. Steinmetzer, M. Maasch, I. Bönninger, and C. M. Travieso, “Analysis and Classification of Motor Dysfunctions in Arm Swing in Parkinson’s Disease,” *Electronics*, vol. 8, no. 12, 2019, doi: 10.3390/electronics8121471.

[3] D. Rincón *et al.*, “Wristbands Containing Accelerometers for Objective Arm Swing Analysis in Patients with Parkinson’s Disease,” *Sensors*, vol. 20, no. 15, 2020, doi: 10.3390/s20154339.

[4] C. Caramia *et al.*, “IMU-Based Classification of Parkinson’s Disease From Gait: A Sensitivity Analysis on Sensor Location and Feature Selection,” *IEEE Journal of Biomedical and Health Informatics*, vol. 22, no. 6, pp. 1765–1774, 2018, doi: 10.1109/JBHI.2018.2865218.

[5] D. Rincón and A. Navarro, “Arm Swinging Measurement and Monitor System for Patients Diagnosed with Parkinson’s Disease.,” in *SSN*, 2018, pp. 53–56.

[6] C. D. Marsden, “Parkinson’s disease.,” *Journal of Neurology, Neurosurgery & Psychiatry*, vol. 57, no. 6, pp. 672–681, 1994, doi: 10.1136/jnnp.57.6.672.

[7] J. Jankovic, “Parkinson’s disease: clinical features and diagnosis,” *Journal of Neurology, Neurosurgery & Psychiatry*, vol. 79, no. 4, pp. 368–376, 2008, doi: 10.1136/jnnp.2007.131045.

[8] F. Magrinelli *et al.*, “Pathophysiology of Motor Dysfunction in Parkinson’s Disease as the Rationale for Drug Treatment and Rehabilitation,” *Parkinson’s Disease*, vol. 2016, no. 1, p. 9832839, 2016, doi: https://doi.org/10.1155/2016/9832839.

[9] C. Caramia *et al.*, “IMU-Based Classification of Parkinson’s Disease From Gait: A Sensitivity Analysis on Sensor Location and Feature Selection,” *IEEE Journal of Biomedical and Health Informatics*, vol. 22, no. 6, pp. 1765–1774, 2018, doi: 10.1109/JBHI.2018.2865218.

[10] J. Zhao, “A Review of Wearable IMU (Inertial-Measurement-Unit)-based Pose Estimation and Drift Reduction Technologies,” *Journal of Physics: Conference Series*, vol. 1087, no. 4, p. 042003, Sep. 2018, doi: 10.1088/1742-6596/1087/4/042003.

[11] C.-H. Lin, F.-C. Wang, T.-Y. Kuo, P.-W. Huang, S.-F. Chen, and L.-C. Fu, “Early Detection of Parkinson’s Disease by Neural Network Models,” *IEEE Access*, vol. 10, pp. 19033–19044, 2022, doi: 10.1109/ACCESS.2022.3150774.

[12] I. APDM, *Opal User Guide*. Accessed: Nov, 2024.

[13] L. Sigcha *et al.*, “Deep learning and wearable sensors for the diagnosis and monitoring of Parkinson’s disease: A systematic review,” *Expert Systems with Applications*, vol. 229, p. 120541, 2023, doi: https://doi.org/10.1016/j.eswa.2023.120541.

[14] F. Cavallo, A. Moschetti, D. Esposito, C. Maremmani, and E. Rovini, “Upper limb motor pre-clinical assessment in Parkinson’s disease using machine learning,” *Parkinsonism & Related Disorders*, vol. 63, pp. 111–116, 2019, doi: https://doi.org/10.1016/j.parkreldis.2019.02.028.

[15] R. Rees, A. Acharya, A. Schrag, and A. Noyce, “An early diagnosis is not the same as a timely diagnosis of Parkinson’s disease [version 1; peer review: 2 approved] ,” *F1000Research*, vol. 7, no. 1106, 2018, doi: 10.12688/f1000research.14528.1.

[16] M. L. Hacker *et al.*, “Deep brain stimulation in early-stage Parkinson disease,” *Neurology*, vol. 95, no. 4, pp. e393–e401, 2020, doi: 10.1212/WNL.0000000000009946.

[17] K. Berganzo *et al.*, “Síntomas no motores y motores en la enfermedad de Parkinson y su relación con la calidad de vida y los distintos subgrupos clínicos,” *Neurología*, vol. 31, no. 9, pp. 585–591, 2016, doi: https://doi.org/10.1016/j.nrl.2014.10.010.

[18] P. Pastor and E. Tolosa, “La enfermedad de Parkinson: diagnóstico y avances en el conocimiento de la etiología y en el tratamiento,” *Medicina Integral*, vol. 37, no. 3, pp. 104–117, 2001, [Online]. Available: https://www.elsevier.es/es-revista-medicina-integral-63-articulo-la-enfermedad-parkinson-diagnostico-avances-10021650

[19] R. Martínez-Fernández., C. Gasca-Salas C., Á. Sánchez-Ferro, and J. Ángel Obeso, “ACTUALIZACIÓN EN LA ENFERMEDAD DE PARKINSON,” *Revista Médica Clínica Las Condes*, vol. 27, no. 3, pp. 363–379, 2016, doi: 10.1016/j.rmclc.2016.06.010.

[20] Aoki, T., Lin, J. F.-S., Kulić, D., & Venture, G. (2016). Segmentation of human upper body movement using multiple IMU sensors. *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 3163–3166. https://doi.org/10.1109/EMBC.2016.7591400

[21] X. Zou, Y. Hu, Z. Tian, y K. Shen, «Logistic regression model optimization and case analysis», en 2019 IEEE 7th international conference on computer science and network technology (ICCSNT)

[22] M. Christ, N. Braun, J. Neuffer, and A. W. Kempa-Liehr, “Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh – A Python package),” *Neurocomputing*, vol. 307, pp. 72–77, 2018, doi: https://doi.org/10.1016/j.neucom.2018.03.067.

[23] M. J. Bayarri, “Inferencia bayesiana sobre el coeficiente de correlacion de una poblacion normal bivariante,” *Trabajos de Estadistica y de Investigacion Operativa*, vol. 32, no. 3, pp. 18–31, Oct. 1981, doi: 10.1007/BF02890837.

[24] T. Kurita, “Principal Component Analysis (PCA),” in *Computer Vision: A Reference Guide*, K. Ikeuchi, Ed. Cham: Springer International Publishing, 2021, pp. 1013–1016. doi: 10.1007/978-3-030-63416-2_649.

[25] P. He, P. Li, and H. Sun, “Feature Extraction of Acoustic Signals Based on Complex Morlet Wavelet,” *Procedia Engineering*, vol. 15, pp. 464–468, 2011, doi: https://doi.org/10.1016/j.proeng.2011.08.088.

[26] Q. Guan *et al.*, “Deep convolutional neural network VGG-16 model for differential diagnosing of papillary thyroid carcinomas in cytological images: a pilot study,” *J Cancer*, vol. 10, pp. 4876–4882, 2019, doi: 10.7150/jca.28769.

[27] R. . Majalca Martínez y P. R. . Acosta Cano de los Ríos, «Una revisión de redes MLP como clasificadores de múltiples clases: A survey on MLP neural networks as multi-class classifiers», *TECNOCIENCIA Chih.*, vol. 9, n.º 3, pp. 148–159, feb. 2016.

[28] Aayesha, M. B. Qureshi, M. Afzaal, M. S. Qureshi, and M. Fayaz, “Machine learning-based EEG signals classification model for epileptic seizure detection,” *Multimedia Tools and Applications*, vol. 80, no. 12, pp. 17849–17877, May 2021, doi: 10.1007/s11042-021-10597-6

[29] E. Cruz, M. González, y J. Rangel, «Técnicas de machine learning aplicadas a la evaluación del rendimiento y a la predicción de la deserción de estudiantes universitarios, una revisión»., *Pri*, vol. 13, n.º 1, pp. 77-87, feb. 2022.

[30] A. J. Mazon, I. Zamora, V. Valverde, E. Torres, and I. Albizu, “Prototipo de localizador de faltas, basado en redes neuronales, para su operación en tiempo real”.

[31] J. M. Pineda, “Modelos predictivos en salud basados en aprendizaje de maquina (machine learning),” *Revista Médica Clínica Las Condes*, vol. 33, no. 6, pp. 583–590, 2022, doi: https://doi.org/10.1016/j.rmclc.2022.11.002.

[32] C. A. Rabbath and D. Corriveau, “A comparison of piecewise cubic Hermite interpolating polynomials, cubic splines and piecewise linear functions for the approximation of projectile aerodynamics,” *Defence Technology*, vol. 15, no. 5, pp. 741–757, 2019, doi: https://doi.org/10.1016/j.dt.2019.07.016.

[33] H. Rai and K. Chatterjee, “Detection of brain abnormality by a novel Lu-Net deep neural CNN model from MR images,” *Machine Learning with Applications*, vol. 2, p. 100004, Dec. 2020, doi: 10.1016/j.mlwa.2020.100004.

[34] Y. Yu, X. Si, C. Hu, and J. Zhang, “A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures,” *Neural Computation*, vol. 31, no. 7, pp. 1235–1270, Jul. 2019, doi: 10.1162/neco_a_01199.

[35] C. Yang, J. Zhai, and G. Tao, “Deep Learning for Price Movement Prediction Using Convolutional Neural Network and Long Short-Term Memory,” *Mathematical Problems in Engineering*, vol. 2020, pp. 1–13, Jul. 2020, doi: 10.1155/2020/2746845.

[36] Q. Wen et al., Transformers in Time Series: A Survey. 2023. [Online]. Available: https://arxiv.org/abs/2202.07125.

[37] H. Demuth, M. Beale y M. Hagan, "Neural Network Toolbox™ 7 User's Guide," en Procedia Engineering, vol. 15, pp. 3206–3210, 2011. [En línea]. Disponible en: https://www.sciencedirect.com/science/article/pii/S1877705811054993.

[38] ZE. Putra, H. Ajie, and I. Safitri, “Designing A User Interface and User Experience from Piring Makanku Application by Using Figma Application for Teens,” IJISTECH (International Journal of Information System and Technology), vol. 5, no. 3, pp. 308–315, 2021, doi: 10.30645/ijistech.v5i3.145.

[39] A. Villalta and J. P. Carvallo, “Modelos de calidad de software: Una revisión sistemática de la literatura,” *Maskana*, vol. 6, no. Supl., pp. 107–117, 2016, [Online]. Available: https://publicaciones.ucuenca.edu.ec/ojs/index.php/maskana/article/view/704.

[40] G. Louppe, "Understanding Random Forests: From Theory to Practice," arXiv:1407.7502 [stat.ML], Jun. 2015. [Online]. Available: https://arxiv.org/pdf/1407.7502.

[41] R. G. Mantovani, T. Horváth, A. L. D. Rossi, R. Cerri, S. Barbon Junior, J. Vanschoren, and A. C. P. L. F. de Carvalho, "Better Trees: An empirical study on hyperparameter tuning of classification decision tree induction algorithms," arXiv:1812.02207 [cs.LG], Dec. 2023. [Online]. Available: https://arxiv.org/pdf/1812.02207.

[42] A. Iparragirre, I. Barrio, J. Aramendi, and I. Arostegui, "Estimation of logistic regression parameters for complex survey data: a real data-based simulation study," *arXiv*, vol. 2303.01754, Mar. 2023. [Online]. Available: https://arxiv.org/abs/2303.01754.

[43] M. Wu, T. Hoque, and A. B. Farahani, "A Comparative Analysis of Gradient Boosted Decision Trees," *arXiv preprint arXiv:1911.01914*, Nov. 2019. [Online]. Available: https://arxiv.org/abs/1911.01914.

[44] P. Cunningham and S. J. Delany, "Make k-Nearest Neighbour Classifiers: 2nd Edition (with Python examples)," *arXiv preprint arXiv:2004.04523*, Apr. 2020. [Online]. Available: https://arxiv.org/abs/2004.04523.

[45] Z. Ji, J. Li, and M. Telgarsky, “Early-stopped neural networks are consistent,” in *Advances in Neural Information Processing Systems*, 2021, vol. 34, pp. 1805–1817. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2021/file/0e1ebad68af7f0ae4830b7ac92bc3c6f-Paper.pdf.

ANEXOS

Anexo 1

Enlace al prototipo desplegado:

https://www.figma.com/proto/UIn5mZevDxuxHIlfN7YFZj/PrototipoTesis

Anexo 2

Enlace a la guía del fabricante de los sensores para corregir valores de la velocidad angular:

https://datasheet.sisoog.com/file/zmedia/dex/438dbac203e570fca1ca63ceda0f48fa_en.DM00286303.pdf