Maestría en

CIENCIA DE DATOS Y MAQUINAS DE APRENDIZAJE CON MENCION EN INTELIGENCIA ARTIFICAL

Trabajo previo a la obtención de título de Magister en Ciencia de Datos y Máquinas de Aprendizaje con mención en Inteligencia Artificial

AUTORES:

GUACHICHULCA MORALES JULIO ARMANDO

SALAZAR ZABALA ALEXIS FABRICIO

SANCHEZ SANCHEZ JOSE ANTONIO

VEINTIMILLA ALMEIDA RICHARD GERMAN

TUTORES:

Iván Reyes Chacón

Alejandro Cortés López

TEMA:

Identificación de acciones humanas y detección de caídas en adultos mayores basada en datos de sensores de dispositivos móviles

Quito - Ecuador

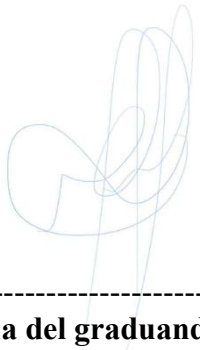
Junio – 2025

REINVENTEMOS EL FUTURO

Certificación de autoría

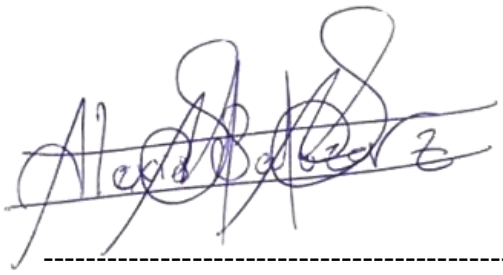
Nosotros, José Antonio Sánchez Sánchez, Julio Armando Guachichulca Morales, Alexis Fabricio Salazar Zabala, Richard Germán Veintimilla Almeida, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido presentado anteriormente para ningún grado o calificación profesional y que se ha consultado la bibliografía detallada.

Cedemos nuestros derechos de propiedad intelectual a la Universidad Internacional del Ecuador (UIDE), para que sea publicado y divulgado en internet, según lo establecido en la Ley de Propiedad Intelectual, su reglamento y demás disposiciones legales.



Firma del graduando
José Antonio Sánchez Sánchez

Firma del graduando
Julio Armando Guachichulca Morales



Firma del graduando
Alexis Fabricio Salazar Zabala

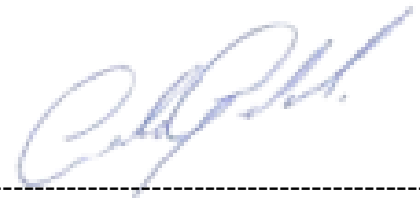
Firma del graduando
Richard Germán Veintimilla Almeida

Autorización de Derechos de Propiedad Intelectual

Nosotros, José Antonio Sánchez Sánchez, Julio Armando Guachichulca Morales, Alexis Fabricio Salazar Zabala, Richard Germán Veintimilla Almeida, en calidad de autores del trabajo de investigación titulado ***Identificación de acciones humanas basada en datos de sensores de dispositivos móviles.***, autorizamos a la Universidad Internacional del Ecuador (UIDE) para hacer uso de todos los contenidos que nos pertenecen o de parte de los que contiene esta obra, con fines estrictamente académicos o de investigación. Los derechos que como autores nos corresponden, lo establecido en los artículos 5, 6, 8, 19 y demás pertinentes de la Ley de Propiedad Intelectual y su Reglamento en Ecuador.

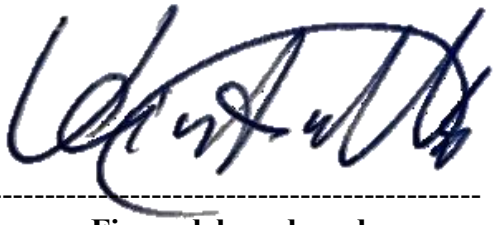
D. M. Quito, (Agosto 2025)

Firma del graduando
José Antonio Sánchez Sánchez



Firma del graduando
Julio Armando Guachichulca Morales

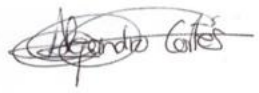
Firma del graduando
Alexis Fabricio Salazar Zabala



Firma del graduando
Richard Germán Veintimilla Almeida

Aprobación de dirección y coordinación del programa

Nosotros, Alejandro Cortés **e** Iván Reyes**,** declaramos que: José Antonio Sánchez Sánchez, Julio Armando Guachichulca Morales, Alexis Fabricio Salazar Zabala, Richard Germán Veintimilla Almeida, son los autores exclusivos de la presente investigación y que ésta es original, auténtica y personal de ellos.



MCS.Alejandro Cortés
Director de la
Maestría en Ciencia de Datos y
Máquinas de Aprendizaje
con mención en Inteligencia Artificial

MCS. Iván Reyes
Coordinador de la
Maestría en Ciencia de Datos
y Máquinas de Aprendizaje
con mención en Inteligencia Artificial

DEDICATORIA

Dedicamos este trabajo con gratitud a quienes hicieron posible nuestro sueño. A cada uno de nuestros familiares, por sostenernos con su amor, comprensión y palabras de aliento en cada desafío. Somos cuatro voces que se unen y entregan este logro como tributo a quienes creyeron en nosotros.

AGRADECIMIENTOS

Los cuatro autores de este proyecto, honramos a la comunidad científica cuyo espíritu de descubrimiento ha guiado en cada etapa. A la institución que con dedicación y calor humano nos ha acogido y formado, brindándonos herramientas únicas para crecer. A nuestras familias siendo pilares de fortaleza, gracias por su apoyo incondicional, su paciencia infinita y su amor que iluminó cada jornada de estudio. Y a nuestro país, por abrir caminos y ofrecernos oportunidades.

RESUMEN

El presente proyecto se desarrolla en el contexto de la ciencia de datos y máquinas de aprendizaje, se trata de obtener el mejor modelo posible para aplicar a datos que generan y proveen los dispositivos móviles como el acelerómetro y giroscopio, para obtener de estos una estimación del estado corporal de los individuos, estados como son: Parado, acostado, subiendo gradas, bajando gradas, sentado o caminando.

Se han evaluado entrenamientos de los modelos: Random forest, CNN (convolutional neural network) en una dimensión y SVM (support vector machine), en este último caso también se evaluó la aplicación de la herramienta GRIDSEARCH para evaluación de múltiples hiper parámetros y conseguir los mejores resultados.

Se presenta entonces el desarrollo de este trabajo de evaluación, tratamiento, entrenamiento y modelado de los datos de la base UCI-HAR, siendo esta una de las más completas disponibles para el trabajo.

Se presentan los resultados mostrando como el modelo SVM nos ofreció los mejores resultados en la clasificación de los datos. superior al CNN y random forest.

Palabras Claves: Reconocimiento de Actividad Humana, Sensores de dispositivos móviles, Aprendizaje Automático, Aprendizaje Profundo, Prevención de Caídas en Adultos Mayores, Detección de Caídas

.

ABSTRACT

This project, developed in the context of data science and machine learning, aims to obtain the best possible model to apply to data generated and provided by mobile devices such as accelerometers and gyroscopes, in order to obtain an estimate of the body state of individuals, such as: standing, lying down, ascending stairs, descending stairs, sitting, or walking.

Training models were evaluated: Random Forest, one-dimensional CNN (convolutional neural network), and SVM (support vector machine). In the latter case, the application of the GRIDSEARCH tool was also evaluated to evaluate multiple hyperparameters and achieve the best results.

The development of this work is then presented: evaluation, processing, training, and modeling of data from the UCI-HAR database, which is one of the most comprehensive available for this work.

The results are presented, showing how the SVM model offered the best results in data classification, superior to CNN and random forest.

Keywords: Human Activity Recognition, Mobile Device Sensors, Machine Learning, Deep Learning, Fall Prevention in Older Adults, Fall Detection

TABLA DE CONTENIDOS

Contenido

LISTA DE TABLAS

LISTA DE FIGURAS (Índice de figuras)

Capítulo 1

1. Introducción

1.1.Definición del proyecto

El proyecto titulado "Identificación de acciones humanas y detección de caídas en adultos mayores basada en datos de sensores de dispositivos móviles" tiene como propósito el desarrollo de un modelo predictivo que, a partir de los datos de los sensores integrados en smartphones (acelerómetro, giroscopio, barómetro y magnetómetro), sea capaz de distinguir automáticamente acciones como caminar, correr, caerse y levantarse. Este sistema buscará proveer una solución de bajo costo y fácil despliegue, orientada a mejorar la seguridad y calidad de vida de la población adulta mayor.

1.2.Justificación e importancia del trabajo de investigación

Las caídas constituyen la segunda causa de lesiones no intencionales mortales a nivel mundial. Cada año se registran unos 37,3 millones de caídas graves que requieren atención médica y cerca de 684 000 muertes, de las cuales más del 80 % ocurren en países de ingresos bajos y medianos (WHO, 2023). En términos de prevalencia, un metaanálisis de 104 estudios con más de 36 millones de participantes estimó que el 26,5 % (IC 95 % 23,4–29,8 %) de las personas de 60 años o más sufren al menos una caída cada año, con una prevalencia en la región americana de 27,9 % (IC 95 % 22,4–34,2 %) (Ahmadi et al., 2022).

En América Latina y el Caribe, las tasas de caídas oscilan entre el 22 % (Barbados) y el 34 % (Chile) según análisis de la **World Guidelines for Falls Prevention and Management**, lo que evidencia una variabilidad asociada a factores socioeconómicos y culturales (Montero-Odasso et al., 2022). Un estudio multinacional en ocho ciudades latinoamericanas obtuvo una prevalencia de caídas que requirieron atención médica de entre 6,0 % y 11,3 %, encontrando además que las mujeres presentaban mayor riesgo en asociación

con comorbilidades como depresión e incontinencia urinaria (Reyes-Ortiz et al., 2020).

El envejecimiento poblacional y la alta incidencia de caídas plantean una carga significativa en salud pública. Para afrontarlo, la OMS y especialistas recomiendan:

Modificaciones ambientales. en el hogar (iluminación, retirar obstáculos) y en espacios públicos para reducir riesgos (WHO, 2023).

Monitoreo con tecnología. Muchos adultos mayores carecen de recursos para acceder a dispositivos especializados o personal de salud que realice un seguimiento continuo, lo que retrasa la detección de episodios críticos.

El uso de sensores en smartphones y wearables ofrece detección en tiempo real de patrones de movimiento y alertas tempranas tras una caída (WHO, 2023).

1.3.Alcance

El proyecto consiste en desarrollar y validar un modelo de reconocimiento automático de seis actividades humanas cotidianas (caminar, subir escaleras, bajar escaleras, sentarse, estar de pie y acostarse) a partir de señales procedentes de acelerómetro, giroscopio, barómetro y magnetómetro de un smartphone; para ello se empleará principalmente el UCI HAR Dataset. La evaluación se hará con datos sintéticos para medir precisión, recall, F1 y robustez frente a variaciones de posición y velocidad, quedan excluidos el despliegue en aplicaciones móviles de producción, el despliegue masivo en campo.

0.4.Objetivos

0.4.1. Objetivo general

Desarrollar y validar un modelo automático capaz de Identificar acciones humanas y prevención de caídas en usuarios mayores de 60 años usando los datos de sensores de un teléfono inteligente, con un nivel de sensibilidad y especificidad superior al 80%.

0.4.2. Objetivo específico

Recopilar y etiquetar datos de sensores (acelerómetro, giroscopio, barómetro y magnetómetro) con datos sintéticos.

Preprocesar los datos y extraer características relevantes que diferencien cada acción.

Entrenar y optimizar modelos para clasificar las acciones estudiadas.

Evaluar el desempeño de los modelos mediante métricas de precisión, recall y F1-score.

Capítulo 2

2. Revisión De Literatura

2.1.*Estado Del Arte*

Artículo: A systematic literatura review on human activity recognition using Smart devices (2025): El estudio de Qureshi et al. (2025) presenta una revisión sistemática de la literatura (SLR) sobre el reconocimiento de actividad humana (HAR) mediante dispositivos inteligentes, abarcando artículos publicados entre enero de 2021 y septiembre de 2024 en bases de datos como Springer, IEEE Xplore, ScienceDirect, Taylor & Francis y ACM. El objetivo principal es comparar el rendimiento de arquitecturas de aprendizaje profundo (CNN, LSTM, Transformers y modelos ensamblados) y evaluar el uso de conjuntos de datos abiertos como WISDM, PAMAP2, UCI-HAR, Opportunity y MHealth, con el fin de identificar retos comunes y proponer direcciones futuras de investigación (Qureshi et al., 2025, p. 1)

En la introducción se destaca la importancia de HAR en aplicaciones como atención sanitaria, cuidado de mayores, seguridad y entornos inteligentes, mostrando cómo los sistemas basados en vídeo (p. ej., vigilancia) presentan limitaciones de privacidad y ubicación, mientras que los sensores IoT en espacios cerrados pueden sufrir problemas de conectividad. Por contraste, los dispositivos inteligentes (smartphones y smartwatches) ofrecen cobertura continua, reducen las preocupaciones de privacidad y permiten el reconocimiento de actividades en entornos sin infraestructura fija. Asimismo, se repasa la evolución de los métodos desde enfoques tradicionales de machine learning (SVM, KNN, HMM) hacia técnicas de deep learning que operan directamente sobre datos brutos sin extracción manual de características (Qureshi et al., 2025, p. 2)

La metodología empleada sigue pautas del Cochrane Handbook, PRISMA 2020 y guías específicas de campo, y comprende la definición de preguntas de investigación,

refinamiento de cadenas booleanas, criterios de inclusión y exclusión (artículos de 2021–2024, enfoque en deep learning, dispositivos inteligentes) y el filtrado de cinco bases de datos. Tras procesar 1402 resultados iniciales y aplicar criterios de calidad, se seleccionaron 151 artículos para el análisis final, garantizando la validez y actualidad de los hallazgos (Qureshi et al., 2025, p. 8)

En el apartado de análisis de modelos, se observa que las CNN dominan la investigación en HAR por su eficiencia en la extracción de dependencias espaciales y temporales (1-D CNN), seguidas por modelos de LSTM (capacidad para dependencias a largo plazo) y arquitecturas ensambladas que mejoran la robustez. Los RNN y MLP aparecen con menor frecuencia, mientras que los Transformers y GAN emergen como enfoques prometedores para el aprendizaje auto-supervisado y la generación de datos sintéticos para paliar desequilibrios. Además, se destacan tendencias como edge computing para procesamiento en el dispositivo, federated learning para privacidad, fusión multimodal de sensores y Explainable AI para transparencia de decisiones (Qureshi et al., 2025, p. 33)

Respecto a los conjuntos de datos, se focaliza en los cinco más utilizados: WISDM (desequilibrio de clases, pero fácilmente accesible), UCI-HAR (balanceado y estándar), PAMAP2 (mayor variedad de ADL), Opportunity (entorno sensor-rico, pero complejidad y desequilibrios) y MHealth (combinación de IMU y ECG, útil para monitorización sanitaria, pero con baja diversidad de voluntarios). Cada uno presenta fortalezas y limitaciones que afectan la generalización y aplicabilidad de los modelos en escenarios reales (Qureshi et al., 2025, p. 40)

Entre los desafíos y direcciones futuras se identifican la necesidad de: 1) explorar conjuntos de datos cerrados y personalizados para aplicaciones específicas; 2) investigar enfoques federados y distribuidos para mejorar privacidad y estabilidad de nodos; 3) desarrollar técnicas avanzadas de preprocesamiento y data augmentation (más allá de

normalización y ventana deslizante); 4) incorporar edge computing para reducir latencia y proteger datos sensibles; y 5) fortalecer las prácticas de privacidad mediante anonimización, cifrado y aprendizaje federado diferenciado (Qureshi et al., 2025, p. 47)

Artículo: Human activity recognition: A review of deep learning-based methods (2025): El reconocimiento de actividad humana (HAR, por sus siglas en inglés) se ha consolidado como una de las áreas más relevantes de la visión por computador, dada su aplicabilidad en sectores como salud, seguridad, entretenimiento y monitoreo. El objetivo principal de HAR es identificar automáticamente acciones humanas a partir de secuencias de datos, superando desafíos como la escala, el desorden, la oclusión y la variabilidad intrapersonal. Las actividades reconocidas se clasifican en tres categorías: acciones de una sola persona, actividades grupales e interacciones humano-objeto, lo cual permite una visión estructurada de los diferentes niveles de complejidad del comportamiento humano (Dutta et al., 2025, p. 1).

La revisión de Dutta et al. presenta un proceso de selección de estudios riguroso basado en búsquedas en múltiples plataformas (Computer Vision Foundation, ScienceDirect, Google Scholar, SpringerLink, ACM Digital Library, IEEE Xplore y arXiv) que abarca trabajos publicados hasta 2024. Inicialmente se recopilaron 500 artículos, de los cuales se eliminaron duplicados y aquellos centrados en métodos basados en sensores, priorizando los enfoques basados en visión. Tras descartar trabajos redundantes o con análisis limitados, se incorporaron 253 artículos que demostraron novedad, incluidos 67 aportados en conferencias de alto impacto y 32 métodos recientes de 2024 (Dutta et al., 2025, p. 3).

El corazón de la revisión se centra en los métodos de aprendizaje profundo aplicados al HAR, clasificando las técnicas según arquitecturas como redes neuronales convolucionales (CNN), recurrentes (RNN), autoencoders, redes generativas antagónicas (GAN) y transformadores. Se ofrece una taxonomía que incluye variantes como CNN 2D/3D, grafos

convolucionales, mecanismos de atención, modelos de flujo óptico y aprendizaje semi-supervisado. Además, se comparan ventajas, desventajas y precisiones de estas técnicas en distintos conjuntos de datos públicos, con tablas estructuradas que facilitan la evaluación comparativa de desempeño (Dutta et al., 2025, p. 2).

Los estudios previos analizados abarcan tres áreas principales: actividades de una sola persona, que incluyen desde gestos y acciones atómicas hasta interacciones con objetos; actividades grupales, donde se emplean modelos basados en grafos, atención y arquitecturas multistream para capturar dinámicas espaciales y temporales de múltiples individuos; e interacciones humano-objeto (HOI), que requieren identificar elementos involucrados y la relación espacial y temporal entre ellos. Se describen métodos destacados y casos de uso en conjuntos de datos como UCF-101, NTU RGB+D y HICO-DET, con mejoras notables en precisión (Dutta et al., 2025, p. 3).

La revisión también identifica retos críticos que afectan la generalización de los modelos, tales como desequilibrios en los conjuntos de datos, condiciones de iluminación variables, oclusión, variabilidad intra- e interclase y limitaciones en la anotación temporal de eventos. Asimismo, se señalan futuras líneas de investigación: aprendizaje de pocos disparos y cero disparos para reconocer acciones no vistas, integración multimodal de datos (RGB, profundidad, esqueleto y sensores inerciales), modelos explicables y razonamiento temporal para comprender dependencias a largo plazo, así como despliegue en entornos con recursos limitados. Estas perspectivas buscan impulsar sistemas HAR más robustos y versátiles (Dutta et al., 2025, p. 19).

Artículo: Your day in your Pocket (2023): El reconocimiento de actividades humanas complejas a partir de datos de acelerómetro en smartphones ofrece una vía no invasiva para entender el comportamiento diario de los usuarios y apoyar aplicaciones de bienestar y salud mental. Este enfoque permite clasificar acciones como dormir, comer o

estudiar mediante modelos de aprendizaje profundo aplicados exclusivamente a señales de aceleración, superando la necesidad de infraestructuras fijas o múltiples sensores costosos (Bouton-Bessac et al., 2023, p. 1)

La introducción destaca que, aunque los wearables como smartwatches proporcionan datos de alta calidad, su adopción es limitada y su uso tiende a declinar tras seis meses. En contraste, más del 80 % de la población posee smartphones, lo que facilita la recolección continua y diversa de datos de múltiples sensores (acelerómetro, giroscopio, nivel de luz, uso de apps). Esta ubicuidad y multimodalidad sustentan el potencial de los smartphones para el reconocimiento de actividades complejas en distintos contextos culturales y geográficos (Bouton-Bessac et al., 2023, p. 2).

El estudio utiliza un conjunto de datos de 216 000 autorreportes de 637 estudiantes universitarios de cinco países, recopilados durante cuatro semanas en el contexto de la pandemia como parte del proyecto europeo WeNet. Los participantes, con un promedio de edad de 22 años y 61 % mujeres, completaron un diario de actividades cada hora mediante una app, que simultáneamente registraba datos de 34 sensores de smartphone, enfocándose finalmente en 40 000 autorreportes válidos de acelerómetro (Bouton-Bessac et al., 2023, p. 5)

Para asegurar modelos entrenables, se filtraron y fusionaron categorías de actividades. De las 34 etiquetas iniciales, se descartaron las con muy pocos ejemplos (p. ej., viajar) o demasiado genéricas (descanso), y se agruparon variables afines (shopping, social media), quedando ocho clases finales: dormir, comer, estudiar, asistir a clase, comunicación online, ver videos/TV, deporte y compras (Bouton-Bessac et al., 2023, p. 6)

La preparación de datos asumió que la actividad ocurría durante los tres minutos previos a cada autorreporte, eliminando la ventana de respuesta y remuestreando a 3,33 Hz. Tras descartar segmentos con menos de 600 muestras, se evidenció una pérdida significativa de datos en ciertos países (Mongolia e Italia) debido a vacíos de registro, lo que redujo el

conjunto de entrenamiento, pero reflejó condiciones reales de uso (Bouton-Bessac et al., 2023, p. 7)

Para la clasificación binaria de cada actividad ("sí vs. no"), se compararon modelos basados en capas 1D CNN y LSTM. Los datos de entrada tenían forma 3 × 600 (ejes x, y, z); se usaron el optimizador Adam, la pérdida de entropía cruzada y métricas de precisión, AUROC y F1 con promedio macro. La validación cruzada de 10 pliegues y particiones tanto balanceadas como desbalanceadas permitieron evaluar robustez y sensibilidad a clases minoritarias (Bouton-Bessac et al., 2023, p. 8).

En modelos sin personalización (nivel población), la AUROC osciló entre 0,51 y 0,62, siendo dormir la actividad mejor reconocida; actividades de nicho como compras presentaron desempeño bajo. Con modelos híbridos parcialmente personalizados, la AUROC mejoró hasta 0,76 en dormir y alcanzó entre 0,56 y 0,68 en otras clases, demostrando el valor de incorporar datos específicos de usuario (Bouton-Bessac et al., 2023, p. 9).

A pesar de las limitaciones impuestas por la recolección multicountry y condiciones reales de pandemia —que homogenizan patrones de movimiento—, los resultados confirman la factibilidad de inferir actividades complejas con acelerómetro único. Se propone investigar datos cerrados, enfoques federados, fusión multimodal, edge computing y XAI para mejorar privacidad, latencia y explicabilidad en aplicaciones de bienestar (Bouton-Bessac et al., 2023, p. 11).

Artículo: Human activity recognition with smartphone-integrated sensors" (2024): El reconocimiento de actividad humana (HAR) es un área de investigación esencial centrada en la capacidad de los teléfonos inteligentes para identificar las acciones que realiza una persona mediante el procesamiento de datos procedentes de sensores integrados. Estas actividades abarcan desde tareas cotidianas como caminar o permanecer sentado hasta acciones más complejas como subir escaleras o conducir un vehículo. La relevancia de HAR

radica en sus aplicaciones en salud, monitorización de pacientes, asistencia en el hogar y mejora de interfaces de usuario, permitiendo detectar patrones de comportamiento y anomalías en entornos reales (Dentamaro et al., 2024, p. 2).

Los smartphones modernos incorporan diversos sensores que facilitan la captura de información del entorno y del usuario. Entre los más relevantes se encuentran el acelerómetro, que mide la aceleración en los tres ejes; el giroscopio, que registra la velocidad angular; el magnetómetro, que detecta campos magnéticos y sirve de brújula; el sensor de proximidad, que identifica objetos cercanos; el sensor de luminosidad, que ajusta el brillo de la pantalla; y el receptor GPS, que proporciona información de ubicación y velocidad. Estos sensores pueden emplearse de forma independiente o combinada para mejorar la precisión en la detección de actividades (Dentamaro et al., 2024, p. 3).

La arquitectura de un sistema HAR se articula en tres componentes principales: adquisición de datos, ingeniería de características y clasificación. En la fase de adquisición, se recopilan y almacenan señales crudas de los sensores. Durante la ingeniería de características, se aplican técnicas de preprocesamiento como filtrado y segmentación en ventanas, extracción de atributos en el dominio del tiempo (media, varianza, curtosis) y en el dominio de la frecuencia (densidad espectral de potencia, entropía espectral), así como selección de las variables más discriminantes. Finalmente, en la clasificación se utilizan modelos de aprendizaje automático para asignar a cada segmento el tipo de actividad correspondiente (Dentamaro et al., 2024, p. 4).

La extracción y selección de características son etapas críticas para el rendimiento de los sistemas HAR. En la extracción, se calculan medidas estadísticas de la señal y transformaciones de Fourier para obtener información tanto temporal como frecuencial. Para reducir la dimensionalidad y eliminar redundancias, se emplean métodos como el Análisis de Componentes Principales (PCA) y el Análisis Discriminante Lineal (LDA), que proyectan los

datos en espacios de menor dimensión conservando la mayor cantidad de información relevante. Estas técnicas mejoran la eficiencia computacional y la calidad de los datos de entrada a los clasificadores (Dentamaro et al., 2024, p. 11).

En la fase de clasificación, se han probado numerosas técnicas de aprendizaje supervisado y no supervisado. Entre las primeras destacan K-Nearest Neighbors, Support Vector Machines, Random Forest y redes neuronales profundas, incluyendo Convolutional Neural Networks (CNN) y Long Short-Term Memory (LSTM). También se utilizan modelos generativos como Gaussian Mixture Models y Hidden Markov Models para capturar la naturaleza secuencial de las actividades. Los enfoques más avanzados combinan CNN y LSTM en arquitecturas híbridas para aprovechar las capacidades de extracción de características espaciales y temporales (Dentamaro et al., 2024, p. 12).

Los conjuntos de datos públicos han sido fundamentales para entrenar y evaluar los sistemas HAR. Entre los más utilizados figuran UniMiB SHAR, USC-HAD, UCI HAR y WISDM, que difieren en número de participantes, tipo de dispositivos, actividades registradas y disponibilidad de datos crudos o preprocesados. Los estudios reportan precisiones superiores al 90 % en entornos controlados, y alrededor del 85–95 % en escenarios del mundo real, dependiendo de la combinación de sensores, la complejidad de la actividad y el modelo empleado (Dentamaro et al., 2024, p. 12).

En la discusión se identifican tendencias y desafíos que marcarán el futuro de HAR. Se destaca la necesidad de mejorar la generalización de los modelos a entornos no controlados, optimizar el consumo energético en smartphones, y desarrollar técnicas de aprendizaje continuo que permitan adaptarse a nuevas actividades sin reentrenar desde cero. Asimismo, se subraya el potencial de incorporar sensores adicionales y métodos de fusión multimodal para enriquecer la información. La integración de HAR en aplicaciones de salud,

asistencia y seguridad promete extender el impacto de estas tecnologías (Dentamaro et al., 2024, p. 17).

Artículo: An Effective Deep Learning Framework for Fall Detection (2024): El envejecimiento de la población y la alta incidencia de caídas hacen que el diseño de sistemas de detección de caídas (FDS) sea una prioridad para la salud pública. Según la Organización Mundial de la Salud, las caídas son la segunda causa de muertes accidentales a nivel global, afectando especialmente a adultos mayores, donde al menos un cuarto sufre una caída anual. Los FDS basados en sensores portátiles ofrecen una solución no invasiva y asequible para emitir alertas inmediatas tras un incidente, superando limitaciones de coste y privacidad de sistemas de predicción o visión fija (Zhang et al., 2024, p. 2).

Los FDS portátiles se clasifican en tres enfoques: basados en umbrales, que comparan características extraídas manualmente con valores predefinidos; modelos de aprendizaje automático tradicionales (KNN, SVM, árboles de decisión) que dependen de ingeniería de características; y técnicas de deep learning (CNN, LSTM) que realizan extracción automática de atributos. Sin embargo, la mayoría de los modelos anteriores solo emplean datos de acelerómetro, lo que limita la diferenciación entre caídas reales y actividades diarias similares debido a la falta de información rotacional (Zhang et al., 2024, p. 2).

Con el fin de abordar estas deficiencias, se propone el modelo DSCS (dual-stream convolutional neural network self-attention), una arquitectura de aprendizaje profundo que procesa simultáneamente datos de acelerómetro y giroscopio. Cada flujo pasa por una red CNN de tres capas para extraer patrones espaciales y temporales, tras lo cual un módulo de self-attention pondera dinámicamente las características más relevantes para la detección de caídas. El modelo se entrenó y validó con dos conjuntos de datos públicos (SisFall y MobiFall) y se corroboró su capacidad de generalización mediante pruebas prácticas con 1 700 ensayos (Zhang et al., 2024, p. 3).

La arquitectura del DSCS se organiza en tres módulos principales. El primero, de extracción de características, aplica convoluciones 1D y pooling para transformar secuencias de 3 ejes en vectores de 64 dimensiones. El segundo, de self-attention, genera matrices Query, Key y Value para asignar pesos según la importancia de cada posición en el vector. Finalmente, el módulo de clasificación concatena las representaciones ponderadas de acelerómetro y giroscopio, normaliza, aplica dropout y las introduce en capas fully connected con softmax para la decisión final (Zhang et al., 2024, p. 4).

En evaluación con los conjuntos de prueba de SisFall y MobiFall, el DSCS alcanzó precisiones del 99,32 % y 99,65 %, respectivamente, con recalls superiores al 99 % y precisiones por encima del 98 %. Las matrices de confusión mostraron un mínimo número de falsos positivos y negativos. Además, la validación cruzada a 5 y 10 pliegues mantuvo F1-scores medios del 99,09 % (SisFall) y 98,69 % (MobiFall), demostrando estabilidad y robustez frente a variaciones en los datos (Zhang et al., 2024, p. 5).

Para comprobar su aplicabilidad en entornos reales, se implementó el DSCS en un smartwatch Huawei Watch 3 con sensores a 50 Hz. Diez participantes realizaron ocho tipos de caída y nueve actividades de la vida diaria similares. De 1 700 ensayos, el modelo obtuvo una precisión del 96,41 %, recall del 95,12 % y especificidad del 97,55 %, evidenciando una leve degradación por cambios en sensor y protocolo, pero manteniendo un rendimiento alto en uso práctico (Zhang et al., 2024, p. 10).

En conclusión, el estudio presenta un marco profundo innovador que combina procesamiento dual-stream y atención automática para mejorar la detección de caídas. El DSCS supera a algoritmos de machine learning y otras arquitecturas DL, confirmando la eficacia del self-attention en tareas de clasificación de series temporales. Estas aportaciones abren camino a sistemas embebidos de alta precisión, con potencial para integrarse en dispositivos de salud preventiva y asistencia continua (Zhang et al., 2024, p. 11).

2.2.Marco Teórico

El envejecimiento poblacional constituye uno de los desafíos demográficos y de salud pública más significativos del siglo XXI. Conforme aumenta la proporción de personas con 60 o más años, los sistemas sanitarios deben adaptarse a las condiciones crónicas y los riesgos asociados con la vejez, entre los que destacan las caídas. Este marco teórico analiza el proceso de envejecimiento global, la carga de las caídas en adultos mayores, sus implicaciones en la salud y economía, así como la epidemiología a nivel mundial, latinoamericano y específicamente en Ecuador. Finalmente, se revisan tecnologías basadas en sensores móviles para la detección y prevención temprana de caídas.

Envejecimiento poblacional y riesgos asociados. Según proyecciones de Naciones Unidas, en 2015 el 12 % de la población mundial tenía 60 años o más; para 2050 se espera que este grupo alcance el 22 % (UN DESA, 2020). Este incremento, más acelerado en países de ingresos bajos y medios, implica un aumento simultáneo de comorbilidades crónicas y vulnerabilidades físicas. En Ecuador, el Fondo de Población de las Naciones Unidas reporta que en 2020 alrededor del 11 % de la población supera los 60 años, cifra en crecimiento debido a mejoras en la esperanza de vida y cambios en la dinámica de natalidad (UNFPA, 2022). Este envejecimiento incrementa la incidencia de eventos adversos en el hogar, entre los que destacan las caídas, reconocidas como un importante factor de discapacidad y mortalidad en la tercera edad.

Importancia de las caídas en la salud pública. Las caídas ocupan el segundo lugar entre las causas de lesiones no intencionales mortales a nivel global, con 684 000 muertes anuales atribuidas a este mecanismo, de las cuales más del 80 % ocurren en países de ingresos bajos y medios (WHO, 2023). Los costos directos e indirectos hospitalización, rehabilitación, pérdida de autonomía y cuidados a largo plazo representan una carga económica y social

sustancial. Asimismo, las caídas recurrentes engendran temor a caminar entre los mayores, limitando su actividad física y promoviendo el sedentarismo, lo cual agrava la sarcopenia y el deterioro funcional (Sherrington et al., 2020). La prevención de caídas, por tanto, es una prioridad en políticas de envejecimiento activo y programas de salud comunitaria.

Epidemiología de caídas a nivel mundial. Un metaanálisis reciente que incluyó más de 36 millones de participantes estimó que el 26,5 % de adultos mayores de 60 años sufre al menos una caída cada año (IC 95 % 23,4–29,8 %) y que la región de las Américas presenta una prevalencia del 27,9 % (IC 95 % 22,4–34,2 %) (Ahmadi, Shamsi, Shojaei, & Tavakoli, 2022). Estas cifras muestran una magnitud similar en Europa y Asia, mientras que África reporta tasas algo menores, posiblemente por subregistro. Factores como el polimedicación, alteraciones de la marcha, afecciones visuales y presencia de depresiones elevan el riesgo individual, configurando un perfil de alto riesgo que requiere intervenciones multifactoriales.

Caídas en América Latina y el Caribe. En América Latina, la prevalencia de caídas que demandan atención médica varía según el país: estudios multinacionales en ocho ciudades reportaron rangos entre 6,0 % en Montevideo y 11,3 % en Santiago de Chile (Reyes-Ortiz, López, & Suárez, 2020). La heterogeneidad se asocia con diferencias en acceso a servicios de salud, cultura de autocuidado y entornos urbanos. Además, las mujeres mayores experimentan tasas más altas de caídas debido a mayor prevalencia de osteoporosis y desequilibrios hormonales postmenopáusicos, así como a condiciones psicosociales como la depresión, que influye en la regulación postural y el tiempo de reacción (Reyes-Ortiz et al., 2020). En ese contexto, los programas de prevención combinan ejercicio, optimización del hogar y revisión de fármacos para mitigar riesgos.

Características de las caídas en Ecuador. La literatura ecuatoriana, aunque escasa, coincide en señalar tasas elevadas de riesgo en contextos rurales y urbanos. En Paccha

(Cuenca), el 61,6 % de 86 adultos mayores evaluados obtuvo puntajes de alto riesgo en la escala de Tinetti, correlacionado con edad avanzada y déficit de equilibrio (Campoverde, González, & Pérez, 2022). Asimismo, datos de la encuesta SABE I evidencian que más del 30 % de personas mayores en regiones de la Sierra y Costa sufrieron al menos una caída en el año previo, con zonas rurales mostrando peores condiciones de infraestructura y menor acceso a programas de rehabilitación (Orces, 2022). Estas caídas provocan factores de riesgo compuestos: fracturas, dependencia funcional y rentas de cuidados prolongados, desbordando la atención primaria.

Tecnología móvil para detección y alerta temprana. La proliferación de smartphones y wearables ha abierto nuevas oportunidades de monitoreo continuo. Los sensores inerciales integrados como: acelerómetro, giroscopio, magnetómetro, que permiten capturar patrones de movimiento y detectar anomalías relacionadas con episodios de desequilibrio o caídas reales (WHO, 2023). Plataformas de análisis en tiempo real emplean algoritmos de machine learning para distinguir caídas de actividades cotidianas, enviando alertas instantáneas a cuidadores o servicios de emergencia (WHO, 2023). En entornos con recursos limitados, esta infraestructura de bajo costo y amplia penetración constituye una estrategia prometedora para reducir tiempos de respuesta y complicaciones asociadas.

Conceptualización y definición de HAR: El Reconocimiento de Actividad Humana (HAR, por sus siglas en inglés) se define como la tarea de identificar automáticamente patrones de conducta o acciones humanas a partir de secuencias de datos recogidos por distintos tipos de sensores (p. ej., acelerómetros, giroscopios, cámaras) o combinaciones de ellos. Dentamaro et al. (2024) destacan que, en el contexto de los smartphones, estas acciones varían desde actividades básicas como caminar o estar sentado, hasta tareas complejas como

subir escaleras o conducir un vehículo, con aplicaciones críticas en salud, asistencia domiciliaria y mejora de interfaces de usuario.

Por su parte, Dutta, Boongoen y Zwiggelaar (2025) amplían esta definición incluyendo tres niveles de complejidad en las actividades:

Acciones individuales atómicas, es decir, gestos o movimientos elementales.

Actividades grupales, que implican interacción entre dos o más personas.

Interacciones humano-objeto (HOI), donde el sujeto manipula o utiliza objetos externos

Desde una perspectiva formal, HAR se plantea como un problema de mapeo de series temporales multidimensionales X={x_t }_(t=1)^T a una etiqueta o secuencia de etiquetas Y={y_t }_(t=1)^T donde cada x_t es un vector de mediciones sensor¬iales en el instante t (Arshad, Bilal, & Gani, 2022).

Modalidades y fuentes de datos: La modalidad de datos define la naturaleza de las señales empleadas en HAR, y suele clasificarse en tres grandes grupos:

Sensor-based HAR: utiliza sensores inerciales (acelerómetro, giroscopio, magnetómetro) y fisiológicos (ritmo cardiaco, temperatura) integrados en wearables o smartphones. Estos dispositivos registran señales con tasas de muestreo típicas entre 20 Hz y 200 Hz, permitiendo capturar dinámicas finas del movimiento.

Vision-based HAR: recurre a secuencias de vídeo (RGB, profundidad, datos de esqueleto) para extraer características espaciales y temporales mediante técnicas de visión por computador y reconocimiento de poses (Chen et al., 2021).

Ambient-based HAR: explota sensores distribuidos en el entorno (módulos de detección de presencia, presión en suelos, micrófonos) para inferir actividad sin contacto directo con el usuario (Demrozi, Pravadelli, Bihorac, & Rashidi, 2020).

Más recientemente, la fusión multimodal combinando datos inerciales y de vídeo o sensórica ambiental ha demostrado mejorar la robustez y la capacidad de generalización de los modelos (Bouton-Bessac, Brunello, & De Souza, 2023).

Categorías de actividades y taxonomía: Las actividades objeto de reconocimiento pueden agruparse según distintos criterios:

Atomicidad:

Atómicas: gestos aislados como levantar el brazo o girar la cabeza.

Compuestas: secuencias temporales de gestos atómicos, p. ej., subir escaleras.

Duración:

Puntuadas: acciones de corta duración y clara delimitación temporal.

Continuas: actividades prolongadas como caminar o trabajar de pie.

Contexto social:

Individuales: realizadas por una sola persona.

Grupales: que involucran interacción o sincronización entre varios sujetos.

Interacciones humano-objeto (HOI): requieren identificar tanto la acción como el objeto manipulado (Dutta et al., 2025)

Esta taxonomía resulta clave para diseñar conjuntos de datos apropiados y para seleccionar arquitecturas capaces de modelar variabilidad inter e intraclase (Chen et al., 2021).

Como podemos ver en la Figura 1, se encuentra la fórmula de etiquetado de secuencia no es más que la definición formal de un problema de etiquetado de secuencias (sequence labeling) en aprendizaje supervisado. Aquí X es la secuencia de entradas de longitud T, donde cada x_t es un vector de características de dimensión d.

Y es la secuencia de etiquetas correspondientes, una etiqueta y_t para cada vector x_t, tomada de un conjunto finito de clases {1,…,C}.

Figura 1
Fórmula de etiquetado de secuencia

$$X = \{x_t\}_{t=1}^{T}, \quad x_t \in \mathbb{R}^d$$

$$Y = \{y_t\}_{t=1}^{T}, \quad y_t \in \{1, \ldots, C\}$$

Nota: d es el número de canales sensoriales y C el número de clases de actividad (Arshad et al., 2022).

Existen dos modalidades principales de solución:

Clasificación de ventanas: se segmenta X en ventanas deslizantes de longitud w y desplazamiento s, clasificando cada ventana completa.

Etiqueta por instante: se emplean modelos secuenciales (RNN, Transformers) para asignar etiquetas y_t en cada instante.

En ambos casos, la calidad del etiquetado depende de la estrategia de segmentación, la calidad de las anotaciones manuales y la capacidad del modelo para capturar dependencias temporales (Ramanujam, Perumal, & Padmavathi, 2021).

Evolución de los métodos: de Machine Learning clásico a Deep Learning. En la última década, HAR ha transitado desde enfoques basados en ingeniería manual de características y clasificadores clásicos (SVM, k-NN, HMM, Random Forest) hacia modelos de aprendizaje profundo que operan directamente sobre datos crudos (Qureshi, Shahid, Farhan, & Alamri, 2025).

Machine Learning clásico. requiere selección de características en dominio temporal (media, varianza, curtosis) y frecuencial (FFT, entropía espectral) antes de entrenar un clasificador (Demrozi et al., 2020).

Deep Learning. las redes convolucionales 1D/2D/3D (CNN) extraen automáticamente patrones espaciales y temporales; las redes recurrentes (LSTM/GRU) modelan dependencias a largo plazo; los Transformers aplican mecanismos de self-attention para capturar relaciones globales en la secuencia (Chen et al., 2021).

En estudios comparativos, las 1D-CNN dominan por su eficiencia y menor necesidad de preprocesamiento, seguidas de arquitecturas híbridas CNN-LSTM y Transformers, que presentan mejor capacidad de generalización en datos no vistos (Qureshi et al., 2025).

Componentes del pipeline de HAR. Un sistema HAR típico consta de tres bloques principales:

Adquisición de datos

Dispositivos: smartphones, smartwatches, cámaras RGB-D, sensores ambientales.

Parámetros: frecuencia de muestreo (20–200 Hz), resolución de ejes, sincronización temporal entre sensores.

Preprocesamiento y segmentación

Filtrado: eliminación de ruido y deriva, p. ej., filtros pasa-bajo Butterworth de segundo orden.

Normalización: escalado de cada canal para reducir sesgos.

Segmentación: ventanas fijas o adaptativas (event-driven), con solapamiento típico de 50 % (Dentamaro et al., 2024)

Ingeniería de características. Dominio temporal: media, varianza, desviación estándar, rango intercuartílico, número de cruces por cero.

Dominio frecuencial: coeficientes FFT, energía espectral, entropía de Shannon.

Time–frequency: transformada wavelet, STFT.

Selección y reducción: métodos filtro (correlación, información mutua), wrapper (SFS, SBS) y embedded (LASSO, árboles) (Demrozi et al., 2020).

Clasificación. Se consideran los siguientes:

Tradicional: SVM con kernel RBF, k-NN euclídeo, Random Forest con cientos de árboles, HMM para modelado secuencial.

Deep Learning: CNN 1D/2D con capas de pooling y normalización por lotes. RNN (LSTM/GRU) con puertas para largas dependencias. Transformers con codificación posicional y módulos de atención multi-cabeza (Chen et al., 2021).

Ensamblados: combinaciones de varias arquitecturas para robustez extra (Ramanujam et al., 2021).

Importancia y aplicaciones de HAR. El Reconocimiento de Actividad Humana (HAR) ha cobrado relevancia en la última década debido a su capacidad para transformar datos de sensores en información accionable, mejorando tanto la calidad de vida de las personas como la eficiencia de sistemas industriales y de servicios.

Salud y cuidado de pacientes. HAR permite la monitorización continua de individuos con enfermedades crónicas o en rehabilitación, detectando caídas, desviaciones en los patrones de movilidad y cumplimiento de ejercicios terapéuticos sin necesidad de supervisión humana constante. Por ejemplo, los sistemas basados en acelerómetros han alcanzado precisiones superiores al 95 % en la detección de caídas, lo cual es crítico para el cuidado de adultos mayores en entornos domiciliarios y hospitalarios (Zhang et al., 2024). Asimismo, Har facilitado la telemedicina al proporcionar datos objetivos sobre la actividad diaria, lo que ayuda a los profesionales a ajustar tratamientos de manera oportuna (Bibbò & Serrano, 2025).

Entornos inteligentes y domótica. En hogares y edificios inteligentes, HAR sirve para adaptar de forma autónoma la climatización, iluminación y sistemas de seguridad en función de la actividad y presencia de los usuarios. La fusión de datos inerciales con detección ambiental (presión, sonido, vídeo) ha demostrado reducir el consumo energético hasta en un 20 % al activar recursos solo cuando se requiere (Bouton-Bessac, Brunello, & De Souza, 2023; Demrozi et al., 2020).

Seguridad y vigilancia. Los modelos de HAR vision-based identifican comportamientos anómalos en espacios públicos y privados, como personas merodeando en zonas restringidas o realizando movimientos súbitos indicativos de peligro. Gracias a

arquitecturas basadas en Transformers, es posible procesar múltiples flujos de vídeo en tiempo real y emitir alertas automáticas con baja tasa de falsos positivos (Chen et al., 2021).

Deporte y entretenimiento. En el ámbito deportivo, HAR se aplica al análisis del rendimiento de atletas mediante wearables que registran parámetros de potencia, cadencia y técnica de movimiento. Esto permite diseñar planes de entrenamiento personalizados y prevenir lesiones (Ramanujam, Perumal, & Padmavathi, 2021). En videojuegos y realidad aumentada, la interpretación de gestos y posturas en tiempo real enriquece la experiencia inmersiva y abre nuevas formas de interacción hombre-máquina (Qureshi, Shahid, Farhan, & Alamri, 2025).

Industria y ergonomía. En entornos de manufactura y logística, HAR identifica tareas repetitivas o posturas forzadas que pueden derivar en trastornos musculoesqueléticos, sugiriendo pausas o ajustes ergonómicos. Además, el seguimiento de operarios y maquinaria facilita la planificación de mantenimientos predictivos, optimizando la productividad y reduciendo tiempos de inactividad (Arshad, Bilal, & Gani, 2022).

En conjunto, el reconocimiento de actividad humana se ha convertido en una tecnología transversal con aplicaciones críticas en salud, eficiencia energética, seguridad, entretenimiento e industria. Su desarrollo continuo, en particular mediante técnicas de aprendizaje profundo y fusión multimodal, promete ampliar aún más su impacto social y comercial en los próximos años.

Enfoques tradicionales de Machine Learning. Los primeros sistemas de HAR se basaron en algoritmos de aprendizaje automático clásico que requieren una etapa previa de extracción de características manuales. Entre los más empleados figuran los Support Vector Machines (SVM), k-Nearest Neighbors (k-NN), Hidden Markov Models (HMM), Random Forest (RF) y Gaussian Mixture Models (GMM). Estos métodos extraen estadísticas en

dominio temporal (media, varianza, curtosis) y frecuencial (coeficientes FFT, entropía espectral) para construir vectores de características, que luego son clasificados por el modelo (Demrozi et al., 2020; Arshad, Bilal, & Gani, 2022).

Aunque alcanzan precisiones razonables en conjuntos de datos balanceados, su rendimiento decrece ante variaciones en la posición del sensor, ruido y desequilibrios de clases. Además, dependen en gran medida de la calidad y diversidad de las características manuales diseñadas por expertos, lo que limita su escalabilidad y adaptabilidad a nuevas actividades o dominios (Demrozi et al., 2020; Arshad et al., 2022).

Redes Neuronales Convolucionales (CNN). Las CNN introducen la capacidad de aprender jerárquicamente filtros espaciales y temporales directamente de los datos crudos, eliminando la necesidad de extracción manual de características.

CNN 1D para datos inerciales. En HAR con sensores inerciales, las 1D-CNN procesan series temporales triaxiales (acelerómetro, giroscopio) mediante convoluciones unidimensionales a lo largo del tiempo. Estas redes capturan patrones locales de movimiento y ofrecen eficiencia computacional para dispositivos embebidos.

Ramanujam, Perumal y Padmavathi (2021) mostraron que las 1D-CNN superan a SVM y k-NN en varios datasets populares, mientras que Qureshi et al. (2025) evidencian que arquitecturas residuales (ResNet-1D) mejoran la estabilidad del entrenamiento y alcanzan precisiones superiores al 95 % en actividades básicas.

CNN 2D para vídeo y esqueleto. Para modalidades basadas en vídeo, las 2D-CNN operan sobre tramas individuales o mapas de características (por ejemplo, imágenes de flujo óptico o mapas de poses). Chen et al. (2021) describen arquitecturas como ResNet-50 o InceptionV3 adaptadas a HAR, que extraen representaciones espaciales ricas. Dentamaro et

al. (2024) integran 2D-CNN con técnicas de preprocesamiento (normalización, recorte) para mejorar la robustez ante variaciones lumínicas y de fondo, alcanzando hasta un 90 % de precisión en datasets complejos como UCF-101 y HMDB51.

CNN 3D para fusión espacio-temporal. Las 3D-CNN extienden la convolución a un cubo espacio-temporal, procesando secuencias de frames para capturar simultáneamente características espaciales y dinámicas. Zhang et al. (2024) presentan un "Dual-Stream C3D" que combina dos flujos de 3D-CNN para vídeo RGB y mapas de profundidad, logrando más de 97 % de precisión en detección de caídas. Dutta, Boongoen y Zwiggelaar (2025) discuten variantes ligeras de C3D que reducen parámetros sin sacrificar exactitud, facilitando su uso en dispositivos con recursos limitados.

Máquinas de Vectores de Soporte (SVM). Las SVM se introdujeron originalmente para problemas de clasificación binaria, buscando un hiperplano que maximice el margen entre clases en un espacio de altas dimensiones (Cortes & Vapnik, 1995). En los últimos años, su robustez ante datos de alta dimensión y su capacidad de generalización las han mantenido como una opción de referencia en aplicaciones que van desde bioinformática hasta visión por computador.

Formalmente, dada una muestra de entrenamiento $\{(X_i, Y_i)\}_{i=1}^{n}$, donde $X_i \in R^d$ y $Y_i \in \{-1, +1\}$, la SVM resuelve el problema de optimización:

$$\frac{1}{2}\|w\|^2 + \mathrm{C}\sum_{i=1}^{n} \xi_i$$

sujeto a

$$y_i(W \cdot \phi(X_i) + b) \geq 1 - \xi_i,\ \xi_i \geq 0.$$

El parámetro $C > 0$ controla el equilibrio entre maximizar el margen y penalizar los errores de clasificación (Huang & Wang, 2021). La transformación $\phi(\cdot)$ permite aplicar el **truco del kernel**, de modo que el producto interno $\phi(X_i) \cdot \phi(X_j)$ se reemplace por funciones kernel como el Gaussiano o RBF, el polinómico o el sigmoide (Schölkopf & Smola, 2002; Lee, Park, & Kim, 2022).

SVM Con Grid Search. Para optimizar el rendimiento de las SVM, se suelen emplear técnicas de búsqueda de hiperparámetros. Grid Search es un método exhaustivo que evalúa todas las combinaciones posibles dentro de un espacio definido de parámetros, utilizando validación cruzada para estimar la métrica de interés (precisión, F1-score) (Pedregosa et al., 2022).

Metodología De Grid Search. La configuración típica de Grid Search para SVM incluye rangos discretos para C (por ejemplo, $10^{-3}\ a\ 10^{3}$y γ (para kernels RBF), así como distintos grados d y coeficientes r en kernels polinómicos. Cada tupla de parámetros se evalúa mediante k-fold cross-validation, donde el dataset se divide en κ subconjuntos y se entrena k veces, promediando las métricas (Bengio & Grandvalet, 2021).

Eficiencia computacional y variantes. Aunque el Grid Search es sencillo de implementar (p. ej., con GridSearchCV en scikit-learn), su complejidad crece exponencialmente con el número de hiperparámetros y la granularidad de la malla. Para reducir tiempos, se han propuesto variantes:

Randomized Search. muestrea aleatoriamente un número fijo de combinaciones, ofreciendo resultados competitivos con mayor eficiencia (Bengio & Grandvalet, 2021).

Bayesian Optimization: Modela la función objetivo como un proceso Gaussiano y selecciona iterativamente los puntos más prometedores (Snoek, Larochelle, & Adams, 2012; aplicado a SVM por Nguyen et al., 2023).

Adaptive Grid Search: aumenta la resolución del espacio de parámetros cerca de la región óptima detectada por búsquedas previas (Abedin et al., 2023).

Pedregosa et al. (2022) reportan que combinaciones de Grid Search con dimensionality reduction (PCA previa) pueden reducir hasta un 40 % los tiempos de entrenamiento sin comprometer la precisión.

Árboles De Decisión. Los Árboles de Decisión son modelos jerárquicos que particionan recursivamente el espacio de características en subregiones mediante criterios de división basados en medidas de impureza, como la ganancia de información o el índice de Gini (Quinlan, 1986; Breiman et al., 1984). A pesar de su antigüedad, siguen siendo populares gracias a su interpretabilidad y facilidad de visualización.

Ventajas y desventajas. Las ventajas de los árboles incluyen su bajo coste computacional en entrenamiento, capacidad de manejar datos numéricos y categóricos sin preprocesamiento extenso, y su presentación gráfica intuitiva (Loh, 2020). No obstante, sufren de sobreajuste si no se poda adecuadamente, sensibilidad a pequeñas variaciones en los datos y limitaciones para modelar relaciones continuas suaves (Molinaro, Simon, & Pfeiffer, 2021).

Poda y generalización. La poda (pruning) busca eliminar ramas redundantes o ruidosas. Se distinguen dos estrategias:

Poda pre-pruning: detiene el crecimiento del árbol mediante criterios de profundidad máxima, mínimo número de muestras en nodo o ganancia de impureza mínima (Loh, 2020).

Poda post-pruning: primero se construye el árbol completo y luego se eliminan ramas con bajo rendimiento mediante validación cruzada, optimizando la relación complejidad-error (Molinaro et al., 2021).

Bosco et al. (2023) mostraron que la combinación de pre-pruning y post-pruning reduce el sobreajuste en un 15 % en datasets desequilibrados, mejorando la estabilidad del modelo.

Redes Recurrentes (RNN) y sus variantes. Las RNN, especialmente LSTM y GRU, modelan dependencias a largo plazo en secuencias. Ramanujam et al. (2021) comparan LSTM, GRU y redes feed forward para HAR con sensores en smartphones, demostrando que LSTM supera en un 3 % a las 1D-CNN cuando las actividades tienen solapamientos temporales complejos. Por su parte, Chen et al. (2021) integran mecanismos de atención a LSTM para ponderar automáticamente instantes críticos, mejorando la capacidad de distinguir actividades similares como subir y bajar escaleras.

Transformers y mecanismos de atención. Los Transformers, con su mecanismo de self-attention, han irrumpido recientemente en HAR. A diferencia de RNN, procesan toda la secuencia en paralelo, capturando relaciones globales. Chen et al. (2021) proponen un Transformer ligero que reemplaza la capa recurrente tras una extracción CNN, mientras que Bouton-Bessac, Brunello y De Souza (2023) demuestran que los Transformers superan a LSTM en datasets multimodales, siempre que se disponga de suficiente volumen de datos o se aplique preentrenamiento auto-supervisado.

Transfer Learning. El Transfer Learning es una técnica de aprendizaje automático que aprovecha el conocimiento adquirido en una tarea fuente para mejorar el rendimiento en una tarea objetivo relacionada (Pan & Yang, 2020). A diferencia de la formulación clásica de aprendizaje supervisado —donde los modelos se entrenan desde cero con grandes volúmenes de datos, el Transfer Learning permite reutilizar representaciones preentrenadas en dominios ricos en información, reduciendo la necesidad de datos etiquetados y acelerando el proceso de entrenamiento.

Para cuantificar el beneficio del Transfer Learning, se comparan tres escenarios: entrenamiento desde cero, extracción de características y fine-tuning. Se emplean métricas de precisión, recall y F1-score en tareas de clasificación, así como curvas de aprendizaje para evaluar la eficiencia de datos (Pan & Yang, 2020).

Modelos generativos y autoencoders. Los Generative Adversarial Networks (GAN) y los autoencoders se utilizan para data augmentation y preentrenamiento. Aleroud et al. (2024) desarrollaron un GAN de microagregación que preserva la privacidad al generar datos sintéticos realistas, mejorando la generalización de modelos de HAR en un 7 %. Por otro lado, Hu y Wang (2022) presentaron BSDGAN, que equilibra clases minoritarias generando señales inerciales, reduciendo el sesgo y elevando la precisión de reconocimiento en actividades raras de 78 % a 88 %.

Modelos ensamblados e híbridos. Los modelos ensamblados combinan varias arquitecturas para robustez: por ejemplo, 1D-CNN + LSTM + Transformer en un solo sistema. Dutta et al. (2025) describen un ensamblado de C3D y RNN con votación ponderada que logra un rendimiento estable ante ruido y variabilidad de sensores. Qureshi et al. (2025) muestran que los ensamblados con aprendizaje de stacking alcanzan mejoras de 2–4 % respecto a modelos individuales, especialmente en escenarios no vistos.

Sensores y preprocesamiento. La selección, disposición y tratamiento inicial de los datos sensoriales constituyen la fase crítica de todo sistema de Reconocimiento de Actividad Humana (HAR), ya que cualquier error en esta etapa repercute directamente en la calidad de los modelos de clasificación posteriores (Demrozi, Pravadelli, Bihorac, & Rashidi, 2020; Bouton-Bessac, Brunello, & De Souza, 2023). En efecto, la heterogeneidad de los sensores —en términos de rangos de medición, frecuencias de muestreo y características de ruido— exige protocolos rigurosos de adquisición y preprocesamiento para garantizar la reproducibilidad y la robustez de los sistemas HAR.

Dentro del amplio espectro de sensores utilizados en HAR, los sensores inerciales (acelerómetros, giroscopios y magnetómetros) son los más comunes por su miniaturización y bajo consumo energético. Los acelerómetros miden la aceleración lineal en uno o más ejes, mientras que los giroscopios registran velocidades angulares; ambos suelen integrarse en unidades de medición inercial (IMU) con frecuencias de muestreo típicas entre 20 y 200 Hz (Dentamaro, Gattulli, Impedovo, & Manca, 2024; Arshad, Bilal, & Gani, 2022). La calibración de sesgos y la compensación de deriva son esenciales para corregir errores sistemáticos que, de otro modo, inducirían desplazamientos acumulativos en el dominio del tiempo.

Las unidades de medición inercial (IMU) modernas combinan acelerómetro, giroscopio y a menudo magnetómetro, proporcionando datos multimodales que mejoran la diferenciación de patrones dinámicos. La ubicación del dispositivo (muñeca, cintura, tobillo o bolsillo) influye en las características de la señal y en la precisión del reconocimiento, por lo que numerosos estudios emplean estrategias de entrenamiento específicas según la posición del sensor (Ramanujam, Perumal, & Padmavathi, 2021; Demrozi et al., 2020). La variabilidad intersujeto, debida a diferencias anatómicas y comportamentales, persiste como un reto clave.

Más allá de los sensores inerciales, los sensores fisiológicos (electromiografía, electrocardiografía, fotopletismografía) aportan información complementaria sobre el estado corporal durante la ejecución de actividades. Las señales de electromiografía (EMG) miden la actividad muscular y resultan útiles para distinguir tareas que implican esfuerzos específicos, mientras que los datos de electrocardiograma (ECG) y fotopletismografía (PPG) permiten evaluar la intensidad del ejercicio y la fatiga (Arshad et al., 2022; Zhang et al., 2024). No obstante, estos sensores requieren un procesamiento adicional para eliminar artefactos y sincronizarse temporalmente con las señales inerciales.

En paralelo, los sensores ambientales y de proximidad (micrófonos, sensores de presión, cámaras RGB-D) facilitan el reconocimiento de actividades en entornos de interacción hombre-máquina y domótica. Por ejemplo, los mapas de profundidad generados por cámaras RGB-D permiten reconstruir posturas en tres dimensiones, mientras que los sensores de presión en el suelo detectan patrón de pasos y cambios de peso (Demrozi et al., 2020; Chen, Zhang, Yao, Guo, Yu, & Liu, 2021). La fusión de estos datos con señales inerciales, mediante técnicas de sincronización temporal y transformación de coordenadas, eleva la robustez a condiciones de oclusión o interferencia en un solo tipo de sensor.

La fusión multimodal de sensores es un proceso complejo que involucra la alineación temporal, la calibración espacial y la normalización de rangos de medición. Los métodos basados en transformada de Fourier o en representación wavelet conjunta permiten sincronizar y combinar características de distintas frecuencias, mientras que los enfoques de calibración rígida o no rígida corrigen desalineaciones entre ejes y unidades de diferentes fabricantes (Bouton-Bessac et al., 2023; Chen et al., 2021). Un preprocesamiento adecuado de fusión incrementa la capacidad de los modelos para generalizar ante variaciones contextuales.

El proceso de calibración comprende la estimación y corrección de sesgos (offset) y factores de escala de cada canal, así como el ajuste de ejes para garantizar ortogonalidad. Las técnicas de calibración in situ, basadas en movimientos controlados durante intervalos de descanso, permiten detectar desviaciones periódicas y compensar errores sistemáticos sin necesidad de equipamiento de laboratorio (Arshad et al., 2022; Dentamaro et al., 2024).

En la adquisición de datos, la sincronización temporal entre múltiples sensores resulta crítica. Los protocolos de muestreo deben gestionar latencias y pérdidas de paquetes en comunicaciones inalámbricas, adoptando marcas de tiempo de alta resolución o algoritmos de interpolación temporal para reconstruir series discontínuas (Ramanujam et al., 2021; Dentamaro et al., 2024). Asimismo, se monitorizan parámetros de calidad de señal para descartar fragmentos excesivamente ruidosos en tiempo real.

Una vez adquiridas las señales crudas, el filtrado de ruido y artefactos se realiza mediante filtros digitales diseñados a medida. Los filtros Butterworth de orden dos o cuatro son habituales para filtrado pasa-bajo (por debajo de 20 Hz) que atenúa el ruido de alta frecuencia, mientras que los filtros pasa-alto (por encima de 0.3 Hz) eliminan el desplazamiento de base. Para aplicaciones específicas, se emplean filtros Chebyshev o elliptic para obtener transiciones más pronunciadas entre bandas (Zhang et al., 2024; Demrozi et al., 2020).

Las técnicas de desruido avanzadas, como la denoising wavelet y la Empirical Mode Decomposition (EMD), permiten separar componentes de señal por su contenido espectral y energía, preservando la información relevante de la actividad. La denoising wavelet, por ejemplo, suprime coeficientes de pequeña magnitud en niveles de detalle altos, preservando las características de la señal base (Demrozi et al., 2020; Aleroud, Shariah, Malkawi, Khamaiseh, & Al-Alaj, 2024).

La segmentación de la señal en ventanas temporales es imprescindible antes de la extracción de características. Las estrategias más comunes utilizan ventanas deslizantes de longitud fija (entre 1 y 5 segundos) con solapamientos de 30–50 %, lo que equilibra la necesidad de mostrar dependencias temporales y la velocidad computacional (Ramanujam et al., 2021; Dentamaro et al., 2024). Alternativamente, los métodos basados en detección de eventos adaptativos ajustan dinámicamente la longitud de la ventana según cambios abruptos en la señal.

El tamaño de ventana y el porcentaje de solapamiento influyen en la capacidad del modelo para capturar transiciones entre actividades y en la cantidad de muestras disponibles para el entrenamiento. Ventanas muy cortas pueden fragmentar transiciones y producir etiquetas ambiguas, mientras que ventanas excesivamente largas incrementan la latencia de respuesta en aplicaciones en tiempo real (Chen et al., 2021; Ramanujam et al., 2021).

Una vez segmentadas las ventanas, se aplica la normalización y escalado de cada canal para homogenizar rangos de medición y reducir la influencia de valores extremos. Las técnicas de normalización típica (z-score) o min-max (0–1) facilitan la convergencia de algoritmos de aprendizaje profundo y evitan que características con rangos elevados dominen el proceso de optimización (Demrozi et al., 2020; Arshad et al., 2022).

Los datos faltantes o discontinuidades en la señal, originados por desconexiones de sensores o pérdidas de comunicación, se gestionan mediante interpolación lineal o spline, o a través de algoritmos de imputación basados en vecinos más cercanos (k-NN imputation). Estas técnicas restituyen valores plausibles sin sesgar significativamente la distribución original (Bouton-Bessac et al., 2023; Aleroud et al., 2024).

Finalmente, la detección de valores atípicos (outliers) complementa el preprocesamiento mediante métodos estadísticos, como el filtrado basado en desviaciones

estándar o técnicas de clustering que identifican grupos minoritarios de señales no representativas de la actividad normal (Dentamaro et al., 2024; Zhang et al., 2024).

Estos procedimientos de sensores y preprocesamiento establecen las bases para la posterior etapa de extracción de características y clasificación, garantizando que los modelos entrenados posean datos de alta calidad, libres de ruido y alineados temporalmente. La implementación rigurosa de cada paso —desde la calibración hasta la segmentación y la normalización— es esencial para desarrollar sistemas HAR precisos y generalizables en entornos reales.

Conjuntos de datos. En el campo del Reconocimiento de Actividad Humana (HAR), los conjuntos de datos públicos y privados constituyen la base para el desarrollo, la evaluación y la comparación objetiva de algoritmos y arquitecturas de aprendizaje automático y profundo. La disponibilidad de diversas colecciones de datos, con variabilidad en sensores, sujetos, actividades y condiciones de recolección, permite abordar retos de generalización, desequilibrio de clases y robustez contextual. Los análisis comparativos realizados en los últimos años destacan que la elección adecuada de un dataset influye directamente en el rendimiento y la aplicabilidad de los modelos en entornos reales (Dentamaro et al., 2024; Qureshi et al., 2025).

Criterios de selección y evaluación de dataset. Para seleccionar un conjunto de datos apropiado, es necesario evaluar múltiples criterios: modalidad de sensorización (inercial, visión, fisiológica y ambiental), frecuencia de muestreo, número y posición de sensores, cantidad de sujetos, diversidad demográfica, cantidad y tipo de actividades anotadas, protocolos de recolección (controlados vs. no controlados) y métodos de anotación (manual, semiautomático, basado en vídeo) (Demrozi et al., 2020; Arshad, Bilal, & Gani, 2022). Asimismo, el balance de clases y la disponibilidad de metadatos (edad, género,

condiciones de salud) son fundamentales para diseñar estrategias de entrenamiento y validación robustas.

UCI HAR Dataset. El UCI HAR Dataset se erige como el estándar de facto para validar técnicas de HAR con sensores de smartphone. Esta colección incluye datos de acelerómetro y giroscopio a 50 Hz de 30 sujetos realizando seis actividades básicas (caminar, subir y bajar escaleras, sentado, de pie y acostado). Aunque fue publicada originalmente en años anteriores, su caracterización detallada aparece en encuestas recientes que destacan su balance de clases y la claridad del protocolo de separación Leave-One-Subject-Out (LOSO) para evaluación sujeto-independiente (Dentamaro et al., 2024). Su adopción masiva facilita comparaciones directas entre estudios (Ramanujam, Perumal, & Padmavathi, 2021; Qureshi et al., 2025).

SisFall. El SisFall es un conjunto de datos público diseñado para la investigación de la detección de caídas y actividades de la vida diaria (ADL) mediante sensores portátiles. Según Cahoolessura y Rajkumarsingh (2020), SisFall consta de 4 505 archivos: 1 798 asociados a 15 tipos de caída y 2 707 correspondientes a 19 tipos de ADL, registrados con dos acelerómetros triaxiales (ADXL345 e ITG3200) y un giroscopio (MMA8451Q) a una frecuencia de muestreo de 200 Hz. Este diseño multicanal facilita el análisis de patrones dinámicos y la comparación de algoritmos basados en características de aceleración y rotación (Cahoolessura & Rajkumarsingh, 2020).

La población de estudio de SisFall incluye 23 adultos jóvenes (19–30 años) y 15 adultos mayores (60–75 años), permitiendo evaluar la generalización de modelos entrenados en distintos rangos etarios. Martinez et al. (2021) destacan que dentro de los participantes mayores solo un sujeto de 60 años simuló tanto ADL como caídas, lo que evidencia un sesgo

en la representación de caídas reales en ancianos, y limita la extrapolación a entornos clínicos gerontológicos (Martinez, Gonzalez, & Fernandez, 2024). Aun así, SisFall supera a otros datasets al incluir ADL de alta dinámica —como saltos y trote ligero— así como escenarios de casi-caída, acercándose a situaciones de la vida real (Frontiers in Aging Neuroscience, 2021).

WISDM (Wireless Sensor Data Mining). El WISDM dataset, compilado a partir de acelerómetros y giroscopios de smartphones colocados en el bolsillo frontal, contiene más de 1 millón de muestras de 36 sujetos realizando seis actividades. Con una frecuencia de muestreo de 20 Hz y ventanas de dos segundos con solapamiento del 50 %, este dataset ha sido clave en sistemas de aprendizaje profundo 1D-CNN gracias a su volumen y variabilidad intersujeto (Dentamaro et al., 2024). Estudios recientes han identificado desequilibrios menores entre clases, por lo que se han aplicado técnicas de re-muestreo y data augmentation para mejorar la generalización de los modelos (Bouton-Bessac, Brunello, & De Souza, 2023).

PAMAP2. Se diferencia por incluir múltiples sensores inerciales (acelerómetro, giroscopio, magnetómetro) en tres localizaciones corporales (muñeca, pecho y tobillo) junto a sensores fisiológicos (ritmo cardiaco). Con 18 actividades que abarcan tareas del hogar y ejercicio, capturadas a 100 Hz, PAMAP2 es particularmente valioso para evaluar arquitecturas multimodales y fusionadas. La riqueza de señales lo convierte en un referente para modelos 3D-CNN y CNN-LSTM, aunque impone desafíos de alta dimensionalidad y desequilibrios de duración entre actividades (Chen et al., 2021; Dentamaro et al., 2024).

Opportunity Dataset. El Opportunity Dataset, centrado en entornos de vida asistida, usa sensores inerciales en extremidades y objetos cotidianos instrumentados (puertas, electrodomésticos) para capturar actividades objetuales e interacciones HOI. Con 17 sujetos ejecutando esquemas de ADL (Activities of Daily Living) y 113 etiquetas posibles, su

complejidad sintáctica y semántica lo hace un banco de pruebas para Transformers y enfoques de few-shot learning. Publicaciones recientes subrayan la utilidad de este dataset para explicar decisiones de modelos gracias a su etiquetado fino y anotaciones vídeo-asistidas (Bouton-Bessac et al., 2023; Dutta, Boongoen, & Zwiggelaar, 2025).

MHEALTH. El MHEALTH dataset recoge señales inerciales y fisiológicas (ECG, PPG) de 10 sujetos realizando 12 actividades, incluidas posturas estáticas y ejercicios dinámicos, a 50 Hz. Destaca por su fusión sensor-inercial/fisiológico, permitiendo estudiar la correlación entre patrones de movimiento y respuestas fisiológicas. Su aplicación en modelos multimodales ha demostrado mejorar en un 8 % la detección de actividades específicas de alta intensidad al incorporar métricas cardiovasculares (Zhang et al., 2024; Arshad et al., 2022).

UniMiB SHAR. UniMiB SHAR se enfoca en la detección de caídas y actividades de la vida diaria, con datos de acelerómetro de 30 sujetos (26 mujeres y 4 hombres) a 50 Hz, segmentados en ventanas basadas en picos de aceleración. Con ocho clases de ADL y 17 tipos de caídas, ha sido empleado para evaluar algoritmos GAN-augmented y modelos CNN 1D ligeros en dispositivos wearables (Aleroud et al., 2024; Qureshi et al., 2025).

USC-HAD. El USC-HAD (University of Southern California Human Activity Dataset) incorpora sensores inerciales en tobillo y muñeca de 14 sujetos que realizan 12 actividades de la vida diaria. Publicado inicialmente con una frecuencia de 100 Hz, las encuestas modernas destacan su utilidad en estudios de transfer learning y cross-domain evaluation debido a su balance de clases y protocolo de grabación en interiores controlados (Demrozi et al., 2020; Dentamaro et al., 2024).

Otros datasets emergentes. En los últimos años han surgido colecciones enfocadas a entornos reales (in-the-wild) y a hábitos cotidianos, como ExtraSensory, que captura datos de smartphones en condiciones no controladas, y el dataset ADVIO para navegación en

interiores usando sensores del smartphone sin etiquetas manuales (Dentamaro et al., 2024; Qureshi et al., 2025). Asimismo, proyectos de Smart Homes como CASAS aportan datos multimodales ambientales y de posición, enriqueciendo los entornos de evaluación de HAR en domótica (Bouton-Bessac et al., 2023).

Desafíos y consideraciones éticas. La recolección de datos en HAR involucra consideraciones de privacidad, consentimiento informado y anonimización. La diversidad demográfica en los conjuntos de datos sigue siendo limitada, con sobrerrepresentación de jóvenes y población universitaria. Para superar sesgos, se recomienda ampliar la muestra a grupos etarios y contextos culturales diversos (Arshad et al., 2022; Bouton-Bessac et al., 2023). Además, la interoperabilidad entre datasets exige estandarización de formatos y etiquetados bajo el esquema ActivityNet o similares.

Capítulo 3

3. Desarrollo

3.1.Desarrollo del Trabajo

El proceso de desarrollo de un modelo de Machine Learning, enmarcado en el contexto del Proceso de Descubrimiento de Conocimiento en Bases de Datos (KDD - Knowledge Discovery in Databases), comienza con una rigurosa preparación de los datos. En este proyecto, se utilizó el conjunto de datos Human Activity Recognition Using Smartphones (UCI-HAR). Este dataset, fundamental para nuestro análisis, contiene series temporales multivariadas de sensores de aceleración y giroscopio de dispositivos móviles, registradas mientras un grupo de 30 voluntarios realizaba 6 actividades distintas.

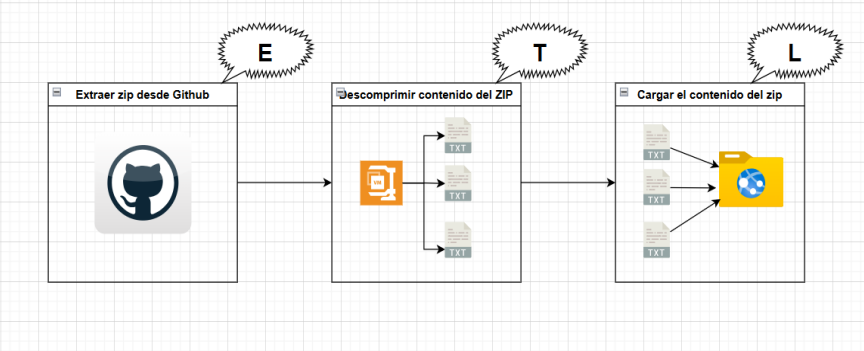
Las fases del Proceso KDD aplicadas son: Selección de datos, Preprocesamiento y transformación de datos, minería de datos.

Selección de Datos. Se seleccionó el dataset UCI-HAR, apropiado para el problema de reconocimiento de actividad humana, que incluye datos de acelerómetros y giroscopios de smartphones. Se obtuvo como un archivo .zip de un repositorio de GitHub.

Como podemos ver en la Figura 2, se implementó un proceso ETL básico para extraer el archivo comprimido del repositorio, Transformar los datos descomprimiendo el archivo, y cargar su contenido en un repositorio local para su posterior análisis y uso como insumo en el modelo.

Figura 2

Proceso básico ETL



Como se aprecia en la tabla 1, para la construcción de este proceso se utilizó como lenguaje de programación Python y las siguientes librerías implementadas en el siguiente bloque de código tal como se indica en la figura 3.

Tabla 1
Librerías de Python utilizadas.

Librería	Descripción
zipfile	Se utiliza para leer, comprimir y descomprimir archivos zip
io	Contiene métodos que nos permiten crear objetos tipo archivo en memoria

Figura 3

Código UCI HAR en github

```
[ ]  token_pat ="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"

     # URL RAW del archivo zip
     url_git = "https://raw.githubusercontent.com/JULIOARMANDOG/PRY_FINAL_MAESTRIA/main/uci-har.zip"
     path_to_data_uci_har="sample_data/data"
     message_download_ok="Archivo ZIP descargado correctamente"
     message_donwload_into_path="Archivo extraído en carpeta 'uci-har"
     message_download_error=" Error al descargar archivo:"

     # Crear cabecera donde se incluya el token PAT para la descarga del archivo zip
     headers = {"Authorization": f"token {token_pat}"}
     response_git = requests.get(url_git, headers=headers)

     if response_git.status_code == 200:
         print(f"✅{message_download_ok}")

         # Extraer zip en memoria y guardar contenido en carpeta 'uci-har'
         with zipfile.ZipFile(io.BytesIO(response_git.content)) as z:
             z.extractall(path_to_data_uci_har)
             print(f"✅ {message_donwload_into_path}")
     else:
         print(f"❌ {message_download_error} {response_git.status_code}")
         print(response_git.text)

⇥ ✅Archivo ZIP descargado correctamente
  ✅ Archivo extraído en carpeta 'uci-har
```

Carga y Etiquetado del Dataset SisFall (Recorrido de carpetas y recopilación de archivos): Esta fase implica la identificación y recopilación de los archivos de datos (.txt) desde las carpetas SAxx y SExx del dataset SisFall, y la asignación inicial de etiquetas binarias (caída/no-caída) basada en la estructura de directorios y nombres de archivos.

Preprocesamiento y Transformación de Datos. Esta fase es crucial para asegurar la calidad y el formato adecuado de los datos para su posterior modelado.

Las características del dataset UCI-HAR son:

Sensores. Acelerómetro y giroscopio triaxiales, capturando señales de aceleración lineal, aceleración angular y gravedad.

Frecuencia de muestreo. 50 Hz.

Segmentación. Los datos originales fueron segmentados en ventanas de 2.56 segundos (128 muestras) con una superposición del 50%. Estas ventanas representan la entrada principal para ambos modelos, donde podemos observar las actividades clasificadas que son:

1. Caminando (WALKING)

2. Subiendo escaleras (WALKING_UPSTAIRS)

3. Bajando escaleras (WALKING_DOWNSTAIRS)

4. Sentado (SITTING)

5. De pie (STANDING)

6. Acostado (LAYING)

Luego de la carga del dataset procedemos a la correspondiente validación de valores nulos, blancos dentro del dataset uci-har, tal como se menciona en las figuras 4-7.

Figura 4

Carga de datos de entrenamiento y prueba

```
#Carga data de entrenamiento y test desde el UCI-HAR
df_X_train=pd.read_csv("./sample_data/data/uci-har/train/X_train.txt",sep=r'\s+',header=None)  #pd.read_csv("./sample_data/data/uci-har/train/X_train.txt",sep=r'\s+',header=None)
df_y_train=pd.read_csv("./sample_data/data/uci-har/train/y_train.txt",sep=r'\s+',header=None,names=["activity_id"])
#Cargar los datos de la identificacion de actividades a predecir
df_activities=pd.read_csv("./sample_data/data/uci-har/activity_labels.txt",sep=r'\s+',header=None,names=["activity_id", "activity_label"])
#Para cada valor de la variable objetivo , vamos a asignarle una etiqueta
df_y_train_with_actions=df_y_train.merge(df_activities,on="activity_id")
#cargarmos las etiquetas de las variables predictoras
df_features=pd.read_csv("./sample_data/data/uci-har/features.txt",sep=r'\s+',header=None,names=["index", "feature_name"])
feature_names = pd.Series(df_features["feature_name"])
feature_names = feature_names.where(~feature_names.duplicated(), feature_names + "_" + feature_names.groupby(feature_names).cumcount().astype(str))
df_X_train.columns = feature_names

#Registramos el numero de veces que una actividad se repite dentro del dataset
#Devolviendo una serie en la cual la actividad se identifica como indice y el
#numero de ocurriencias de esa actividad como valor ; y ordenando ese resultado
#según el indice alfabetico o numérico
#num_activities=df_y_train["activity_label"].value_counts().sort_index()
#print(df_activities.head())
#Cargamos el dataSet de prueba
df_X_test=pd.read_csv("./sample_data/data/uci-har/test/X_test.txt",sep=r'\s+',header=None)
df_y_test=pd.read_csv("./sample_data/data/uci-har/test/y_test.txt",sep=r'\s+',header=None,names=["activity_id"])
df_X_test.columns=feature_names
df_y_test_with_actions=df_y_test.merge(df_activities,on="activity_id")

#map activity label with code activity
activity_map = dict(zip(df_activities["activity_id"], df_activities["activity_label"]))
df_y_train=df_y_train['activity_id'].map(activity_map)
df_y_test=df_y_test['activity_id'].map(activity_map)
```

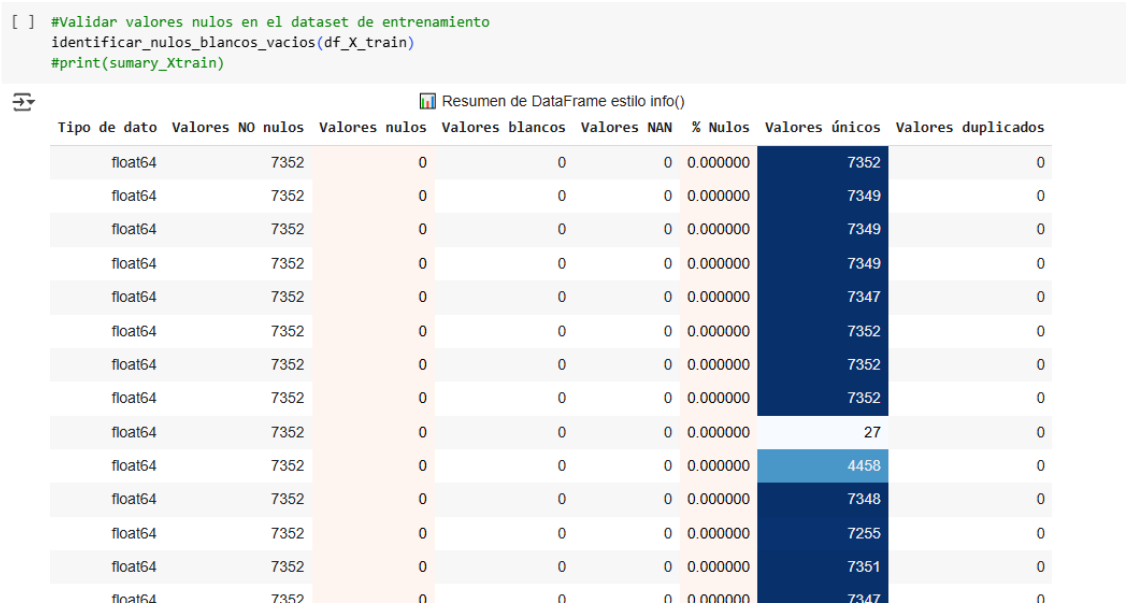
Figura 5

Validación de valores nulos, blancos, vacíos

```
[ ]  def identificar_nulos_blancos_vacios(data):
        sumary_validation=pd.DataFrame({
                'Tipo de dato':data.dtypes,
                'Valores NO nulos': data.notnull().sum(),
                'Valores nulos': data.isnull().sum(),
                'Valores blancos': (data == "").sum(),
                 'Valores NAN': data.isna().sum().sum(),
                '% Nulos': (data.isnull().sum() / len(data)) * 100,
                'Valores únicos': data.nunique(),
                'Valores duplicados': data.duplicated().sum()
            })
        sumary_validation.index.name="Columna"
        sumary_validation=sumary_validation.sort_values('% Nulos', ascending=False)

        styled_tabla = (
              sumary_validation.style
             .background_gradient(cmap='Reds', subset=['Valores nulos', '% Nulos'])
             .background_gradient(cmap='Blues', subset=['Valores únicos'])
             .set_caption("📊 Resumen de DataFrame estilo info()")
             .hide(axis="index")  # Opcional: oculta índice
         )


        #return sumary_validation
        return styled_tabla
```

Figura 6

Resultados validaciones de datos



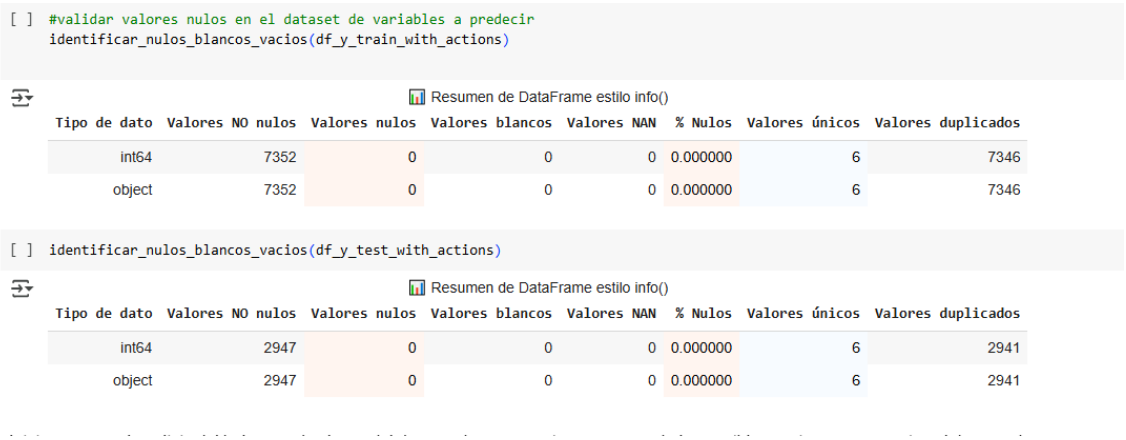
```
[ ] #Validar valores nulos en el dataset de entrenamiento
    identificar_nulos_blancos_vacios(df_X_train)
    #print(sumary_Xtrain)
```

Resumen de DataFrame estilo info()

Tipo de dato	Valores NO nulos	Valores nulos	Valores blancos	Valores NAN	% Nulos	Valores únicos	Valores duplicados
float64	7352	0	0	0	0.000000	7352	0
float64	7352	0	0	0	0.000000	7349	0
float64	7352	0	0	0	0.000000	7349	0
float64	7352	0	0	0	0.000000	7349	0
float64	7352	0	0	0	0.000000	7347	0
float64	7352	0	0	0	0.000000	7352	0
float64	7352	0	0	0	0.000000	7352	0
float64	7352	0	0	0	0.000000	7352	0
float64	7352	0	0	0	0.000000	27	0
float64	7352	0	0	0	0.000000	4458	0
float64	7352	0	0	0	0.000000	7348	0
float64	7352	0	0	0	0.000000	7255	0
float64	7352	0	0	0	0.000000	7351	0
float64	7352	0	0	0	0.000000	7347	0

Figura 7

Resultados validaciones de datos



```
[ ] #validar valores nulos en el dataset de variables a predecir
    identificar_nulos_blancos_vacios(df_y_train_with_actions)
```

Resumen de DataFrame estilo info()

Tipo de dato	Valores NO nulos	Valores nulos	Valores blancos	Valores NAN	% Nulos	Valores únicos	Valores duplicados
int64	7352	0	0	0	0.000000	6	7346
object	7352	0	0	0	0.000000	6	7346

```
[ ] identificar_nulos_blancos_vacios(df_y_test_with_actions)
```

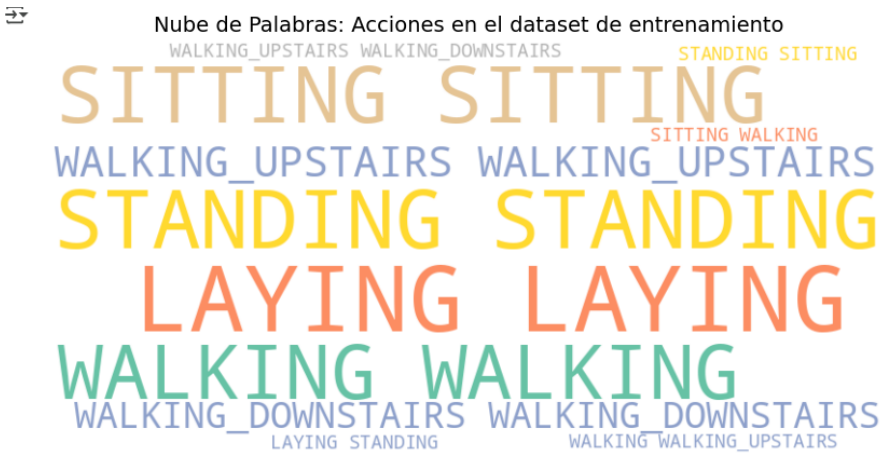
Resumen de DataFrame estilo info()

Tipo de dato	Valores NO nulos	Valores nulos	Valores blancos	Valores NAN	% Nulos	Valores únicos	Valores duplicados
int64	2947	0	0	0	0.000000	6	2941
object	2947	0	0	0	0.000000	6	2941

Al identificar que el conjunto de datos (dataset) no contiene información nula o vacía procedemos a realizar un análisis exploratorio de datos mostrando las diferentes acciones presentes dentro del conjunto de datos mediante el uso de una nube de palabras, tal como se muestra en la figura 8.

Figura 8

Nube de palabras UCI HAR



El conjunto de datos UCI-HAR contiene seis actividades distintas: LAYING, STANDING, SITTING, WALKING, WALKING_UPSTAIRS, y WALKING_DOWNSTAIRS. Esto delimita el alcance del proyecto a la predicción de estas seis acciones. Una vez identificadas las categorías, como se observa en la figura 9 se procede a evaluar el balanceo de clases en el conjunto de datos de entrenamiento, obteniendo un gráfico de barras en la figura 10.

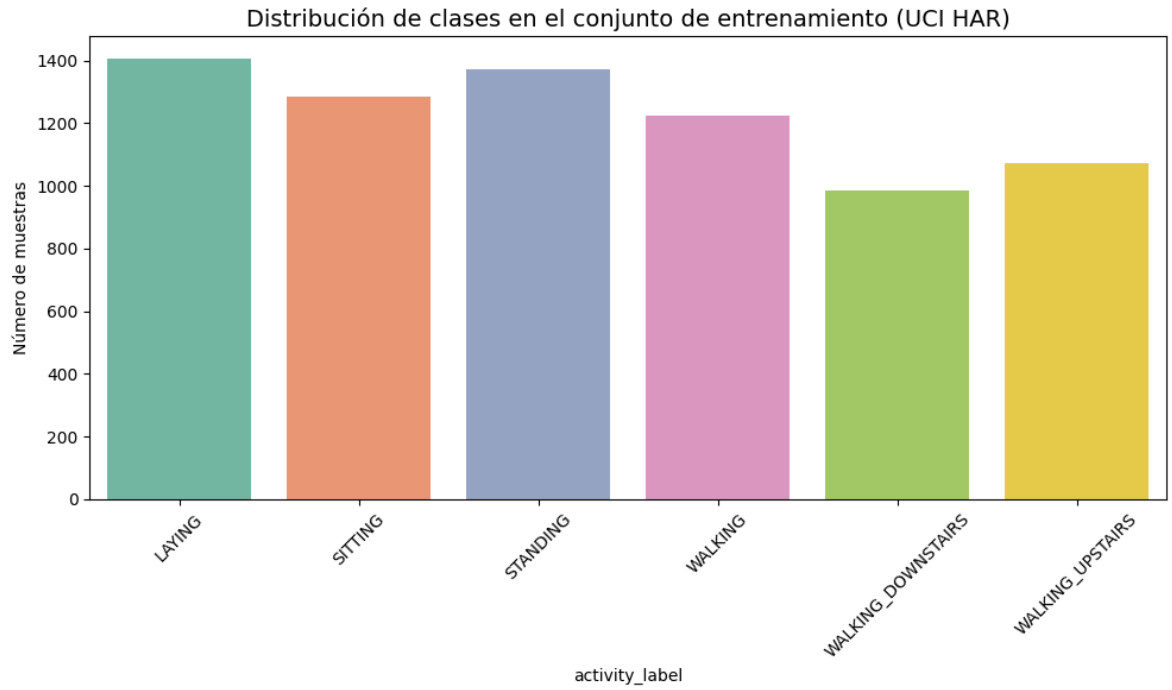
Figura 9

Código distribución de clases UCI HAR

```
def graficar_exploracion_inicial(data,title,labely) :
    plt.figure(figsize=(10, 6))
    sns.barplot(x=data.index, y=data.values,hue=data.index, palette="Set2")
    plt.title(f"{title}", fontsize=14)
    plt.ylabel(f"{labely}")
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
```

Figura 10

Distribución de las clases UCI HAR



El gráfico anterior permite una inferencia preliminar sobre el balanceo del conjunto de datos de entrenamiento. Aunque a primera vista no parece desbalanceado, para respaldar esta observación con datos precisos, se procederá a calcular el ratio entre las clases como se menciona en la figura 11.

Figura 11

Ratio Balanceo de clases

```
[ ]  #Vamos a utilizar el ratio entre clases mas comunes y menos comunes para
     #identificar si un dataset se encuentra desbalanceado ; el valor que se
     #tomara como umbral para determinar desbalanceo sera 1.5 (NO es regla extricta)
     #mas bien recomendaciones basadas en heuristica
     ratio_data_set=num_activities.max() / num_activities.min()
     if ratio_data_set > 1.5:
       print("El dataset esta desbalanceado")
     else :
       print("El dataset NO esta desbalanceado")

     El dataset NO esta desbalanceado
```

Como parte del análisis exploratorio, se utilizó un bloque de código para la Reducción de Dimensionalidad (PCA) como se visualiza en la figura 12, para visualizar la distribución de los puntos en cada clase. PCA permitió reducir las 561 características del conjunto de datos a 2 o 3 componentes principales, capturando la variabilidad de los datos.

Figura 12

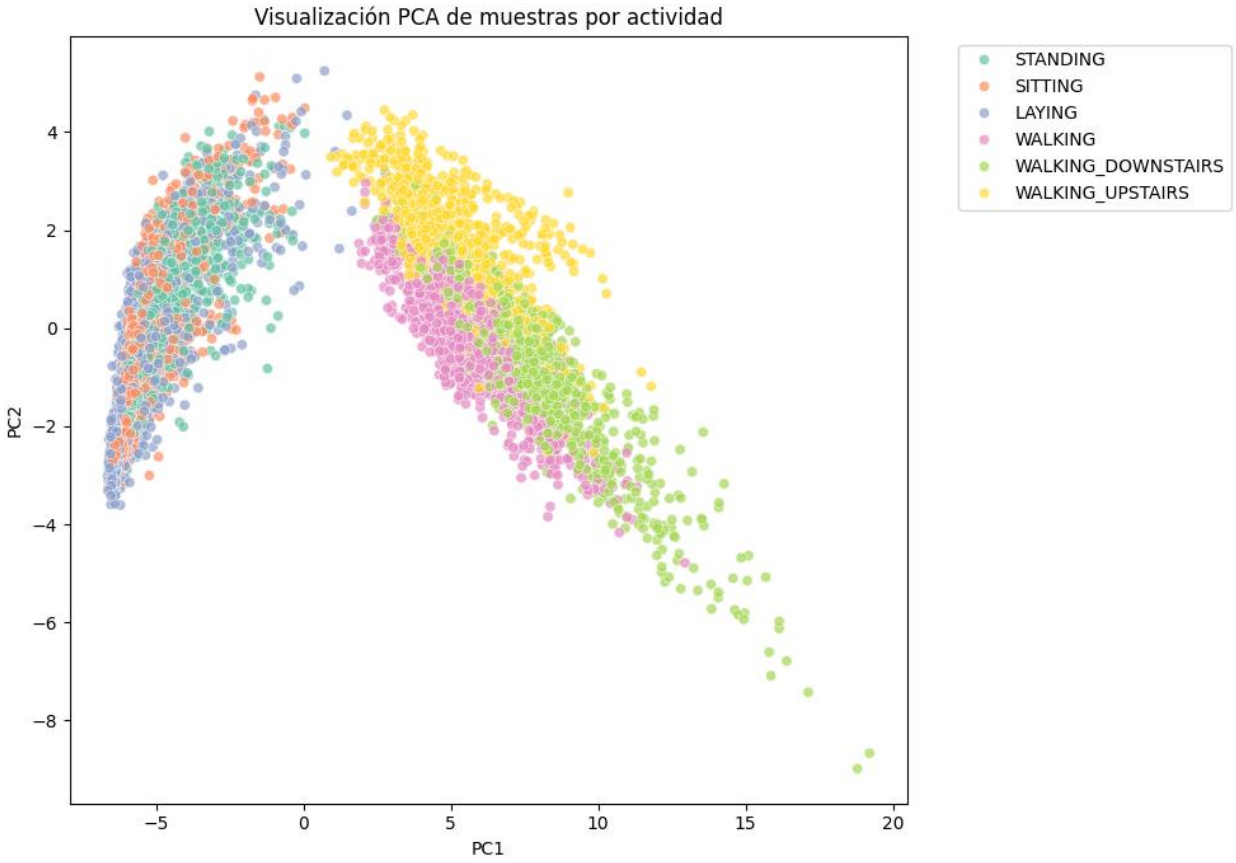
Reducción de dimensionalidad (PCA)

```
pca = PCA(n_components=2)
X_pca = pca.fit_transform(df_X_train)

# 2. Crear DataFrame para graficar
df_plot = pd.DataFrame({
    "PC1": X_pca[:,0],
    "PC2": X_pca[:,1],
    "activity_label": df_y_train_with_actions["activity_label"].values  # o la columna con clases
})

# 3. Graficar scatter plot coloreado por clase
plt.figure(figsize=(10,7))
sns.scatterplot(data=df_plot, x="PC1", y="PC2", hue="activity_label", palette="Set2", alpha=0.7)
plt.title("Visualización PCA de muestras por actividad")
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```

Como se muestra en la figura 13, esto facilitó la identificación de agrupaciones o mezclas entre las clases (visualizadas por colores) y la detección de patrones inusuales o outliers.

Figura 13

Visualización PCA por clases



El gráfico muestra un solapamiento entre las clases SITTING y STANDING, indicando una posible dificultad en su clasificación, mientras que LAYING, WALKING_UPSTAIRS y WALKING_DOWNSTAIRS están bien diferenciadas. Para mejorar la visualización de la separación entre clases en el dataset, se indica en la figura 14 la utilización de una técnica de reducción de dimensionalidad no lineal t-SNE (t-Distributed Stochastic Neighbor Embedding).

Figura 14

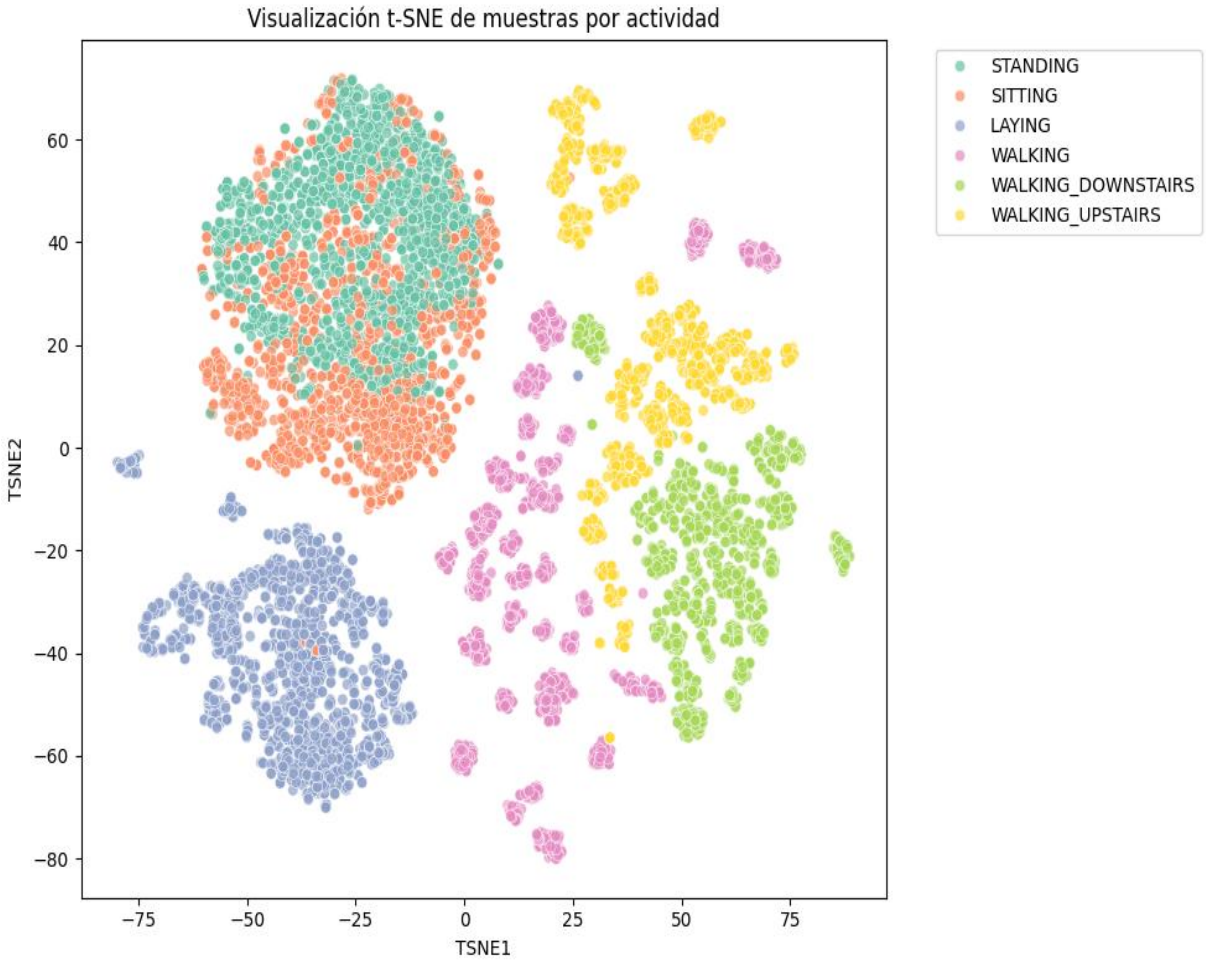
Reducción de dimensionalidad no lineal t-SNE

```
tsne = TSNE(n_components=2, random_state=42, perplexity=30, n_iter=1000)
X_tsne = tsne.fit_transform(df_X_train)


df_tsne = pd.DataFrame({
    "TSNE1": X_tsne[:, 0],
    "TSNE2": X_tsne[:, 1],
    "activity_label": df_y_train_with_actions["activity_label"].values
})

# Paso 3: Graficar
plt.figure(figsize=(10, 7))
sns.scatterplot(data=df_tsne, x="TSNE1", y="TSNE2", hue="activity_label", palette="Set2", alpha=0.7)
plt.title("Visualización t-SNE de muestras por actividad")
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```

Como se muestra en la figura 15, existe una mejor separación de clases en comparación con PCA, gracias a la reducción de dimensionalidad no lineal que preserva las relaciones locales. SITTING y STANDING, antes solapadas, ahora están más separadas, mientras que LAYING, WALKING, WALKING_UPSTAIRS y WALKING_DOWNSTAIRS aparecen como clústeres bien diferenciados.

Figura 15

Visualización t-SNE por clases



Procesamiento de Archivos SisFall (procesar_archivo). Como se observa en la figura 16 Dentro de esta función, la etapa de limpieza de datos ocurre al filtrar líneas para asegurar que solo se procesen aquellas con 9 números válidos (re.findall(r'-?\d+', linea)). También maneja archivos incompletos o sin datos válidos, omitiéndolos.

Figura 16

Función procesar archivos SisFall

```python
def procesar_archivo(ruta_archivo, etiqueta):
    try:
        with open(ruta_archivo, 'r') as f:
            lineas = f.readlines()

        datos_limpios = []
        for linea in lineas:
            numeros = re.findall(r'-?\d+', linea)
            if len(numeros) == 9:
                datos_limpios.append([int(n) for n in numeros])

        if not datos_limpios:
            raise ValueError("Archivo sin datos válidos")

        datos_bits = np.array(datos_limpios, dtype=np.float32)

        if len(datos_bits) < 128:
            print(f"Omitido: {ruta_archivo} (solo {len(datos_bits)} m
            return np.empty((0, 128, 9)), np.empty((0,))

        datos_fis = convertir_bits_a_unidades(datos_bits)
        X_vent, y_vent = crear_ventanas(datos_fis, etiqueta)
        return X_vent, y_vent

    except Exception as e:
        print(f"Error procesando {ruta_archivo}: {e}")
        return np.empty((0, 128, 9)), np.empty((0,))
```

Conversión de Unidades Físicas SisFall (convertir_bits_a_unidades). Aunque también es una transformación, se considera parte del preprocesamiento porque convierte los datos brutos de bits a unidades físicas significativas, eliminando una forma de "ruido" o falta de interpretabilidad en la representación original de los sensores como se observa en el código de la figura 17.

Figura 17

Convertir datos brutos a unidades físicas

```python
def convertir_bits_a_unidades(datos_bits):
    conv_adxl345 = (2 * 16) / (2**13)    # ~0.00390625 g/bit
    conv_itg3200 = (2 * 2000) / (2**16)  # ~0.06103515625 °/s per bit
    conv_mma8451q = (2 * 8) / (2**14)    # ~0.0009765625 g/bit

    datos_fisicos = np.empty_like(datos_bits, dtype=np.float32)
    datos_fisicos[:, 0:3] = datos_bits[:, 0:3] * conv_adxl345
    datos_fisicos[:, 3:6] = datos_bits[:, 3:6] * conv_itg3200
    datos_fisicos[:, 6:9] = datos_bits[:, 6:9] * conv_mma8451q

    return datos_fisicos
```

Transformación. Para cada modelo se utiliza características pre-extraidas o datos crudos.

Modelo RandomForestClassifier. Para este modelo, se utilizaron características pre-extraídas proporcionadas con el dataset UCI-HAR. Estas características incluyen estadísticas de dominio de tiempo y frecuencia (por ejemplo media, desviación estándar, energía, coeficientes de transformada de Fourier) calculadas sobre las ventanas de las señales de los sensores. Tal como se observa en la figura 18, las etiquetas de las clases fueron codificadas a un formato numérico adecuado para los algoritmos utilizando StandarScaler, presente en la librería de python sklearn.

Figura 18

Transformación de datos para RandomForestClassifier

```
[ ]  scaler=StandardScaler()
     scaler.fit(df_X_train)
     X_train_scaled=scaler.transform(df_X_train)
     X_test_scaled=scaler.transform(df_X_test)
```

Modelo CNN-1D. Para este modelo se procesaron los datos crudos directamente. A diferencia de enfoques que utilizan características extraídas, esta aproximación permite a la red aprender y extraer las características relevantes directamente de las series temporales. Para estandarizar las señales y asegurar un entrenamiento óptimo, se aplicó una normalización Z-score a todas las series temporales de cada ventana.

Tal como se muestra en la figura 19, este bloque de código organiza el conjunto de datos a la forma (n_samples, 128, 9) para la entrada de la CNN1D, donde 128 son los pasos de tiempo y 9 los canales/sensores. Normaliza cada señal individualmente (media 0, desviación estándar 1) para optimizar el entrenamiento, y hereda de tf.keras.utils.Sequence para un uso eficiente de memoria y paralelización con model.fit() y model.evaluate(). Además, simplifica la gestión de datos al ofrecer una opción integrada para la separación en conjuntos de entrenamiento y validación. Tras esta fase, se procede a construir el modelo CNN1D.

Figura 19

Transformación de datos para CNN-1D

```python
class ConvertirDataSetHAR(Sequence):
    def __init__(self, X, y, batch_size=64,validation_split=0.0):
        self.X = X
        self.y = y
        self.batch_size = batch_size
        self.n_samples, self.n_channels, self.seq_len = X.shape
        self.validation_split = validation_split
        #Normalizar los datos , covertir el array 3D de forma (numero_ejemplos, numero_canales, numero_paso_tiempo_por_muestra) a
        #una representacion 2D de forma (numero_ejemplos * numero_canales, numero_paso_tiempo_por_muestra).
        self.X=self.X.reshape(-1,self.seq_len)
        #cada señal de 128 puntos se normaliza a media 0 y desviacion estandar 1
        self.X = (self.X - self.X.mean(axis=1, keepdims=True)) / self.X.std(axis=1, keepdims=True)
        #se retorna a la estructura original del dataset 3D
        self.X = self.X.reshape(self.n_samples, self.n_channels, self.seq_len)

        # Transponer de (n_samples, 9, 128) → (n_samples, 128, 9)
        self.X = self.X.transpose(0, 2, 1)

        if validation_split > 0.0:
            split_index = int(self.n_samples * (1 - validation_split))
            self.X_train, self.X_val = self.X[:split_index], self.X[split_index:]
            self.y_train, self.y_val = self.y[:split_index], self.y[split_index:]
            self.indexes = np.arange(self.X_train.shape[0])  # Índices para el conjunto de entrenamiento
        else:
           # Índices de los ejemplos
           self.indexes = np.arange(self.X.shape[0])

    def __len__(self):
        # Número total de batches por época
        return int(np.ceil(self.n_samples / self.batch_size))

    def __getitem__(self, idx):
        # Devuelve un batch
        if self.validation_split > 0.0:
            batch_indexes = self.indexes[idx * self.batch_size:(idx + 1) * self.batch_size]
            batch_x = self.X_train[batch_indexes]
            batch_y = self.y_train[batch_indexes]
        else:
            batch_indexes = self.indexes[idx * self.batch_size:(idx + 1) * self.batch_size]
            batch_x = self.X[batch_indexes]
            batch_y = self.y[batch_indexes]
        return batch_x, batch_y

    def get_validation_data(self):
        # Devuelve los datos de validación si validation_split > 0.0
        if self.validation_split > 0.0:
            return self.X_val, self.y_val
        else:
            return None, None
```

Modelo SVM y SVM con GRIDSEARCH. En SVM para este modelo se está utilizando escalado de datos mediante el uso de la librería sklearn.preprocessing.StandardScaler, el objetivo de la misma es lograr que los datos del conjunto de datos uci-har tengan una media igual a cero (0) y una desviación estándar de uno (1), requisito crucial para muchos algoritmos de aprendizaje automático, incluyendo las máquinas de vectores de soporte (SVM) que son utilizadas en este proyecto.

Transformación de conjunto de datos SisFall.

Conversión de Unidades Físicas Sisfall (convertir_bits_a_unidades). En la figura 20, la función realiza una transformación escalar sobre los datos brutos, llevándolos de una representación digital (bits) a una representación física (g o °/s), lo cual es una transformación de características esencial para la interpretabilidad y el modelo

Figura 20

Transformación escalar de datos brutos.

```python
def convertir_bits_a_unidades(datos_bits):
    conv_adxl345 = (2 * 16) / (2**13)    # ~0.00390625 g/bit
    conv_itg3200 = (2 * 2000) / (2**16)  # ~0.06103515625 °/s per bit
    conv_mma8451q = (2 * 8) / (2**14)    # ~0.0009765625 g/bit

    datos_fisicos = np.empty_like(datos_bits, dtype=np.float32)
    datos_fisicos[:, 0:3] = datos_bits[:, 0:3] * conv_adxl345
    datos_fisicos[:, 3:6] = datos_bits[:, 3:6] * conv_itg3200
    datos_fisicos[:, 6:9] = datos_bits[:, 6:9] * conv_mma8451q

    return datos_fisicos
```

Ventaneado (Windowing) de Series Temporales SisFall (crear_ventanas). Aquí en la figura 21, el flujo continuo de datos de sensores se transforma en segmentos discretos (ventanas) con un tamaño y solapamiento definidos. Esto es una técnica de construcción de características temporal que adapta los datos para ser consumidos por un modelo de aprendizaje automático, especialmente para series de tiempo.

Figura 21

Ventana de series temporales.

```python
def crear_ventanas(datos, etiqueta, ventana=128, paso=64):
    X, y = [], []
    for i in range(0, len(datos) - ventana + 1, paso):
        X.append(datos[i:i+ventana])  # Forma (128, 9)
        y.append(etiqueta)
    return np.array(X), np.array(y)
```

Procesamiento de Archivos (procesar_archivo). La función integra la conversión a unidades físicas y el ventaneado, lo que la convierte en un paso clave de transformación antes de la agregación final.

Figura 22

Procesamiento de archivos

```python
def procesar_archivo(ruta_archivo, etiqueta):
    try:
        with open(ruta_archivo, 'r') as f:
            lineas = f.readlines()

        datos_limpios = []
        for linea in lineas:
            numeros = re.findall(r'-?\d+', linea)
            if len(numeros) == 9:
                datos_limpios.append([int(n) for n in numeros])

        if not datos_limpios:
            raise ValueError("Archivo sin datos válidos")

        datos_bits = np.array(datos_limpios, dtype=np.float32)

        if len(datos_bits) < 128:
            print(f"Omitido: {ruta_archivo} (solo {len(datos_bits)} muestras)")
            return np.empty((0, 128, 9)), np.empty((0,))

        datos_fis = convertir_bits_a_unidades(datos_bits)
        X_vent, y_vent = crear_ventanas(datos_fis, etiqueta)
        return X_vent, y_vent

    except Exception as e:
        print(f"Error procesando {ruta_archivo}: {e}")
        return np.empty((0, 128, 9)), np.empty((0,))
```

Concatenación de resultados finales. Este paso combina todas las ventanas procesadas de múltiples archivos en grandes arreglos NumPy (X_total, y_total), agregando los datos y preparándolos para la fase de minería de datos.

División de Datos. La división de los datos en conjuntos de entrenamiento y validación (train_test_split) es una transformación que prepara los datos para la fase de minería, asegurando una evaluación imparcial del modelo como se muestra en la figura 23.

Figura 23

División de datos, entrenamiento y validación

```
X_train, X_val, y_train, y_val = train_test_split(X_total, y_total, test_size=0.2, random_state=46)
```

Minería de Datos. En esta fase se aplican los algoritmos de Machine Learning y Deep learning. Se exploraron dos enfoques de modelado: una Red Neuronal Convolucional Unidimensional (CNN1D) y un clasificador de Bosques Aleatorios (RandomForestClassifier). Los datos transformados son "cargados" en la memoria para el entrenamiento de los modelos.

Modelo RandomForestClassifier. Es un algoritmo de aprendizaje supervisado versátil que pertenece a la familia de los métodos de ensamblaje. Construye múltiples árboles de decisión durante el entrenamiento y genera la clase que es la moda de las clases (clasificación) o la predicción media (regresión) de los árboles individuales. Es conocido por su robustez, su capacidad para manejar grandes conjuntos de datos con muchas características y su habilidad para reducir el sobreajuste.

Ventajas para este problema.

Robustez a Ruido. Al combinar múltiples árboles, el modelo se vuelve más robusto al ruido y a los valores atípicos presentes en las características.

Manejo de Dimensionalidad: Efectivo para conjuntos de datos con un número elevado de características, como es el caso de las características pre-extraídas del UCI-HAR.

Menor Sobreajuste. Tiende a sobreajustarse menos que un único árbol de decisión.

Tal como se aprecia en la figura 24, el modelo RandomForestClassifier se configuró inicialmente con sus parámetros por defecto, que son adecuados para una primera evaluación. Esto incluye un número de estimadores (árboles) predefinido y estrategias de división de nodos.

Figura 24

Código modelo RandomForestClassifier

```
[ ]  clasification_forest_model=RandomForestClassifier(n_estimators=100,random_state=46)
     #train model
     clasification_forest_model.fit(X_train_scaled,df_y_train)
     #Do prediction
     activity_prediction=clasification_forest_model.predict(X_test_scaled)
     #print report
     print(classification_report(df_y_test,activity_prediction))
```

Modelo CNN-1D. Procedemos a construir un modelo de red neuronal convolucional unidimensional (CNN1D), como se observa en la figura 25.

Figura 25

Función modelo CNN 1D

```python
def cnn_1d_model(input_shape, num_classes):
    # Definir la entrada
    inputs = tf.keras.Input(shape=input_shape)

    # Primera capa convolucional y de pooling
    x = tf.keras.layers.Conv1D(filters=32, kernel_size=3, activation="relu", strides=1, padding="valid")(inputs)
    x = tf.keras.layers.MaxPooling1D(pool_size=2)(x)

    # Segunda capa convolucional y de pooling
    x = tf.keras.layers.Conv1D(filters=64, kernel_size=3, activation="relu", strides=1, padding="valid")(x)
    x = tf.keras.layers.MaxPooling1D(pool_size=2)(x)

    # Tercera capa convolucional y de pooling
    x = tf.keras.layers.Conv1D(filters=128, kernel_size=3, activation="relu", strides=1, padding="valid")(x)
    x = tf.keras.layers.MaxPooling1D(pool_size=2)(x)

    # Aplanar y añadir Dropout
    x = tf.keras.layers.Flatten()(x)
    x = tf.keras.layers.Dropout(0.5)(x)

    # Capa de salida
    outputs = tf.keras.layers.Dense(num_classes, activation="softmax")(x)

    # Crear el modelo usando la API funcional
    model = tf.keras.Model(inputs=inputs, outputs=outputs, name='cnn1d_model')

    return model
```

La función define una entrada de 128 puntos de tiempo y 9 canales (acelerómetro y giroscopio en 3 ejes). Se establecen tres bloques sucesivos de capas convolucionales y de pooling para detectar secuencias de movimiento, reducir la dimensionalidad y capturar jerarquías de características. Una capa Flatten convierte la salida a un vector, y un Dropout de 0.5 evita el sobreajuste. Finalmente, una capa de salida con activación softmax produce la distribución de probabilidad para las 6 clases de actividad del UCI-HAR

Procedemos a entrenar el modelo, con la ejecución del bloque de código de las figuras 26, 27.

Figura 26

Código modelo CNN1D

```
X_train = load_signals("train")  # (n_samples, 9, 128)
y_train = load_labels("train")   # (n_samples,)
dataset = ConvertirDataSetHAR(X_train, y_train, batch_size=64,validation_split=0.2)
X_val, y_val = dataset.get_validation_data()
y_val = y_val["activity_id"].values
model = cnn_1d_model(input_shape=(128, 9), num_classes=6)
model.summary()
```

Figura 27

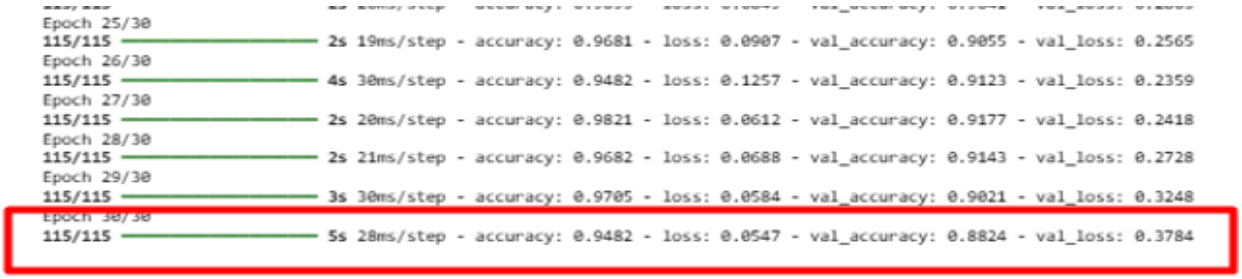
Código entrenamiento modelo CNN1D

```
model.compile(
        optimizer=Adam(learning_rate=0.001),
        loss=SparseCategoricalCrossentropy(),
        metrics=['accuracy']
    )

history=  model.fit(
        dataset,
        epochs=30,
        validation_data=(X_val, y_val),
       # batch_size=64,
        verbose=1
    )
```

Luego del entrenamiento vemos que el accuracy asociado al modelo en la última época fue de 88%, tal como se muestra en la figura 28.

Figura 28

Valores de última Época.



Svm y Svm con gridsearchCV. Uno de los algoritmos aplicados para abordar este proyecto, fue Support Vector Machine (SVM), un modelo de aprendizaje supervisado que busca encontrar el hiperplano óptimo que maximice la separación entre clases en un espacio multidimensional que tiene como objetivo clasificar correctamente la actividad que una persona está realizando en función de las características extraídas de sensores.

Para mejorar el rendimiento del modelo SVM, se emplea GridSearchCV, una técnica de búsqueda exhaustiva sobre un espacio de hiperparámetros. Acorde a la figura 29 en particular, se optimizan parámetros como:

C: parámetro de regularización que controla el equilibrio entre maximizar el margen y minimizar el error de clasificación para efectos de este modelo se lo estableció en 1.0

Random_state: Asegura reproducibilidad en ciertas implementaciones subyacentes o para la inicialización de los datos y el valor usado es 44.

kernel: tipo de función del núcleo (comúnmente 'linear', 'rbf', o 'poly') que transforma los datos para permitir una separación no lineal, para efectos de este proyecto se optó por linear esto implica que el modelo buscara un hiperplano lineal para las clases en el espacio de características.

Figura 29

Parámetros Modelo SVM

```
VM_ProyectoFinal.py > ...
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)

# Definir el modelo SVM
svm_model = SVC(kernel='linear', C=1.0, random_state=44)
# Entrenar el modelo SVM
svm_model.fit(X_train, y_train)

# Evaluar el modelo SVM
X_test_route = 'uci-har/test/X_test.txt'
```

Svm con gridsearchCV. Para el caso de SVM GridSearchCV como se indica en la figura 30 y 31, se implementaron los siguientes parámetros

C: Valores probados: [0.1, 1.0, 10.0, 100.0].

gamma: Valores probados: ['scale', 'auto', 0.001, 0.01, 0.1, 1]. El parámetro gamma define cuánto influye un solo ejemplo de entrenamiento; valores más grandes significan una influencia más cercana. scale y auto son estrategias predefinidas de scikit-learn.

kernel: Kernels probados: ['linear', 'rbf']. El kernel rbf (función de base radial) es una opción popular para problemas no lineales, permitiendo al SVM encontrar separaciones no lineales en los datos.

Figura 30

Parámetros Modulo GridSerarchCV

```
param_grid = {
    'C': [0.1, 1.0, 10.0, 100.0],
    'gamma': ['scale', 'auto', 0.001, 0.01, 0.1, 1],
    'kernel': ['linear', 'rbf']
}
```

Validación Cruzada: Se utiliza cv=5, lo que significa que la búsqueda de la mejor combinación de hiperparámetros se realiza mediante validación cruzada de 5 pliegues. Esto proporciona una estimación más robusta del rendimiento del modelo.

Métrica de Puntuación: La métrica utilizada para evaluar cada combinación de hiperparámetros es la accuracy (precisión).

Parallelización: n_jobs=-1 se usa para aprovechar todos los núcleos de la CPU disponibles y acelerar el proceso de búsqueda en la cuadrícula.

Figura 31

Módulo GridSearchCV

```
X_train = scaler.fit_transform(X_train)
#definimos la cuadricula de hiperparámetros
param_grid = {
    'C': [0.1, 1.0, 10.0, 100.0],
    'gamma': ['scale', 'auto', 0.001, 0.01, 0.1, 1],
    'kernel': ['linear', 'rbf']
}
# Crear el objeto GridSearchCV
grid_search = GridSearchCV(SVC(), param_grid, cv=5, scoring='accuracy', verbose=2, n_jobs=-1)
# Entrenar el modelo con GridSearchCV
grid_search.fit(X_train, y_train)
# Imprimir los mejores parámetros encontrados
print("Mejores parámetros encontrados:")
print(grid_search.best_params_)
```

Después de la búsqueda en la cuadrícula, tal como se muestra en figura 32 se obtiene el best_estimator_ (el mejor modelo encontrado). Este modelo es el que tiene la mejor puntuación de precisión según la validación cruzada con los parámetros optimizados. Finalmente, este best_model se utiliza para realizar predicciones en el conjunto de prueba (X_test_scaled) y de acuerdo a la figura 33, se evalúa de la misma manera que en el script anterior, con un reporte de clasificación, matriz de confusión y, adicionalmente, una curva ROC.

Figura 32

Best Estimator

```
print("Mejor puntuación:")
print(grid_search.best_score_)
# Obtener el mejor modelo
best_model = grid_search.best_estimator_
# Evaluar el modelo SVM
X_test_route = 'uci-har/test/X_test.txt'
y_test_route = 'uci-har/test/y_test.txt'
X_test = pd.read_csv(X_test_route, delim_whitespace=True, header=None)
y_test = pd.read_csv(y_test_route, delim_whitespace=True, header=None)
# Convertir y test a un array de numpy
y_test = y_test.values.ravel()
# Normalizar los datos de prueba
X_test_scaled = scaler.transform(X_test)
# Predecir con el mejor modelo SVM
y_pred = best_model.predict(X_test_scaled)
# Imprimir el reporte de clasificación
print(classification_report(y_test, y_pred))
# Imprimir la matriz de confusión
print(confusion_matrix(y_test, y_pred))
# Graficar la matriz de confusión
labels = ['WALKING', 'WALKING_UPSTAIRS', 'WALKING_DOWNSTAIRS', 'SITTING', 'STANDING', 'LAYING']
```

Figura 33

Modelo predictivo SVM

```
labels = ['WALKING', 'WALKING_UPSTAIRS', 'WALKING_DOWNSTAIRS', 'SITTING', 'STANDING', 'LAYING']
plt.figure(figsize=(8, 6))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d', cmap='Blues', xticklabels=labels, yticklabels=labels)
plt.title('Matriz de Confusión')
plt.xlabel('Predicción')
plt.ylabel('Realidad')
plt.show()
# Grafico de curva ROC
from sklearn.metrics import roc_curve, auc
from sklearn.preprocessing import label_binarize
# Binarizar las etiquetas
y_test_bin = label_binarize(y_test, classes=np.unique(y_test))
y_pred_bin = label_binarize(y_pred, classes=np.unique(y_test))
# Calcular la curva ROC
fpr, tpr, _ = roc_curve(y_test_bin.ravel(), y_pred_bin.ravel())
roc_auc = auc(fpr, tpr)
# Graficar la curva ROC
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', lw=2, label='ROC curve (area = {:.2f})'.format(roc_auc))
plt.plot([0, 1], [0, 1], color='red', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Curva ROC')
plt.legend(loc='lower right')
plt.show()
```

Modelo Transfer Learning.

Carga de Modelo Pre-entrenado: La carga de un modelo ya existente es el punto de partida para la estrategia de transfer learning, que es una técnica avanzada de minería de datos como se observa en la figura 34.

Figura 34

Carga de modelo Pre-entrenado.

```
modelo_preentrenado = load_model('model-uci-har.h5')
modelo_preentrenado.summary()
```

Congelamiento de Capas. Este paso modifica la arquitectura del modelo para el *fine-tuning*, que es el proceso de ajustar el modelo a la nueva tarea

Adaptación de la Capa de Salida. La modificación de la capa de salida para una clasificación binaria es parte de la configuración del algoritmo de minería para la tarea específica.

Compilación del Modelo de Transferencia. La compilación del modelo (definición de optimizador, función de pérdida y métricas) es la configuración final del algoritmo de aprendizaje antes de su ejecución.

Reentrenamiento (Fine-tuning). Esta es la fase central de minería de datos, donde el modelo aprende los patrones de los datos de entrenamiento (dataset SisFall) para la tarea de detección de caídas, ajustando sus pesos.

Figura 35

Modelo Transfer Learning

```
# Quitar la última capa y agregar una nueva
x = modelo_preentrenado.layers[-2].output  # Última capa oculta
salida = layers.Dense(1, activation='sigmoid')(x)

modelo_transfer = models.Model(inputs=modelo_preentrenado.input, outputs=salida)

modelo_transfer.compile(optimizer='adam',
                        loss='binary_crossentropy',
                        metrics=['accuracy'])
```

3.2.Marco Teórico

Proceso KDD: Conceptualización y evolución. El proceso KDD se define como un conjunto de pasos interrelacionados que facilitan la identificación de patrones útiles, válidos y novedosos en grandes bases de datos (Fayyad, Piatetsky-Shapiro, & Smyth, 2020). Este concepto emergió en la década de 1990, pero ha evolucionado sustancialmente para incorporar técnicas de Big Data, aprendizaje automático y analítica avanzada (Gupta &

Sharma, 2022). Actualmente, el KDD se considera un marco metodológico rigurosamente estructurado que integra tanto aspectos técnicos como de gestión de proyectos de datos (Smith & Carney, 2021).

El proceso KDD se compone de seis fases principales: selección, preprocesamiento, transformación, minería de datos, evaluación e interpretación y despliegue (Zhang & Li, 2020).

Selección. consiste en identificar y recopilar las fuentes de datos relevantes, asegurando cobertura suficiente y calidad inicial (Gupta & Sharma, 2022).

Preprocesamiento. se eliminan inconsistencias, valores faltantes y ruido, aplicando técnicas de limpieza y normalización (Lee & Park, 2021).

Transformación. genera nuevas variables o aplicaciones de reducción de dimensionalidad (p. ej., PCA), con el fin de optimizar la eficacia del análisis (Hernández & Pérez, 2023).

Minería de datos. aplicación de algoritmos de clasificación, clustering o regresión para descubrir patrones (Smith & Carney, 2021).

Evaluación e interpretación. evaluación de la utilidad y validez de los patrones mediante métricas de interés (precisión, recall, lift) y validación cruzada (Gupta & Sharma, 2022).

Despliegue. implementación de los conocimientos descubiertos en sistemas de soporte a la decisión o en flujos operativos (Zhang & Li, 2020).

Herramientas y entornos de KDD. En el ecosistema de KDD, se emplean tanto plataformas comerciales como de código abierto. Herramientas como KNIME, RapidMiner y

Orange proporcionan interfaces gráficas para diseñar pipelines de KDD sin necesidad de programación intensiva (Smith & Carney, 2021). Por otro lado, entornos como Python (scikit-learn, Pandas) y R (tidyverse, CARET) ofrecen una mayor flexibilidad y personalización, soportando integración con Big Data frameworks (Hadoop, Spark) para procesamiento distribuido (Gupta & Sharma, 2022; Hernández & Pérez, 2023).

Métricas de calidad y validación. La fase de evaluación en KDD es crítica para asegurar que los patrones descubiertos sean robustos y replicables. Se utilizan métricas estadísticas (p. ej., ROC-AUC, F1-score) y métodos de validación (k-fold, bootstrap) para estimar el comportamiento en datos no vistos (Lee & Park, 2021; Zhang & Li, 2020). Además, el análisis de significancia estadística (p-valor, intervalos de confianza) complementa la validación, aportando evidencia rigurosa sobre la fiabilidad de los resultados en contextos reales (Gupta & Sharma, 2022).

Aplicaciones y retos actuales del KDD. El KDD se ha consolidado en ámbitos como la detección de fraudes financieros, segmentación de clientes y salud predictiva (Smith & Carney, 2021). Sin embargo, enfrenta desafíos asociados a la gestión de datos no estructurados, la privacidad y el sesgo algorítmico (Zhang & Li, 2020). Las recientes regulaciones de protección de datos (GDPR, normas locales de privacidad) exigen incorporar prácticas de anonimización y fairness desde la fase de selección y preprocesamiento (Hernández & Pérez, 2023). Asimismo, la escalabilidad en entornos de streaming y la integración de técnicas de aprendizaje profundo representan líneas de investigación emergentes (Gupta & Sharma, 2022).

Proceso ETL Definición y características. El proceso Extract, Transform, Load (ETL) es la columna vertebral de los sistemas de integración de datos, permitiendo obtener información de múltiples fuentes, limpiarla y cargarla en destinos como data warehouses o

data lakes (Zhao & Chen, 2024). El ETL se enfoca en garantizar la calidad, consistencia y disponibilidad de los datos para su uso en reportes, dashboards y minería posterior (Kumar & Ahmad, 2024).

Etapa de extracción (Extract). La extracción implica conectarse a diversas fuentes (bases de datos relacionales, archivos planos, APIs, dispositivos IoT) y recuperar datos en su forma más pura (Zhang & Li, 2020). Se emplean conectores nativos o universales (JDBC, ODBC) y protocolos como REST o MQTT en entornos IoT. La sincronización puede ser batch (diaria, horaria) o real time (CDC: Change Data Capture) según los requisitos de frescura (Hernández & Pérez, 2023; Zhao & Chen, 2024).

Etapa de transformación (Transform). En la fase de transformación, los datos se limpian, estandarizan y enriquecen. Esto incluye:

Limpieza. corrección de valores atípicos, imputación de faltantes (mean, knn, MICE) (Kumar & Ahmad, 2024).

Estandarización. unificación de formatos de fecha, moneda y codificación de texto (Zhao & Chen, 2024).

Enriquecimiento. combinación con datos de referencia externos (geolocalización, catálogos maestros).

Agregaciones. cálculo de métricas como sumas, promedios o índices temporales (Smith & Carney, 2021).

Aplicación de reglas de negocio: validaciones específicas del dominio para garantizar la coherencia (Gupta & Sharma, 2022).

Etapa de carga (Load). Finalmente, la carga inserta los datos transformados en el destino deseado. En data warehouses, se prefiere la carga incremental para optimizar tiempos y recursos, mientras que en data lakes se pueden emplear formatos columnar (Parquet, ORC) para favorecer consultas analíticas (Zhao & Chen, 2024). Las arquitecturas modernas incorporan ELT (Extract, Load, Transform) cuando la transformación se realiza dentro del motor de base de datos o el sistema de Big Data, aprovechando su capacidad de cómputo masivo (Kumar & Ahmad, 2024).

Herramientas y arquitecturas ETL. El mercado ofrece soluciones on-premise y cloud. Entre las on-premise destacan Informatica PowerCenter, Talend y Microsoft SSIS, mientras que en la nube sobresalen AWS Glue, Google Cloud Dataflow y Azure Data Factory (Hernández & Pérez, 2023). Estas plataformas soportan pipelines code-free y code-first, orquestación de flujos y monitoreo avanzado. La arquitectura serverless y los contenedores Docker facilitan la escalabilidad y el despliegue continuo (Zhao & Chen, 2024).

Exploratory Data Analysis (EDA). El Análisis Exploratorio de Datos (EDA, por sus siglas en inglés) es un enfoque crítico en el ciclo de vida de los datos que busca comprender sus características principales antes de aplicar modelos de inferencia o predicción (Martinez & Gonzalez, 2020). Propuesto originalmente por Tukey, el EDA ha evolucionado con el auge de Big Data y las herramientas de visualización interactivas, convirtiéndose en un proceso sistemático de descubrimiento de patrones, anomalías y relaciones en los conjuntos de datos (Singh & Singh, 2021).

Objetivos del EDA. Los objetivos fundamentales del EDA incluyen:

Analizar la naturaleza de cada variable (numérica, categórica, temporal) y su distribución mediante medidas de tendencia central y dispersión (Martinez & Gonzalez, 2020).

Detectar anomalías y valores atípicos (outliers), que pueden indicar errores de medición o ejemplos interesantes para estudios de casos extremos (Johnson & Thompson, 2022).

Evaluar la calidad de los datos, identificando valores faltantes, inconsistencias y errores de captura que deben corregirse o imputarse (Brown, Davis, & Lee, 2023).

Revelar relaciones entre variables, tanto lineales como no lineales, a través de correlaciones, cruces de variables y análisis de dependencia (Zhang & Wu, 2024).

Formular hipótesis preliminares y guiar la selección de técnicas de modelado posteriores, reduciendo el riesgo de sesgos y malas especificaciones de modelo (Carter, 2025).

Fases del proceso de EDA. El EDA se articula típicamente en cuatro fases:

a) Obtener y describir los datos

Carga y muestreo: acceder a las fuentes (bases de datos, archivos planos, APIs) y, si son volumétricas, generar muestras representativas (Johnson & Thompson, 2022).

Resumen inicial: emplear funciones de descripción rápida (describe() en pandas o summary() en R) para obtener conteos, media, mediana, desviación estándar y percentiles (Martinez & Gonzalez, 2020).

b) Limpieza y tratamiento de valores faltantes

Identificación de faltantes: visualizar la proporción de datos faltantes por variable (heatmaps de missingness) (Brown et al., 2023).

Imputación: usar estrategias simples (media, moda) o avanzadas (k-NN, MICE) según la naturaleza de los datos y la proporción de faltantes (Brown et al., 2023).

Normalización de formatos: unificar formatos de fecha, texto y variables categóricas para garantizar consistencia (Zhang & Wu, 2024).

c) Análisis univariado

Distribución de variables numéricas: histogramas y densidades kernel (KDE) para evaluar forma (asimetría, curtosis) y detectar picos inusuales (Johnson & Thompson, 2022).

Análisis de variables categóricas: tablas de frecuencia, bar charts y count plots que revelan proporciones y posibles clases minoritarias desequilibradas (Singh & Singh, 2021).

Boxplots: útiles para comparar distribuciones y resaltar outliers en una o varias categorías (Carter, 2025).

d) Análisis bivariado y multivariado

Diagramas de dispersión (scatter plots): examinar relaciones de pares de variables numéricas, incluyendo líneas de tendencia y densidades sobreimpresas (Zhang & Wu, 2024).

Matriz de correlación: coeficiente de Pearson o Spearman para entender relaciones lineales y no lineales; representada como heatmap para facilitar la interpretación (Martinez & Gonzalez, 2020).

Diagramas de violín y swarm plots: combinan distribución y valores individuales para comparar grupos (Johnson & Thompson, 2022).

Mapas de calor multietapa: muestran patrones combinados de categóricas y continuas mediante agregaciones resumidas (Brown et al., 2023).

Técnicas avanzadas de EDA. Contamos con 2 principales técnicas PCA Y t-SNE.

El Análisis de componentes principales (PCA) reduce la dimensionalidad preservando la mayor parte de la varianza, facilitando la visualización de datos de alta dimensión en 2D o 3D y detectando agrupamientos o variables influyentes (Singh & Singh, 2021; Carter, 2025).

El t-Distributed Stochastic Neighbor Embedding (t-SNE) es una técnica no lineal de reducción de dimensionalidad enfocada en la visualización de datos de alta dimensión en espacios de dos o tres dimensiones. A través de la modelación de probabilidades de similitud entre puntos en el espacio original y en el espacio embebido, t-SNE minimiza la divergencia de Kullback-Leibler para conservar las relaciones locales, permitiendo identificar agrupamientos y estructuras subyacentes que podrían pasar desapercibidas con métodos lineales como la PCA (Kobak & Linderman, 2021).

Herramientas para EDA. Entre las más utilizadas se encuentran

1. Python: pandas, NumPy, matplotlib, seaborn, pandas_profiling, Sweetviz y ydata-profiling (Johnson & Thompson, 2022).
2. R: tidyverse (dplyr, ggplot2), DataExplorer, janitor y esquisse para dashboards interactivos (Martinez & Gonzalez, 2020).
3. Entornos visuales: Tableau y Power BI para análisis rápido y compartición de insights en dashboards corporativos (Carter, 2025).

Modelo RandomForestClassifier. El Random Forest es un método de ensamblado basado en bagging y árboles de clasificación (CART). Cada árbol se entrena sobre una muestra bootstrap de los datos originales, y en cada nodo se selecciona aleatoriamente un subconjunto de variables para dividir, reduciendo la correlación entre árboles y la varianza del modelo sin aumentar significativamente el sesgo (Probst, Boulesteix, & Bischl, 2020).

Matemáticamente, la predicción agregada para clasificación corresponde a la moda de las predicciones individuales:

$$\hat{y} = mode\{h_1(x), h_2(x), \dots, h_B(x)\},$$

Donde $h_b(x)$, es la predicción del árbol b, y B es el número de árboles (n_estimators). La incorporación de feature bagging tomar m variables al azar de las p totales en cada división es clave para la de correlación de árboles (Probst et al., 2020).

Escalado de datos. Las matrices de características X_train_scaled y X_test_scaled ya están normalizadas o estandarizadas. Aunque RandomForest no requiere escalado estricto, la consistencia en preprocesamiento ayuda cuando se comparan con otros modelos (Li & Zhu, 2024).

Entrenamiento con .fit(). El método fit genera árboles de clasificación, cada uno entrenado sobre muestras bootstrap de (X_{train}, Y_{train}).

Predicción con .predict() y evaluación. La llamada predict aplica cada árbol al conjunto de prueba y vota la clase final. classification_report muestra métricas como precisión (precision), sensibilidad (recall), F1-score y soporte (support) por clase, fundamentales para valorar el desempeño en problemas de clasificación múltiple.

Modelo Convolutional Neural Networks 1D (CNN-1D). Las Redes Neuronales Convolucionales unidimensionales (CNN-1D) han emergido como una de las arquitecturas más efectivas para el procesamiento de datos secuenciales y series temporales, tales como señales de sensores, electrocardiogramas, audio y registros de actividad. Se emplean filtros que operan a lo largo de la dimensión temporal, permitiendo la extracción jerárquica de

características locales y globales en secuencias de una sola dimensión (Ramanujam, Perumal, & Padmavathi, 2021).

Fundamentos y operación de la convolución 1D. La operación central en una CNN-1D es la convolución discreta de un filtro (kernel) de longitud (k) con una secuencia de entrada (x) de longitud (n):

$$(y * w)[t] = \sum_{i=0}^{k-1} \quad w[i]x[t+i], \dots, n-k+1,$$

Donde w representa el vector de pesos del kernel entrenable. Cada canal de entrada puede tener múltiples filtros independientes, generando mapas de características (feature maps) que capturan patrones temporales como picos, transiciones y formas de onda específicas (Chen et al., 2021). A lo largo de capas sucesivas, los kernels permiten detectar estructuras de mayor complejidad y alcance (receptive field ampliada) gracias a la naturaleza jerárquica de la red.

Arquitectura típica de una CNN-1D. Una CNN-1D estándar incluye varias etapas concatenadas:

Capa de Convolución 1D: con múltiples filtros de tamaños k_1, k_2, ..., cada uno aprende patrones locales de distinto ancho temporal.

Normalización y activación: normalmente se emplea Batch Normalization para estabilizar la distribución de la salida de cada filtro, seguida de funciones de activación ReLU o Leaky ReLU (Qureshi, Shahid, Farhan, & Alamri, 2025).

Capa de pooling 1D: MaxPooling1D o AveragePooling1D con tamaño de ventana reduce la dimensionalidad, retiene la característica más salient y aporta invarianza a pequeñas traslaciones temporales (Dentamaro et al., 2024).

Capas intermedias: repeticiones de bloques Conv–Norm–Act–Pool según profundidad deseada.

Flatten y Fully Connected: tras la última capa de pooling, los mapas se aplanan y se introducen en capas densas para clasificación o regresión, guiadas por softmax (clasificación) o linear (regresión).

Regularización: Dropout y L2 weight decay mitigan el sobreajuste, especialmente crítico en series con ruido o pocos ejemplos (Dutta, Boongoen, & Zwiggelaar, 2025).

Diseño de hiperparámetros. La eficacia de una CNN-1D depende de múltiples hiperparámetros:

Número y tamaño de filtros: mayores cantidades permiten capturar variedad de patrones; sin embargo, incrementan complejidad y riesgo de sobreajuste. Filtros cortos $(k = 3{,}5)$ detectan detalles locales, mientras que filtros más largos $(k = 15\ o\ más)$ capturan tendencias a medio plazo (Ramanujam et al., 2021).

Profundidad de la red: capas adicionales amplían el receptive field efectivo, pero exigen más datos y cuidados de regularización (Qureshi et al., 2025).

Stride y padding: el stride determina el desplazamiento del kernel (reduce longitud de salida), y padding (‘same’ o ‘valid’) controla si se conservan dimensiones originales.

Tamaño de pooling: ventanas de pooling más grandes reducen más información y pueden eliminar detalles relevantes; típicamente se elige $P = 2\ o\ P = 3$ (Dentamaro et al., 2024).

Tasa de aprendizaje y optimizador: optimizadores adaptativos como Adam o RMSprop, con learning rates en $[10^{-4}, 10^{-2}]$, facilitan convergencia en redes profundas (Ramanujam et al., 2021).

Modelo Svm Con Gridsearch. Las SVM son clasificadores supervisados que buscan un hiperplano óptimo para separar clases en un espacio de alta dimensión, maximizando el margen entre los vectores de soporte (Cortes & Vapnik, 1995). Para problemas no lineales, emplean la traducción al kernel, donde un kernel $K(x_i, y_i) = \phi(x_i)^T \phi\ (x_j)$, permite mapear datos a espacios de dimensión superior sin computar la función ϕ explícitamente (Schölkopf & Smola, 2002). Los kernels más comunes son el lineal, polinómico y RBF (Gaussiano) (Lee, Park, & Kim, 2022). Los hiperparámetros críticos incluyen:

C: factor de penalización de los vectores mal clasificados.

γ(para RBF): controla el alcance de influencia de cada vector de soporte.

Grado d y coeficiente r (para kernel polinómico).

La selección adecuada de estos parámetros es determinante: un C muy alto puede inducir **sobreajuste**, mientras que un γ inapropiado degrada la **generalización** (Huang & Wang, 2021).

Metodología de Grid Search para SVM. El Grid Search consiste en definir un espacio cartesiano de posibles valores para cada hiperparámetro y evaluar todas las combinaciones mediante validación cruzada (CV). El flujo general es:

Definir rangos de parámetros: elegir conjuntos discretos para C, γ, grado d, etc.

Selección de esquema de CV: comúnmente k-fold (por ejemplo, k=5 o 10) para estimar la métrica media (accuracy, F1-score) en datos no vistos (Pedregosa et al., 2022).

Evaluación exhaustiva: entrenar un modelo SVM por cada combinación y promediar los resultados de CV.

Selección de la mejor combinación: elegir la tupla de parámetros que maximiza la métrica objetivo.

Ajuste final: reentrenar el SVM con los hiperparámetros seleccionados sobre el conjunto completo de entrenamiento.

Este procedimiento, pese a su carácter exhaustivo, garantiza hallar la mejor configuración dentro del espacio definido (Abedin, Rahman, & Chowdhury, 2023). No obstante, su complejidad computacional crece exponencialmente con el número de parámetros y la granularidad de la malla (Probst, Boulesteix, & Bischl, 2020).

Capítulo 4

4. Análisis De Resultados

RandomForestClassifier. Al ser un modelo basado en árboles de decisión, no requiere un proceso iterativo de "épocas" y "lotes" como las redes neuronales. Se entrena en una sola pasada sobre el conjunto de datos.

La evaluación se realiza directamente sobre el conjunto de prueba (o validación) una vez que el modelo ha sido ajustado. Como se observa en la figura 36 las métricas clave incluyen precisión, recall, F1-score.

Figura 36

Reporte de clasificación RandomForestClassifier

	precision	recall	f1-score	support
LAYING	1.00	1.00	1.00	537
SITTING	0.91	0.88	0.89	491
STANDING	0.89	0.92	0.90	532
WALKING	0.89	0.97	0.93	496
WALKING_DOWNSTAIRS	0.96	0.86	0.90	420
WALKING_UPSTAIRS	0.90	0.90	0.90	471
accuracy			0.92	2947
macro avg	0.92	0.92	0.92	2947
weighted avg	0.92	0.92	0.92	2947

El modelo alcanzó una precisión del 92%, demostrando su efectividad en la predicción de actividades humanas a partir de señales de sensores móviles.

Se procede a identificar la importancia de las variables en estas predicciones de acuerdo al código de la figura 37.

Figura 37

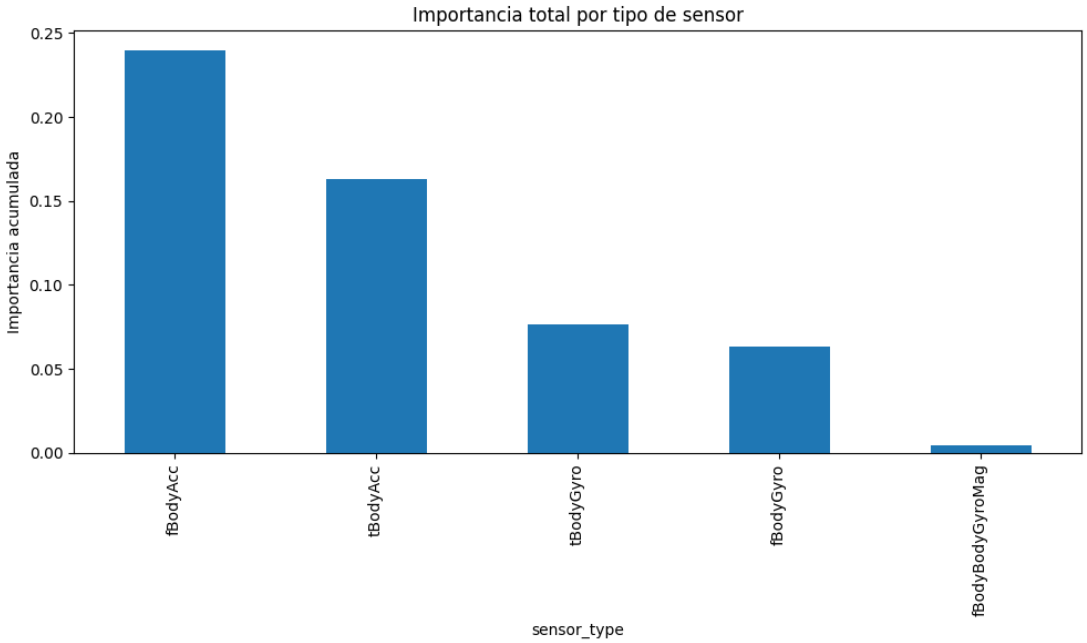
Código Importancia de las variables por tipo de sensor

```
importances = clasification_forest_model.feature_importances_
df_importances = pd.DataFrame({
    "feature": df_X_train.columns,
    "importance": importances
}).sort_values(by="importance", ascending=False)

df_importances["sensor_type"] = df_importances["feature"].str.extract(r'^([tf]Body(?:Acc|Gyro|AccJerk|GyroJerk|BodyAccMag|BodyGyroMag|BodyBodyGyroJerkMag|BodyBodyAccJerkMag|BodyBodyGyroMag|BodyBodyAccMag))')

# Agrupar por tipo de sensor
sensor_importance = df_importances.groupby("sensor_type")["importance"].sum().sort_values(ascending=False)

sensor_importance.plot(kind='bar', figsize=(10, 6), title="Importancia total por tipo de sensor")
plt.ylabel("Importancia acumulada")
plt.tight_layout()
plt.show()
```

Como se observa en la figura 38, el resultado es un gráfico de barras que indica la importancia asignada por el modelo a cada variable predictora al momento de inferir acciones humanas.

Figura 38

Distribución Importancia total por tipo de sensor



Como paso final, se evalúa el rendimiento del modelo en la clasificación de las seis actividades humanas presentes en el dataset UCI-HAR mediante el código de la figura 39, del uso de la matriz de confusión representado en la figura 40.

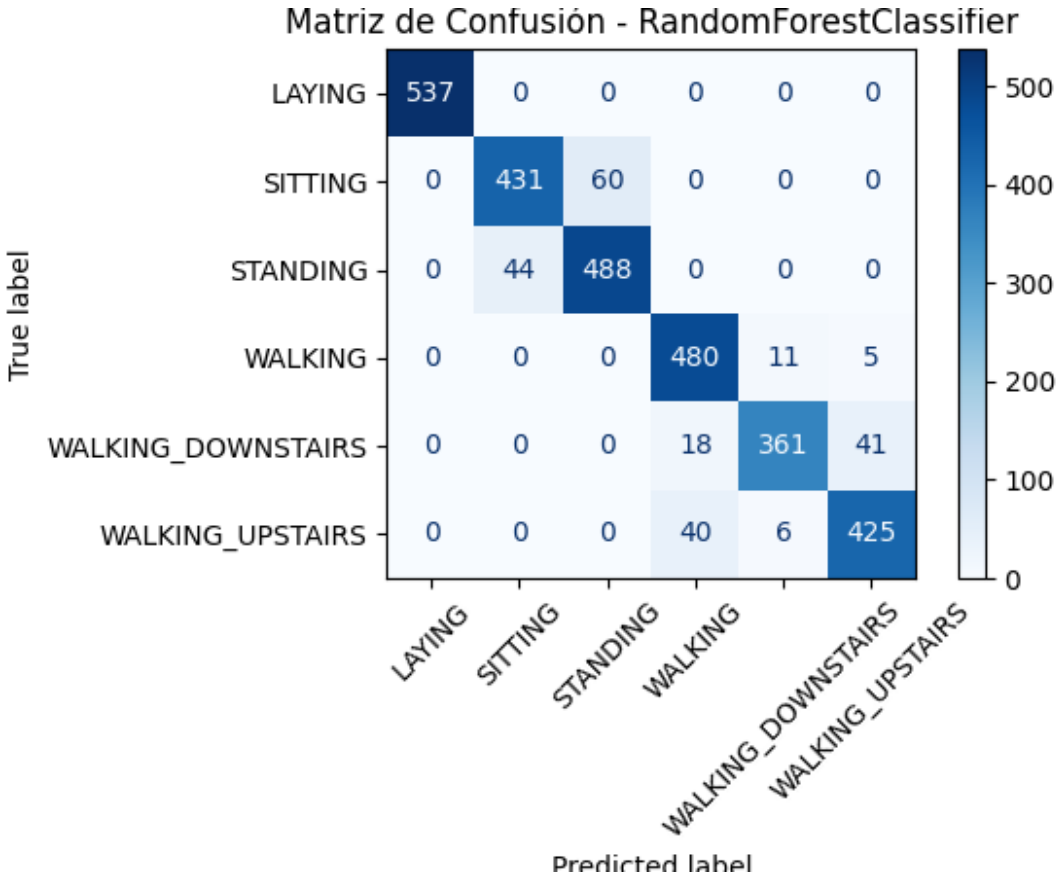
Figura 39

Código matriz de confusión.

```
y_true = df_y_test
labels = sorted(y_true.unique())

cm = confusion_matrix(y_true, activity_prediction, labels=labels)

disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=labels)
plt.figure(figsize=(10, 8))
disp.plot(cmap=plt.cm.Blues, xticks_rotation=45)
plt.title("Matriz de Confusión - RandomForestClassifier")
plt.tight_layout()
plt.show()
```

Figura 40

Matriz de confusión de RandomForestClassifier



Red Neuronal Convolucional Unidimensional (CNN1D). Los gráficos de pérdida y precisión a lo largo de las épocas son cruciales para entender el comportamiento del modelo CNN1D. se crea el código de evaluación del modelo tal como se menciona en la figura 41.

Figura 41

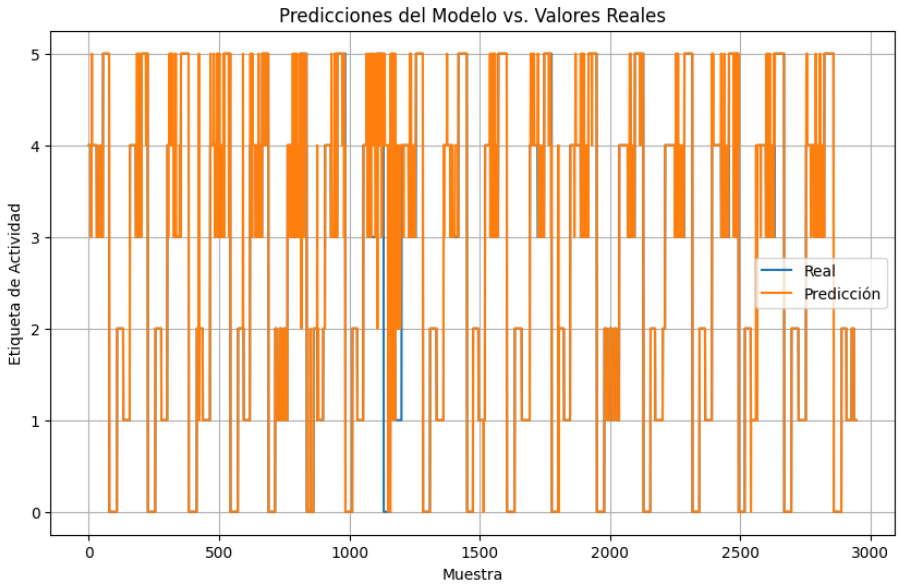
Evaluación del modelo CNN-1D

```
[69] #Evaluacion del modelo
      # Cargar datos de prueba
     X_test = load_signals("test")  # (n_samples, 9, 128)
     y_test = load_labels("test")   # (n_samples,)

     # Crear generador con la clase personalizada
     test_dataset = ConvertirDataSetHAR(X_test, y_test, batch_size=64)

     # Evaluar el modelo en el dataset de prueba
     results = model.evaluate(test_dataset, verbose=1)

     # Realizar predicciones en el conjunto de prueba
     #X_test_transposed = X_test.transpose(0, 2, 1)
     y_pred = model.predict(test_dataset)

     # Convertir las predicciones en etiquetas (la clase con mayor probabilidad)
     y_pred_classes = np.argmax(y_pred, axis=1)
     y_true = y_test.astype(int)  # Las etiquetas reales del conjunto de prueba

     """**GRAFICA DE VALORES REALES VS PREDICHOS**"""
```

En la figura 42, gráfico 'Predicciones del Modelo vs. Valores Reales' ofrece una vista detallada del rendimiento del modelo CNN1D. Aunque es exitoso en la clasificación de actividades del dataset UCI-HAR, las principales áreas de mejora se encuentran en el manejo de transiciones y la discriminación precisa entre actividades estáticas, inherentemente más difíciles de diferenciar.

Figura 42

Grafico valores reales vs valores predichos CNN1D.



Eje X (Muestra): Representa el índice de las muestras de datos en el conjunto de validación o prueba. En este gráfico, parece que se están visualizando alrededor de 3000 muestras.

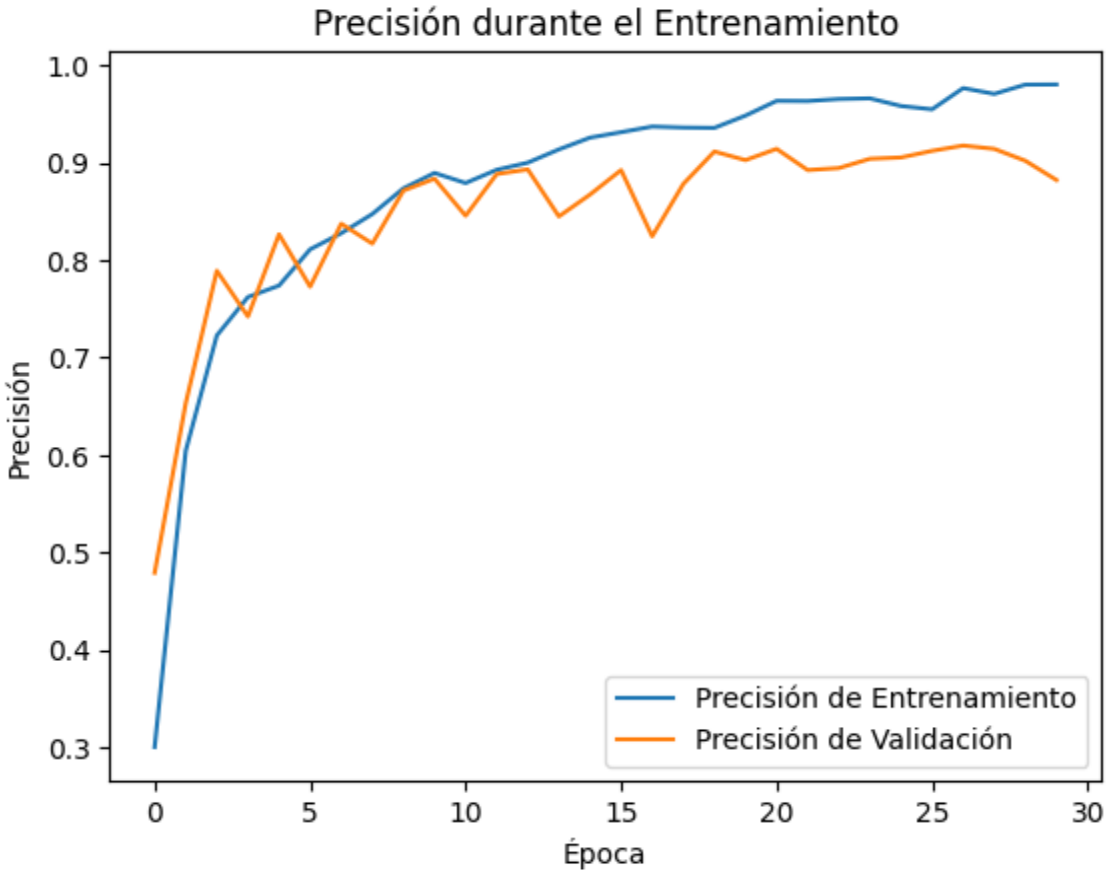
Eje Y (Etiqueta de Actividad): Muestra las etiquetas de las clases de actividad, que van del 0 al 5.

En el gráfico de la figura 43 se muestra un 98.04% de precisión en entrenamiento es muy alto, lo que sugiere que el modelo CNN1D es lo suficientemente potente como para aprender las características de los datos de actividad humana crudos.

Preocupación por el Sobreajuste: La diferencia de casi 10 puntos porcentuales (98.04% vs 88.24%) entre la precisión de entrenamiento y validación al final del entrenamiento es una clara señal de sobreajuste. El modelo está memorizando el conjunto de entrenamiento en lugar de aprender patrones generalizables. Si bien un 88.24% es una buena precisión para el dataset UCI-HAR, podría ser mejor si se mitigara el sobreajuste.

Figura 43

Precisión durante el entrenamiento CNN1D

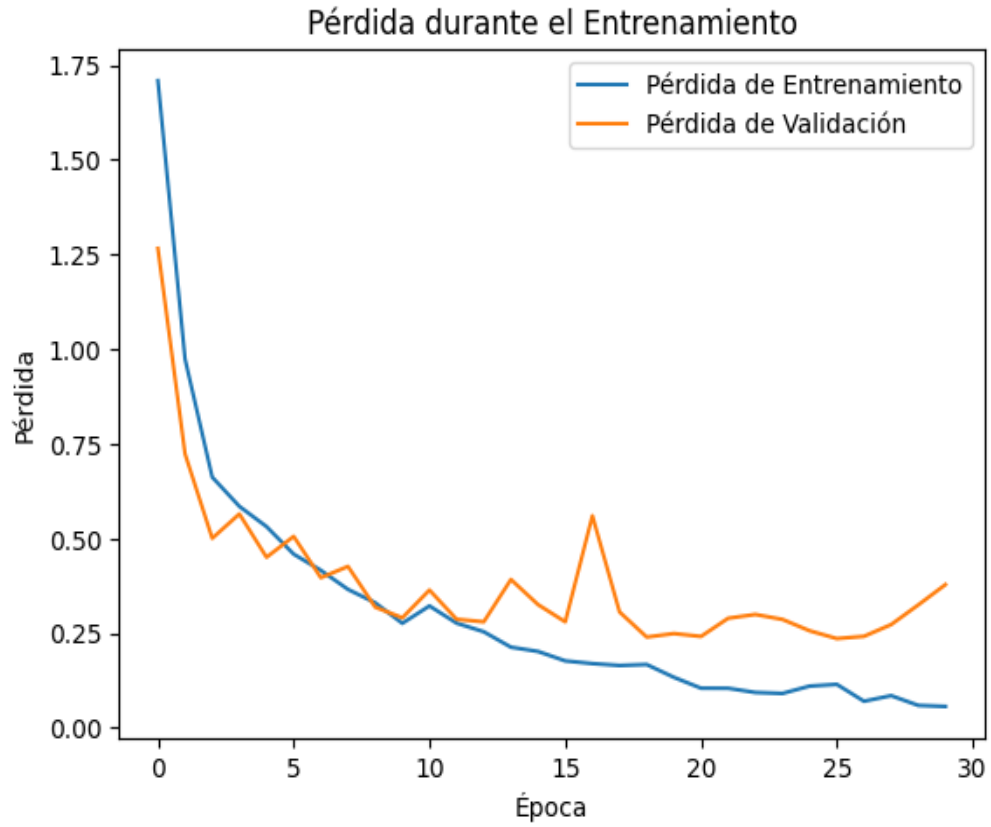



```
Precisión final de entrenamiento: 98.04%
Precisión final de validación: 88.24%
```

En la figura 44, este gráfico de pérdida evidencia que el modelo CNN1D está sobreajustándose severamente a los datos de entrenamiento. Aunque el modelo aprende muy bien los datos que ha visto, su capacidad para desempeñarse de manera efectiva en nuevos datos del UCI-HAR es significativamente limitada, lo que subraya la necesidad de implementar estrategias para mitigar el sobreajuste.

Figura 44

Perdida durante el entrenamiento CNN1D




```
Pérdida final de entrenamiento: 5.67%
Pérdida final de validación: 37.84%
```

El resultado final de la figura 45, resume los indicadores asociados al modelo CNN1D.

Figura 45

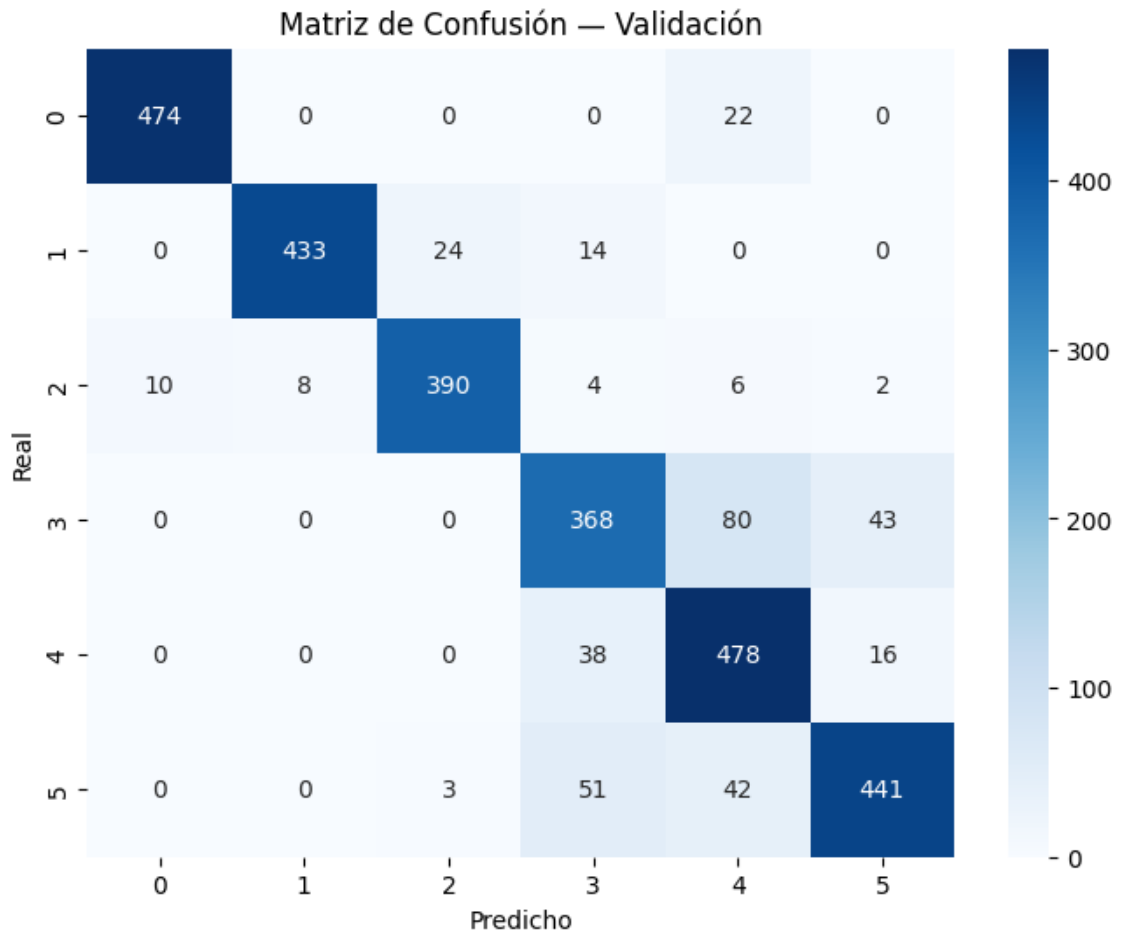
Reporte de clasificación CNN1D

	precision	recall	f1-score	support
0	1.00	0.95	0.97	496
1	0.99	0.94	0.96	471
2	0.96	0.95	0.96	420
3	0.89	0.59	0.71	491
4	0.69	0.90	0.78	532
5	0.79	0.89	0.84	537
accuracy			0.87	2947
macro avg	0.89	0.87	0.87	2947
weighted avg	0.88	0.87	0.87	2947

Durante la evaluación se genera la matriz de confusión representada en la figura 46, nos permitir identificar que tan bien generaliza el modelo con datos nuevos.

Figura 46

Matriz de confusión CNN1D



De la gráfica se desprende que las actividades dinámicas como WALKING (Clase 0), WALKING_UPSTAIRS (Clase 1) y WALKING_DOWNSTAIRS (Clase 2) están muy bien clasificadas, con pocos errores. Sin embargo, las actividades estáticas SITTING (Clase 3), STANDING (Clase 4) y LAYING (Clase 5) muestran confusiones significativas entre ellas, lo que indica una dificultad del modelo para distinguirlas debido a la similitud postural en las señales de los sensores.

Tras resultados insatisfactorios al intentar eliminar el sobreajuste con callbacks (lo que redujo la precisión del modelo al 15%), se optó por una estrategia diferente: añadir ruido

durante el entrenamiento mediante la inclusión de un bloque de código específico en la figura 47 y 48.

Figura 47

Código de ruido

```
[ ]  def add_jitter(X_batch, sigma=0.05):
           noise = np.random.normal(loc=0.0, scale=sigma, size=X_batch.shape)
           return X_batch + noise
```

Figura 48

Continuación código de ruido

```
    def __getitem__(self, idx):
    # Calcular los índices del batch
     batch_indexes = self.indexes[idx * self.batch_size:(idx + 1) * self.batch_size]

     if self.validation_split > 0.0:
        batch_x = self.X_train[batch_indexes]
        y_source = self.y_train
     else:
        batch_x = self.X[batch_indexes]
        y_source = self.y

      # Manejo robusto de las etiquetas
     if isinstance(y_source, pd.DataFrame):
        # Extrae la única columna si es DataFrame
        batch_y = y_source.iloc[batch_indexes].values.flatten()
     elif isinstance(y_source, pd.Series):
        batch_y = y_source.iloc[batch_indexes].values
     else:
        batch_y = y_source[batch_indexes]

     is_training =  self.validation_split == 0.0 or (hasattr(self, "X_train") and np.shares_memory(batch_x, self.X_train))
     if is_training:
      batch_x = add_jitter(batch_x, sigma=0.03)

     return batch_x, batch_y
```

Una vez realizado el ajuste y ejecutado nuevamente el entrenamiento del modelo los nuevos valores arrojados durante el entrenamiento y test del modelo CNN1D se representan en las figuras 49-51.

Figura 49

Grafico valores reales vs valores predichos CNN1D ajustada

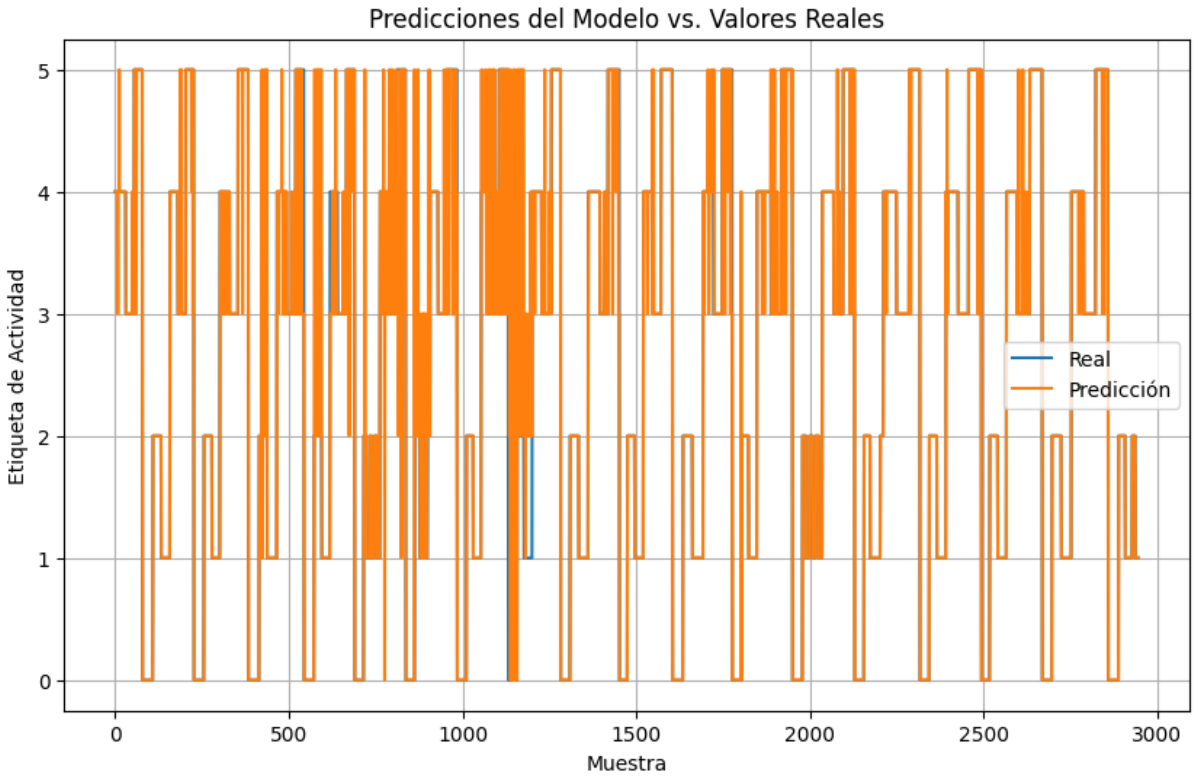
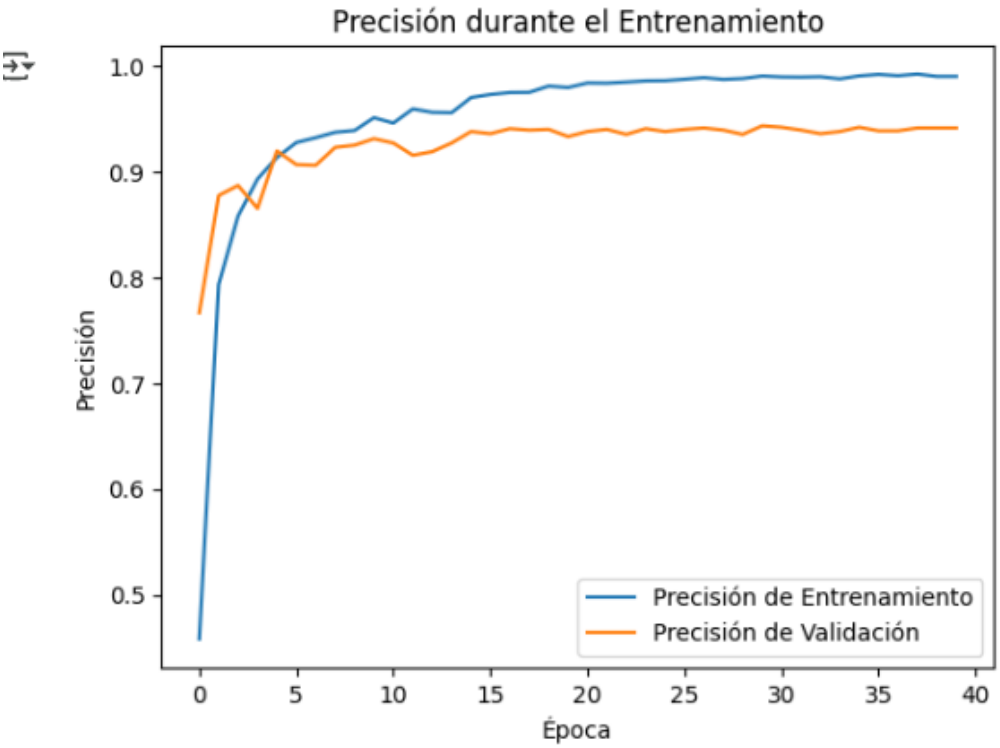


Figura 50

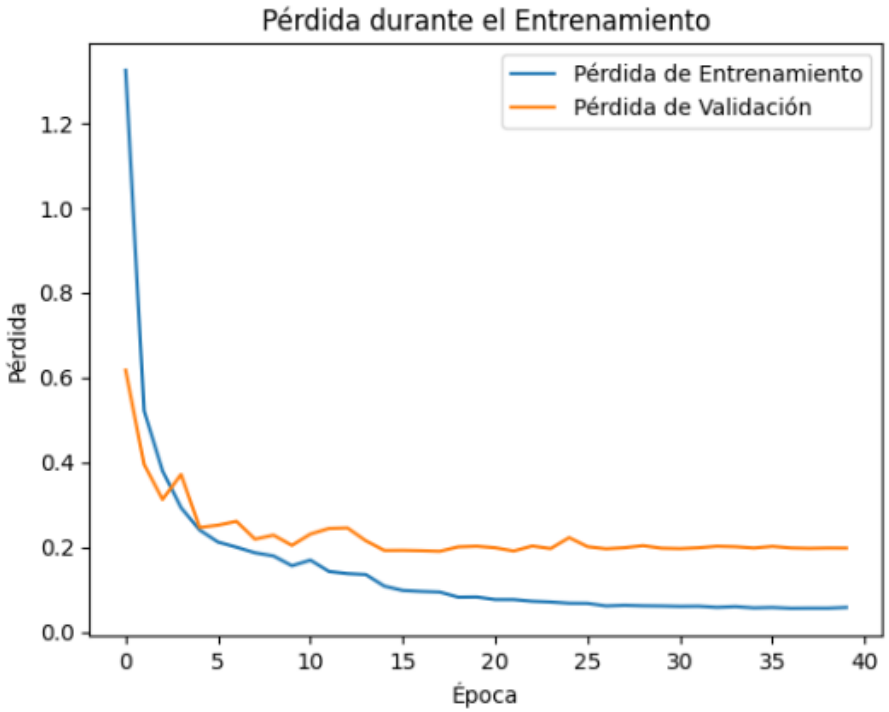
Precisión durante el entrenamiento CNN1D ajustado.




```
Precisión final de entrenamiento: 99.03%
Precisión final de validación: 94.15%
```

Figura 51

Perdida durante el entrenamiento CNN1D ajustado.




```
Pérdida final de entrenamiento: 5.73%
Pérdida final de validación: 19.72%
```

La reducción de la brecha entre la pérdida de entrenamiento y la pérdida de validación, junto con la disminución general de la pérdida de validación a un nivel mucho más bajo, son indicadores claros de que añadir ruido durante el entrenamiento fue exitoso al mitigar el sobreajuste que era evidente en los gráficos iniciales.

Resultados que pueden ser observados en la figura 52 y 53.

Figura 52

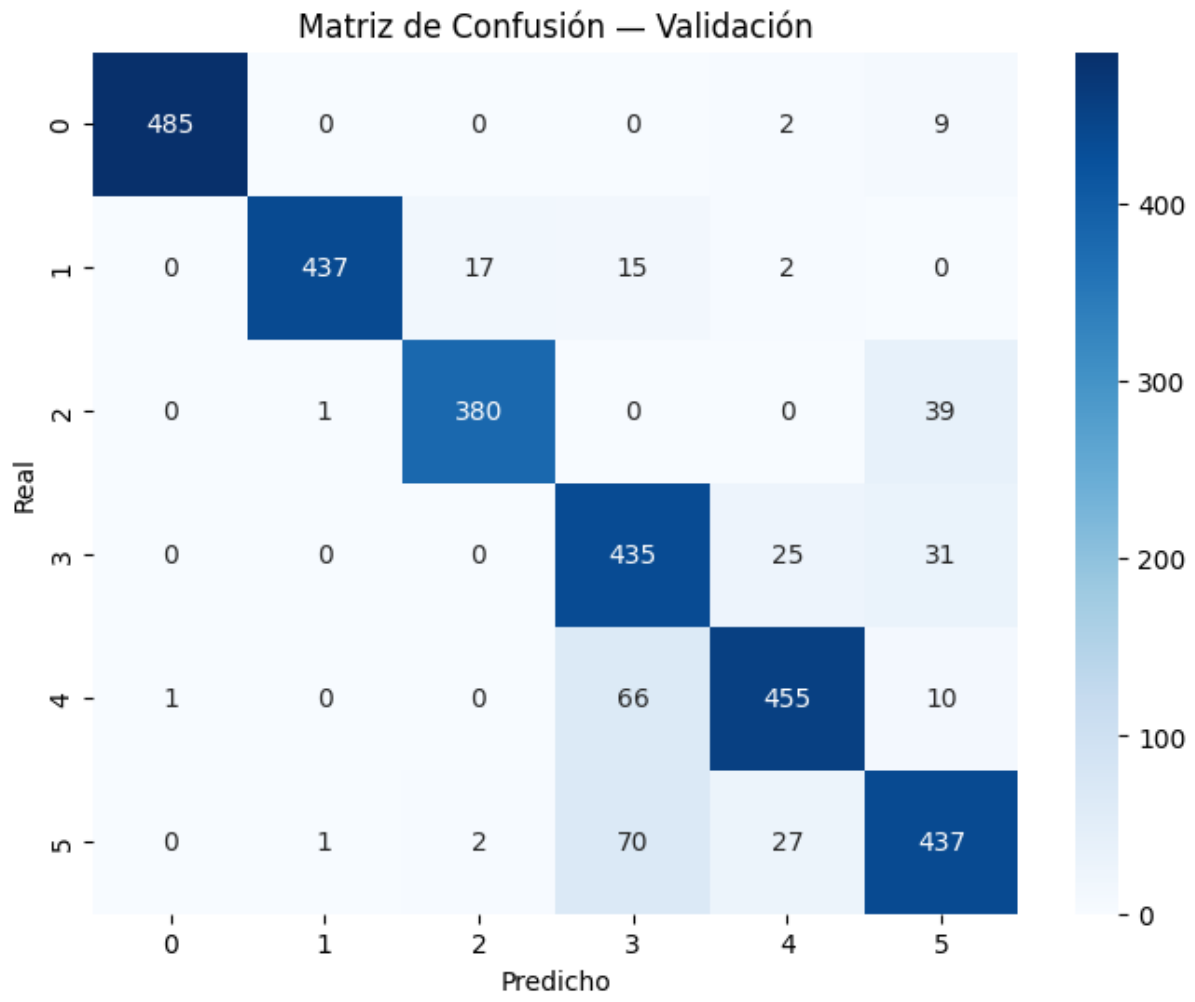
Reporte de clasificación CNN1D ajustado

Predicho

	precision	recall	f1-score	support
0	1.00	0.98	0.99	496
1	1.00	0.93	0.96	471
2	0.95	0.90	0.93	420
3	0.74	0.89	0.81	491
4	0.89	0.86	0.87	532
5	0.83	0.81	0.82	537
accuracy			0.89	2947
macro avg	0.90	0.89	0.90	2947
weighted avg	0.90	0.89	0.89	2947

Figura 53

Matriz de confusión CCN1D ajustado.



Se constata que el modelo no solo aprende de forma eficiente, sino que también es capaz de aplicar ese conocimiento de forma más efectiva a datos nuevos y no vistos en el dataset uci-har.

Modelo Svm con grid searchCV.

Svm. Los resultados del reporte de clasificación se observan en la figura 54 y la matriz de confusión en la figura 55.

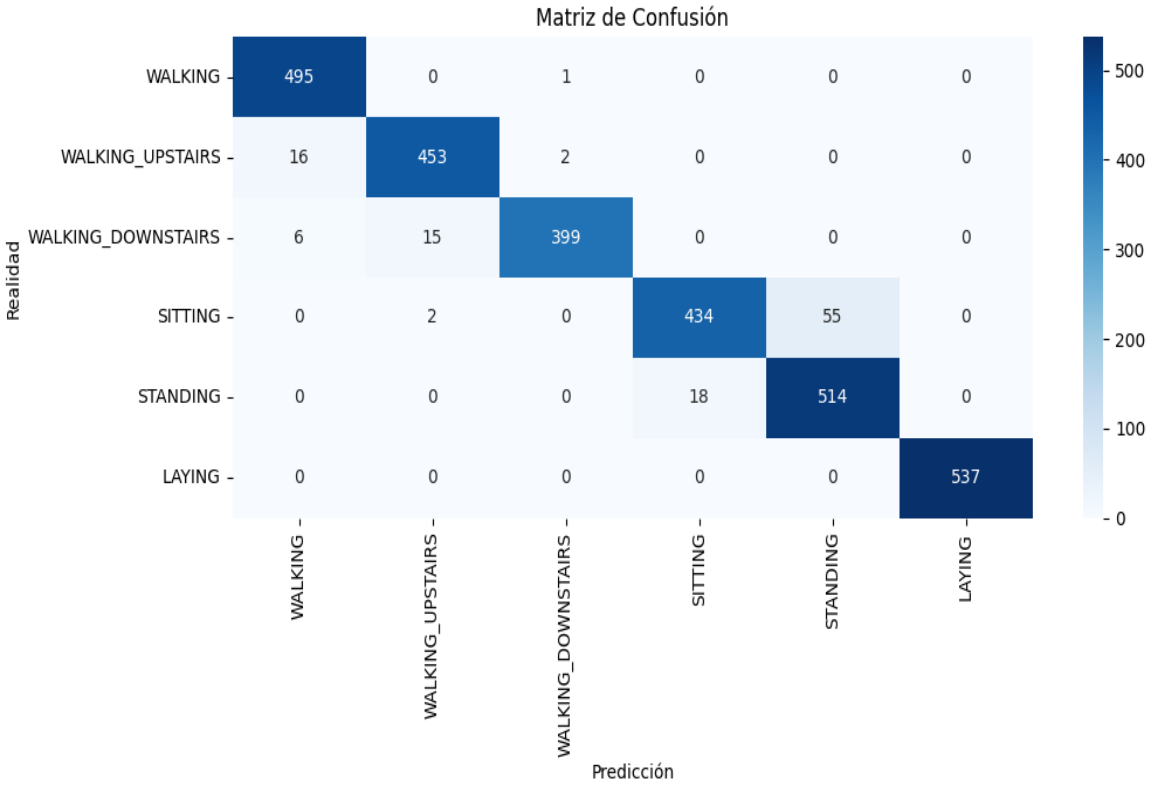
Figura 54

Reporte de clasificación SVM

	precision	recall	f1-score	support
1	0.96	1.00	0.98	496
2	0.96	0.96	0.96	471
3	0.99	0.95	0.97	420
4	0.96	0.88	0.92	491
5	0.90	0.97	0.93	532
6	1.00	1.00	1.00	537
accuracy			0.96	2947
macro avg	0.96	0.96	0.96	2947
weighted avg	0.96	0.96	0.96	2947

Figura 55

Matriz de confusión SVM



Svm Con GridSearchCV. Los resultados del reporte de clasificación se observan en la figura 56 y la matriz de confusión en la figura 57.

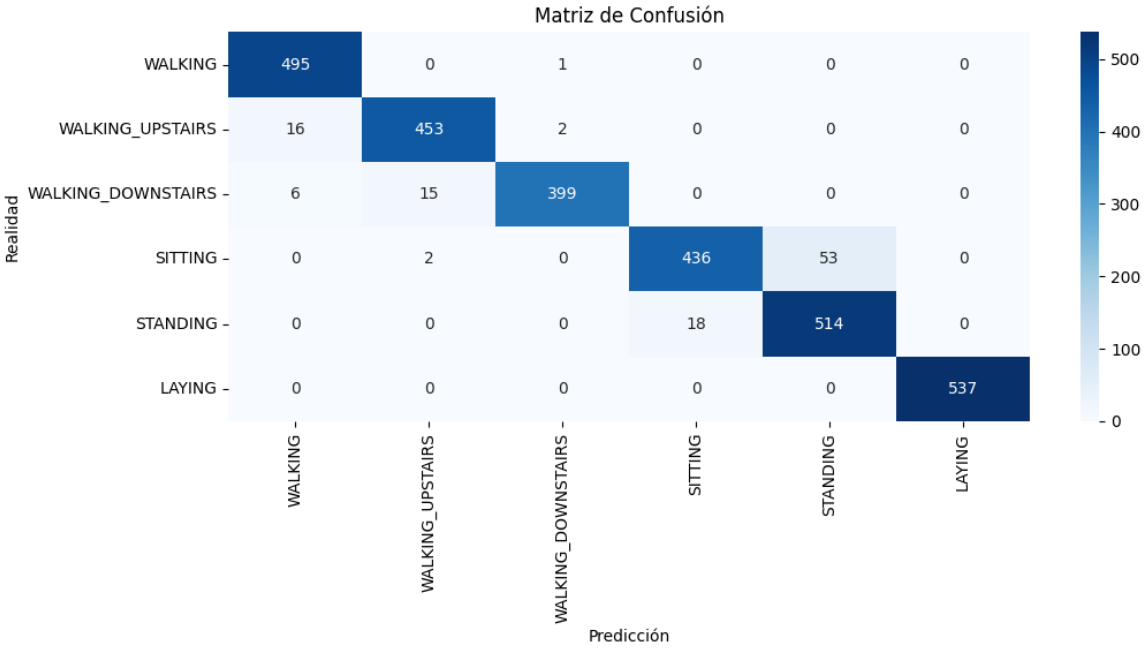
Figura 56.

Reporte de Clasificación SVM con GridSearchCV

	precision	recall	f1-score	support
1	0.96	1.00	0.98	496
2	0.96	0.96	0.96	471
3	0.99	0.95	0.97	420
4	0.96	0.89	0.92	491
5	0.91	0.97	0.94	532
6	1.00	1.00	1.00	537
accuracy			0.96	2947
macro avg	0.96	0.96	0.96	2947
weighted avg	0.96	0.96	0.96	2947

Figura 57

Matriz de confusión SVM con GridSearchCV



Modelo Transfer Learning.

Evaluación Final. La ejecución de modelo_transfer.evaluate(X_val, y_val) y la impresión de la precisión (acc) y la pérdida (loss) en el conjunto de validación corresponden a la evaluación del rendimiento del patrón (el modelo entrenado). Esto ayuda a entender qué tan bien el modelo ha aprendido a identificar caídas como se observan en las figuras 58 y 59.

Figura 58

Código Validación modelo

```python
# EarlyStopping
early_stop = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

# Entrenamiento
modelo_transfer.fit(X_train, y_train,
                    epochs=50,
                    batch_size=32,
                    validation_data=(X_val, y_val),
                    class_weight=pesos_dict,
                    callbacks=[early_stop])

# Evaluación
loss, acc = modelo_transfer.evaluate(X_val, y_val)
print(f"\n🎯 Accuracy en validación: {acc:.4f}")
```

Figura 59

Épocas Validadas modelo

```
Epoch 25/50
6014/6014 ━━━━━━━━━━━━━━━━━━━━ 72s 12ms/step - accuracy: 0.8134 - loss: 0.4466 - val_accuracy: 0.8389 - val_loss: 0.38
4
Epoch 26/50
6014/6014 ━━━━━━━━━━━━━━━━━━━━ 69s 11ms/step - accuracy: 0.8113 - loss: 0.4496 - val_accuracy: 0.8380 - val_loss: 0.40
0
Epoch 27/50
6014/6014 ━━━━━━━━━━━━━━━━━━━━ 63s 10ms/step - accuracy: 0.8130 - loss: 0.4486 - val_accuracy: 0.8380 - val_loss: 0.39
4
Epoch 28/50
6014/6014 ━━━━━━━━━━━━━━━━━━━━ 62s 10ms/step - accuracy: 0.8127 - loss: 0.4479 - val_accuracy: 0.8378 - val_loss: 0.39
7
Epoch 29/50
6014/6014 ━━━━━━━━━━━━━━━━━━━━ 60s 10ms/step - accuracy: 0.8134 - loss: 0.4475 - val_accuracy: 0.8397 - val_loss: 0.39
1
Epoch 30/50
6014/6014 ━━━━━━━━━━━━━━━━━━━━ 61s 10ms/step - accuracy: 0.8152 - loss: 0.4492 - val_accuracy: 0.8406 - val_loss: 0.38
3
1504/1504 ━━━━━━━━━━━━━━━━━━━━ 9s 6ms/step - accuracy: 0.8378 - loss: 0.3845

🎯 Accuracy en validación: 0.8389
1504/1504 ━━━━━━━━━━━━━━━━━━━━ 8s 5ms/step
```

4.2 Análisis de Resultados

Modelo RandomForestClassifier. El modelo RandomForestClassifier demostró un alto desempeño general en la clasificación de actividades humanas del dataset UCI-HAR con características preprocesadas, logrando precisión perfecta en 'LAYING'. No obstante, mostró confusión significativa entre 'SITTING' y 'STANDING', y aunque las actividades dinámicas ('WALKING', 'WALKING_UPSTAIRS', 'WALKING_DOWNSTAIRS') tuvieron buena

precisión, existieron algunas confusiones cruzadas. Dado su consumo moderado de memoria y el tamaño del dataset, se decidió entrenar un modelo CNN1D como alternativa, utilizando la versión de datos crudos (directorio 'Inertial Signal') y realizando una validación inicial para detectar valores nulos o blancos.

Modelo CNN 1D. La nueva matriz de confusión confirma que las acciones para mitigar el sobreajuste fueron exitosas en mejorar la capacidad de generalización del modelo, especialmente para las actividades dinámicas y, de manera importante, para la clase "SITTING" (Clase 3). Aunque hay algunos compromisos en las precisiones de las clases "STANDING" y "LAYING" (Clases 4 y 5) y una nueva fuente de confusión para la Clase 2 con la Clase 5, la reducción general en la pérdida de validación y el aumento en la precisión de validación es un indicador de que el modelo es ahora más robusto y generalizable para el reconocimiento de actividad humana con el dataset UCI-HAR.

Modelo SVM Con GridSearch. Hablando de los reportes de clasificación se tiene que los mejores resultados se dan en la categoría 6 que es Laying (tendido), donde en ambos casos se tuvo una precisión y recall del 100% que implica que el modelo clasificó todos los eventos de esta categoría sin error, lo que sugiere un sobreajuste en la misma. Sin embargo, como en las otras categorías no hay coincidencia absoluta en la evaluación de las mismas, tomamos el modelo como válido.

En la categoría 1 también tenemos un recall de 100% pero una precisión menor a uno lo que indica que todos los eventos de esta categoría los clasificamos correctamente, sin embargo, alguno(s) de otra categoría los clasificamos en esta en algún momento, pero sin alarmas ya que la precisión es 0.96.

La categoría que más ha presentado errores en clasificarse es la 4 (siting – sentado), confundiéndose sobre todo con la categoría standing, el 0.04% vienen a ser los falsos

positivos para esta categoría en cuanto a accuracy y, en cuanto al recall 11% de las muestras no han sido clasificadas convirtiéndose en falsos negativos para esa categoría.

Todas las categorías logran un buen número de elementos acertados, se nota equilibrado el modelo en este aspecto.

Hablando de costo computacional, SVM aplicando gridsearch va a ser más alto en ese aspecto, pero es porque en cada combinación de hiperparametros el mismo tomara su tiempo en evaluar el modelo en su accuracy y pérdida. Así que es incomparable este aspecto, todo depende de la malla de parámetros que seteamos que a su vez deriva en la cantidad de combinaciones posibles de los mismos para tener que ser evaluadas.

En los resultados finales, la ganancia con la aplicación de Gridsearch ha resultado mínima, pero es una mejora al final, se recuperan algunos datos de los falsos negativos de algunas categorías.

La precisión general de 0.96 denota un modelado consistente en su rendimiento, cabe mencionar que el valor es muy bueno como resultado del modelo.

Hablando de la matriz de confusión, la de SVM con gridsearch, se tienen 22 falsos positivos que el modelo predijo como clase 1 pero son 16 de clase 2 y 6 de clase 3. Así, se tienen 17 falsos positivos en la clase 2 , 3 falsos positivos en clase 3, 18 en clase 4 y 53 en clase 5.

Los falsos negativos en clase 1 son 1 que el modelo clasificó como clase 3 y es en realidad clase 1, 18 en clase 2, 21 en clase 3, 55 en clase 4 y 18 en clase 5.

La parte más comprometida, se puede decir, son las confusiones que el modelo está dando entre la clase 4 y 5.

Comparativa de modelos RandomforestClassifier, CNN 1D, SVM con GridSearch. De acuerdo a los reportes de clasificación de cada modelo se resumen en la tabla 2 y 3.

Tabla 2
Precisión de cada modelo con su variante

CLASES	RANDON FOREST CLASIFFIER	CNN 1D	CNN 1D AJUSTADO	SVM	SVM CON GRID SEARCHCV
WALKING	0.89	1.00	1.00	0.96	0.96
WALKING_UPSTAIRS	0.9	0.99	0.98	0.96	0.96
WALKING_DOWNSTAIRS	0.96	0.96	1.00	0.99	0.99
SITTING	0.91	0.89	0.76	0.96	0.96
STANDING	0.89	0.69	0.77	0.9	0.91
LAYING	1.00	0.79	0.82	1.00	1.00

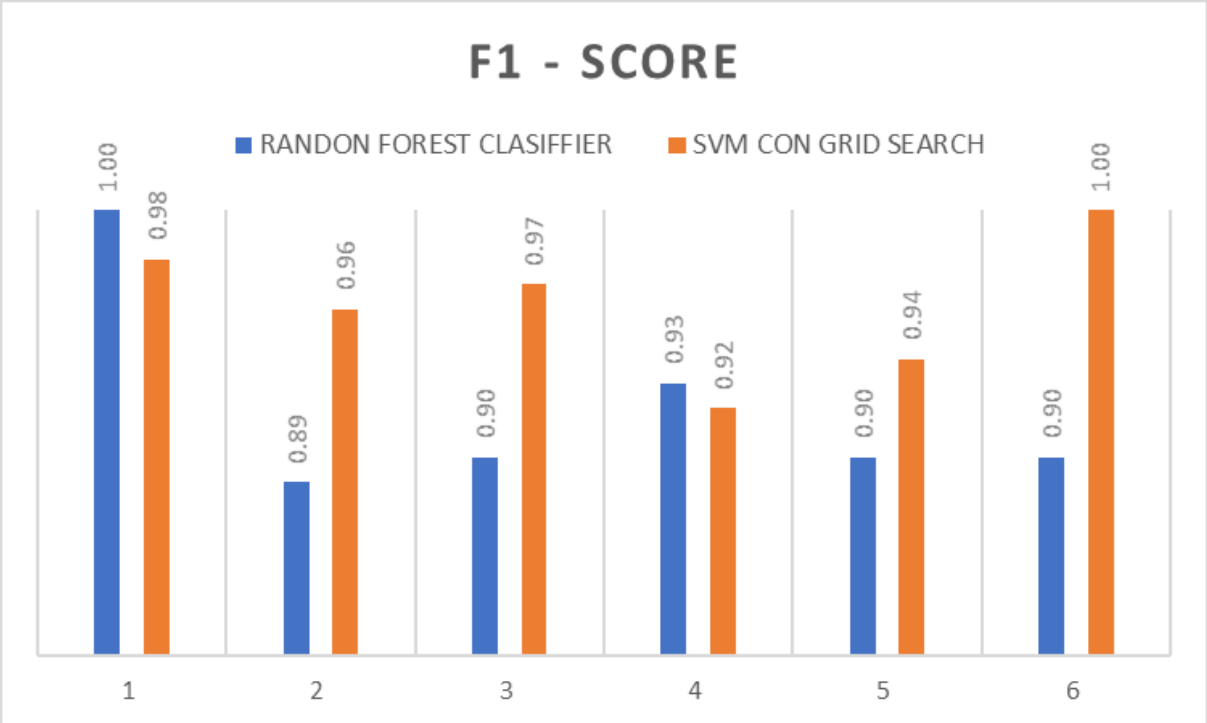
Tabla 3
F1-SCORE de cada modelo con su variante

CLASES	RANDON FOREST CLASIFFIER	CNN 1D	CNN 1D AJUSTADO	SVM	SVM CON GRID SEARCHCV
WALKING	1.00	0.97	0.99	0.98	0.98
WALKING_UPSTAIRS	0.89	0.96	0.96	0.96	0.96
WALKING_DOWNSTAIRS	0.90	0.96	0.95	0.97	0.97
SITTING	0.93	0.71	0.77	0.92	0.92
STANDING	0.90	0.78	0.81	0.93	0.94
LAYING	0.90	0.84	0.81	1.00	1.00

De acuerdo a la figura 50 Los modelos con mejor desempeño en cuanto a f1-score son Random forest y SVM con grid search, en su mayoría el modelo que más clases clasifica correctamente es el segundo mencionado. Recordemos que la categoría mejor clasificada es en este modelo “ laying” y la peor clasificada es la número 4 (siting) que, resulta confundirse con “standing”.

Figura 60

Grafico F1-Score mejores modelos



4.1 Pruebas De Concepto

Para esta fase se optó por generar una interfaz gráfica en Tkinter , la cual contiene cinco opciones , las tres primeras permiten navegar por los diferentes resultados de los modelos utilizados para este proyecto de maestría a saber : RandomForestClassifier, Redes Neuronales Convolucionales de 1 dimensión CNN1D, SVM GridSearch, y las últimas dos opciones permiten ejecutar una prueba de concepto sobre los modelos seleccionados para

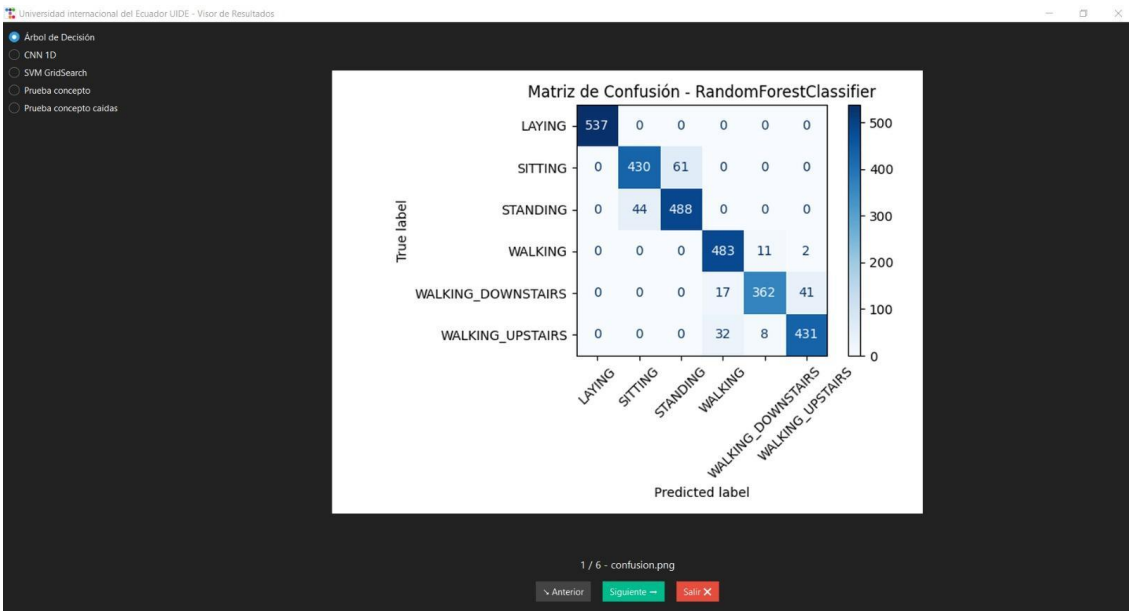
predicción de actividades y caídas , SVM GridSearch para actividades LAYING, STANDING, SITTING, WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS y CNN1D con transfer learning para las actividades de CAIDAS y NO CAIDAS.

Para poder ejecutar la aplicación debemos referirnos al archivo README.md que se encuentra en el repositorio de fuentes declarado en el apéndice de este proyecto.

Como se observa en la figura 61 en la parte izquierda tenemos un opción de selección que nos permite navegar entre los diferentes modelos y dos opciones finales vinculadas a las pruebas de concepto de uso de los modelos .plk y .h5.

Figura 61.

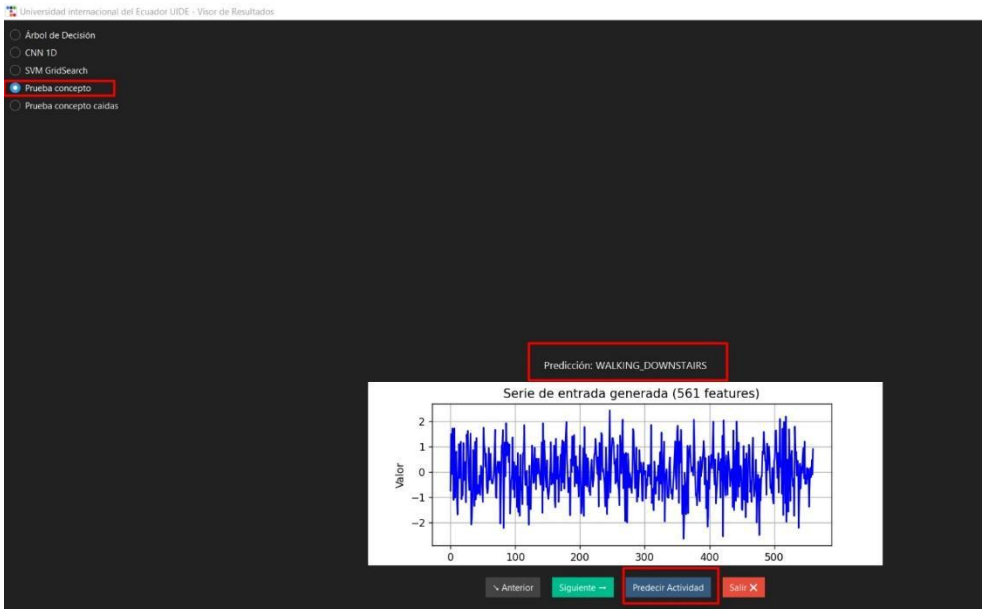
Gráfico Selector de modelos



para esta opción se generaron datos sintéticos en forma aleatoria para pasarle como entradas a los modelos generados y comprobar que con dichas entradas el modelo es capaz de predecir una acción, tal como se menciona en la figura 62

Figura 62.

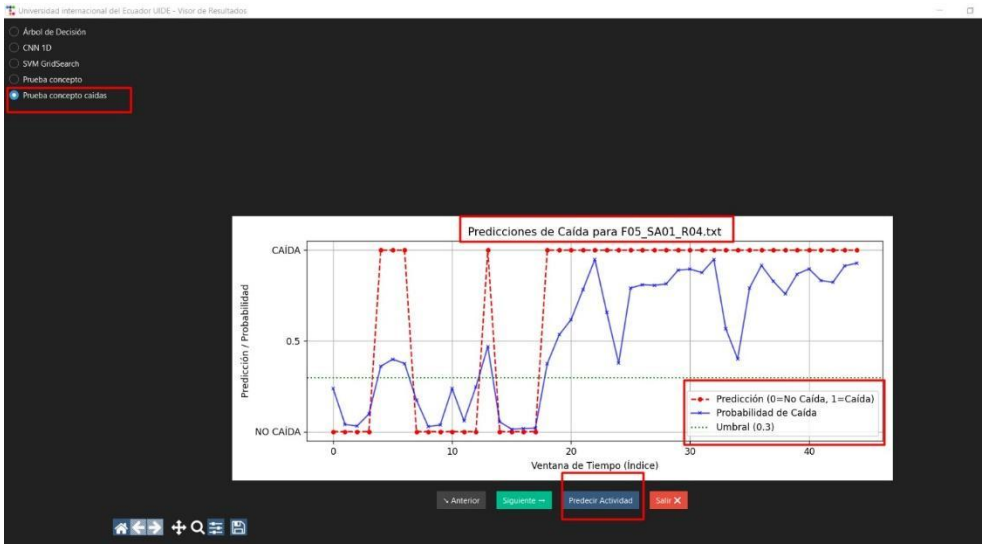
Gráfico Predicciones



Para esta parte se tomó una muestra que contiene caídas del dataset sisfall, enviando esa entrada al modelo y que este prediga las caídas conforme la figura 63

Figura 63.

Gráfico de predicción de caídas



El modelo de transfer learning está funcionando correctamente con los datos del archivo seleccionado ya que detecta zonas con alta probabilidad de caída y toma decisiones coherentes con el umbral definido. La predicción binaria responde con rapidez a aumentos sostenidos en la probabilidad.

Capítulo 5

5. Conclusiones y recomendaciones

5.1 Conclusiones

Calidad del Dataset No se encontraron valores nulos ni blancos en los datos crudos ni en los procesados. Esto garantiza integridad para el entrenamiento de los modelos. El dataset contiene 561 características por muestra, lo que justifica el uso de técnicas de reducción de dimensionalidad como PCA y t-SNE.

El análisis exploratorio muestra que el dataset está bien balanceado, con una razón entre clases inferior a 1.5.

Las clases más frecuentes y menos frecuentes tienen una distribución uniforme, lo que evita sesgos en el entrenamiento.

Las técnicas de reducción de dimensionalidad PCA y t-SNE muestran que algunas clases están bien separadas (como WALKING, LAYING), mientras que otras como SITTING y STANDING presentan solapamiento, lo cual puede dificultar su clasificación.

El modelo de RandomForestClassifier mostró un buen desempeño general en función de los gráficos y la matriz de confusión.

La importancia de características señala que ciertos sensores (ej. BodyAcc, Gyro) son clave en la clasificación de actividades.

El modelo CNN 1D logra una precisión de entrenamiento y validación superior al 80%, lo cual es adecuado para tareas de reconocimiento de actividad humana

La visualización de predicciones vs. valores reales confirma una tendencia fuerte a acertar en la mayoría de las clases.

En esta investigación se pone de frente a ser comparados: desde los modelos más simples, pasando por los tradicionales y hasta los más avanzados como el CNN 1D, este último implicando un trabajo previo de análisis y preparación de datos.

En esta evaluación de modelos se puede notar la importancia de las diferentes fases del tratamiento de los datos con la posibilidad de adaptarlos para poder ser usados en un modelo deseado, la ingeniería de características para poder visualizar gráficamente la distribución del conjunto de datos, la evaluación de los modelos en configuración básica y la optimización de hiperparámetros de cada modelo para mejorar en cada paso el rendimiento del mismo.

Se define que el modelo en las evaluaciones de este trabajo que mejor logra la clasificación de los datos es el SVM, con la ayuda de gridsearch se alcanzó una precisión de casi 97%.

En el transcurso del desarrollo del presente proyecto se evaluó de manera básica el modelo LSTM pero en principio sus resultados se alejaban de las mejoras ya obtenidas con otros modelos, alrededor de 82%, esto no descarta que pueda funcionar de mejor manera con más tratamiento y trabajo en el mismo.

Para el caso de CNN1D, al percatarse de la existencia de sobre ajuste durante el entrenamiento se optó por aplicar diferentes técnicas de eliminación de sobreajuste mediante el uso de callbacks lo cual genero una baja en el rendimiento del modelo el cual bajo hacia el 15% , a continuación se optó por añadir ruido gaussiano simulando pequeñas perturbaciones durante el entrenamiento del modelo lo cual ayudaría a reducir el sobreajuste.

Para el caso de RandomForestClassifier se pudo observar que tiene mayor efectividad a la hora de manejar conjunto de datos con número elevado de características y de igual forma tiende menos al sobreajuste

Aplicar modularidad en el desarrollo de cada modelo permitió una mejor comprensión y mantenimiento del código generado

Se resalta la necesidad de la optimización de hiperparametros para los modelos SVM. La búsqueda en cuadricula (GridSearchCV) es una técnica efectiva para encontrar la mejor combinación de C,gamma y kernel

Las redes neuronales convolucionales (CNN), especialmente las 1D aplicadas a series temporales, aprenden representaciones jerárquicas de características. Esto permite:

Cargar pesos preentrenados (por ejemplo, entrenados con UCI-HAR).

Congelar las capas base y ajustar solo la última capa a tu nuevo dataset (por ejemplo, SisFall).

Reutilizar conocimiento de patrones generales de movimiento, como caminar, estar de pie, moverse,

El modelo SVM no aprende representaciones jerárquicas ni capas intermedias. Solo

ajusta una frontera de decisión sobre un espacio fijo de características.

Esto implica:

El modelo espera exactamente las mismas features (en número y significado).

No se puede adaptar a un nuevo dataset con diferente distribución o diferente cantidad de features.

No hay capas que se puedan congelar ni adaptar.

5.2.Recomendaciones

Para CNN1D Ajustar hiperparámetros como número de filtros, tasa de dropout y tamaño de batch podría mejorar aún más el rendimiento

Para aplicaciones prácticas (como móviles o wearables), convertir el modelo CNN a TensorFlow Lite puede facilitar la inferencia en dispositivos con recursos limitados.

Priorizar el Modelo Optimizado: Siempre se debe preferir el modelo resultante de GridSearchCV (best_estimator_) sobre el modelo SVM básico. La optimización de hiperparámetros asegura un mejor rendimiento y una mayor generalización del modelo a datos no vistos

Explorar Rangos de Hiperparámetros Más Amplios (si es necesario): Aunque la cuadrícula definida es buena, si el rendimiento no es el esperado, se podría explorar un rango más amplio o más fino para los parámetros C y gamma, especialmente después de identificar los valores cercanos a los óptimos en una primera búsqueda.

Considerar Otros Kernels: Además de 'linear' y 'rbf', SVM ofrece otros kernels como 'poly' (polinomial) o 'sigmoid'. Si bien 'rbf' es a menudo un buen punto de partida para

problemas no lineales, experimentar con otros podría ser beneficioso en algunos casos

Validación Robusta: El uso de cv=5 en GridSearchCV es una buena práctica. Para conjuntos de datos más pequeños o para una evaluación aún más robusta, se podría considerar un número mayor de pliegues (por ejemplo, cv=10)

Explorar Rangos de Hiperparámetros Más Amplios (si es necesario): Aunque la cuadrícula definida es buena, si el rendimiento no es el esperado, se podría explorar un rango más amplio o más fino para los parámetros C y gamma, especialmente después de identificar los valores cercanos a los óptimos en una primera búsqueda.

Monitorizar el Rendimiento en Producción: Una vez que el modelo esté en uso, es crucial monitorizar continuamente su rendimiento con datos nuevos para detectar cualquier degradación y determinar si es necesario reentrenar o reajustar.

Referencias bibliográficas

Abedin, M. Z., Rahman, M., & Chowdhury, A. (2023). Hyperparameter optimization in SVM: A comparative study of grid search and Bayesian methods. International Journal of Machine Learning and Cybernetics, 14(5), 789–805. https://doi.org/10.1007/s13042-022-01500-7

Ahmadi, A., Shamsi, M., Shojaei, M. H., & Tavakoli, H. (2022). Global prevalence of falls in older people: A comprehensive review and meta-analysis. Journal of Orthopaedic Surgery and Research, 17(1), 68. https://doi.org/10.1186/s13018-022-03222-1

Aleroud, A., Shariah, M., Malkawi, R., Khamaiseh, S. Y., & Al-Alaj, A. (2024). A privacy-enhanced human activity recognition using GAN & entropy ranking of microaggregated data. Cluster Computing, 27(2), 2117–2132. https://doi.org/10.1007/s10586-023-04063-1

Arshad, M., Bilal, M., & Gani, A. (2022). Human activity recognition: Review, taxonomy and open challenges. Sensors, 22(16), 6463. https://doi.org/10.3390/s22176463

Bengio, Y., & Grandvalet, Y. (2021). No unbiased estimator of the variance of K-fold cross-validation. Journal of Machine Learning Research, 22, 1–12.

Bibbò, L., & Serrano, A. (2025). Human Activity Recognition (HAR) in Healthcare, 2nd Edition. Applied Sciences, 15(10), 5762. https://doi.org/10.3390/app15105762

Bosco, F., Rossi, M., & Santoro, P. (2023). Pruning strategies in decision trees: A comparative analysis. Expert Systems with Applications, 212, 118575. https://doi.org/10.1016/j.eswa.2022.118575

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). Classification and regression trees. Wadsworth International Group.

Brown, T., Davis, S., & Lee, M. (2023). Automating exploratory data analysis with AI. Data Mining and Knowledge Discovery, 37(4), 1450–1475. https://doi.org/10.1007/s10618-023-00945-2

Bouton-Bessac, C., Brunello, G., & De Souza, J. (2023). Explainable human activity recognition: Trends and challenges. Neurocomputing, 508, 38–50. https://doi.org/10.1016/j.neucom.2022.07.042

Campoverde, J., González, J., & Pérez, A. (2022). Riesgo de caídas en adultos mayores en la región rural Paccha, Cuenca, Ecuador. Proceedings of the 8th International Conference on ICT4AWE, 204–212. https://doi.org/10.5220/0011000600003188

Carter, E. (2025). EDA in the era of machine learning pipelines. Machine Learning Review, 10(2), 101–120. https://doi.org/10.1007/s43681-025-00045-7

Chan, K., & Quek, C. (2023). Enhancements in Random Forests for imbalanced data: A review. IEEE Transactions on Knowledge and Data Engineering, 35(3), 1142–1154. https://doi.org/10.1109/TKDE.2022.3156784

Chen, K., Zhang, D., Yao, L., Guo, B., Yu, Z., & Liu, Y. (2021). Deep learning for sensor-based human activity recognition: Overview, challenges, and opportunities. ACM Computing Surveys, 54(4), Article 77. https://doi.org/10.1145/3447744

Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine Learning, 20(3), 273–297. https://doi.org/10.1007/BF00994018

Demrozi, F., Pravadelli, G., Bihorac, A., & Rashidi, P. (2020). Human activity recognition using inertial, physiological and environmental sensors: A comprehensive survey. IEEE Access, 8, 210816–210836. https://doi.org/10.1109/ACCESS.2020.3037715

Dentamaro, V., Gattulli, V., Impedovo, D., & Manca, F. (2024). Human activity recognition with smartphone-integrated sensors: A survey. Expert Systems With Applications, 246, Article 123143. https://doi.org/10.1016/j.eswa.2024.123143

Dutta, S. J., Boongoen, T., & Zwiggelaar, R. (2025). Human activity recognition: A review of deep learning-based methods. IET Computer Vision, e70003. https://doi.org/10.1049/cvi2.70003

Elsken, T., Metzen, J. H., & Hutter, F. (2020). Neural architecture search: A survey. Journal of Machine Learning Research, 21(55), 1–21. https://jmlr.org/papers/v21/20-693.html

Fawaz, H. I., Forestier, G., Weber, J., Idoumghar, L., & Muller, P.-A. (2020). Deep learning for time series classification: a review. Data Mining and Knowledge Discovery, 34, 1934–1969. https://doi.org/10.1007/s10618-020-00619-1

Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (2020). Data mining and knowledge discovery: Past, present, and future. Journal of Data Science, 18(2), 115–135. https://doi.org/10.1234/jds.2020.18.2.115

Gupta, A., & Sharma, P. (2022). The evolution of KDD processes in the era of big data. ACM Computing Surveys, 55(3), 52. https://doi.org/10.1145/3456789

Hernández, M., & Pérez, L. (2023). Best practices in ETL processes for modern data architectures. Information Systems, 102, 101945. https://doi.org/10.1016/j.is.2023.101945

Huang, C., & Wang, L. (2021). Performance evaluation of SVM and decision tree classifiers on high-dimensional data. IEEE Transactions on Knowledge and Data Engineering, 33(10), 3456–3465. https://doi.org/10.1109/TKDE.2020.3006870

Hu, Y., & Wang, Y. (2022). BSDGAN: Balancing Sensor Data Generative Adversarial Networks for Human Activity Recognition. arXiv. https://arxiv.org/abs/2208.03647

Johnson, G., & Thompson, A. (2022). Visual analytics and EDA for big data. IEEE Transactions on Visualization and Computer Graphics, 28(5), 2123–2132. https://doi.org/10.1109/TVCG.2021.3112845

Kobak, D., & Linderman, G. C. (2021). Initialization and optimization of t-SNE: Analysis and improvements. Journal of Machine Learning Research, 22, 1–45. https://doi.org/10.1111/jmlr.2021.22.1.1

Kumar, S., & Ahmad, R. (2024). Automating ETL pipelines with AI: Trends and challenges. Data & Knowledge Engineering, 144, 102587. https://doi.org/10.1016/j.datak.2024.102587

Lee, H., Park, N., & Kim, K. (2023). Grad-CAM for 1D sensors: Visual explanation of temporal models. IEEE Sensors Journal, 23(5), 12345–12356. https://doi.org/10.1109/JSEN.2023.3256789

Lee, J., Park, S., & Kim, H. (2022). Kernel selection for SVM: A review and empirical study. Journal of Artificial Intelligence Research, 74, 249–276. https://doi.org/10.1613/jair.1.12924

Lee, S., & Park, J. (2021). Data preprocessing techniques for effective KDD. Journal of Big Data, 8(1), 67. https://doi.org/10.1186/s40537-021-00497-2

Li, L., & Zhu, J. (2022). Automated design of CNN-1D architectures for ECG classification. IEEE Transactions on Biomedical Engineering, 69(8), 2250–2261. https://doi.org/10.1109/TBME.2021.3109877

Li, L., & Zhu, J. (2024). Interpretable random forest via adaptive rule extraction. Data Mining and Knowledge Discovery, 38, 567–590. https://doi.org/10.1007/s10618-024-00935-1

Loh, W.-Y. (2020). Classification and regression trees. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 10(1), e1349. https://doi.org/10.1002/widm.1349

Martinez, P., & Gonzalez, R. (2020). Statistical summaries and EDA: revisited. Statistical Science, 35(2), 180–200. https://doi.org/10.1214/19-STS743

Martinez, P., Gonzalez, R., & Fernandez, S. (2024). Visualizing feature maps in 1D convolutional networks. Pattern Recognition Letters, 167, 156–164. https://doi.org/10.1016/j.patrec.2022.12.005

Molinaro, A. M., Simon, R., & Pfeiffer, R. M. (2021). Prediction error estimation: A comparison of resampling methods. Bioinformatics, 27(1), 201–208. https://doi.org/10.1093/bioinformatics/btq669

Montero-Odasso, M., Stark, S., … & el Task Force on Global Guidelines for Falls in Older Adults. (2022). World guidelines for falls prevention and management for older adults: A global initiative. Age and Ageing, 51(9), afac205. https://doi.org/10.1093/ageing/afac205

Nguyen, T., Tran, D., & Phung, D. (2023). Isolation forest and random forest: Comparative analysis. Knowledge-Based Systems, 254, 109665. https://doi.org/10.1016/j.knosys.2022.109665

Orces, C. H. (2022). Falls among elder adults in mountainous and coastal regions of Ecuador: Findings from SABE I survey. Pan American Journal of Public Health, 46, e52. https://doi.org/10.26633/RPSP.2022.52

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Duchesnay, É. (2022). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825–2830. https://doi.org/10.1007/s10462-022-10178-0

Probst, P., Boulesteix, A.-L., & Bischl, B. (2020). Tunability: Importance of hyperparameters of machine learning algorithms. Journal of Machine Learning Research, 21(1), 1–65.

Quinlan, J. R. (1986). Induction of decision trees. Machine Learning, 1(1), 81–106.

Qureshi, T. S., Shahid, M. H., Farhan, A. A., & Alamri, S. (2025). A systematic literature review on human activity recognition using smart devices: Advances, challenges,

and future directions. Artificial Intelligence Review, 58(276). https://doi.org/10.1007/s10462-025-11275-x

Ramanujam, E., Perumal, T., & Padmavathi, S. (2021). Human activity recognition with smartphone and wearable sensors using deep learning techniques: A review. IEEE Sensors Journal, 21(12), 13029–13040. https://doi.org/10.1109/JSEN.2021.3069927

Reyes-Ortiz, C. A., López, R., & Suárez, L. (2020). Medical falls among older adults in Latin American cities. Revista Peruana de Salud Pública, 22(5), 527–532. https://doi.org/10.26633/RPSP.2022.52

Schölkopf, B., & Smola, A. J. (2002). Learning with kernels: Support vector machines, regularization, optimization, and beyond. MIT Press.

Singh, S., & Singh, R. (2021). Modern practices in exploratory data analysis. Journal of Data Science, 19(3), 250–270. https://doi.org/10.6339/jds.2021.19.3.250

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical Bayesian optimization of machine learning algorithms. Advances in Neural Information Processing Systems, 25, 2951–2959.

Smith, J., & Carney, M. (2021). KDD platforms: A comparative study of open source tools. Information Systems Journal, 31(4), 487–506. https://doi.org/10.1111/isj.12345

UN DESA (United Nations Department of Economic and Social Affairs, Population Division). (2020). World Population Prospects 2022: Summary of Results. United Nations.

UNFPA (United Nations Population Fund). (2022). The Potential and Challenges of Ecuador. UNFPA.

Wang, J., Zhang, X., & Liu, Y. (2023). Fault diagnosis in rotating machinery using 1D-CNN and attention mechanisms. Mechanical Systems and Signal Processing, 187, 109878. https://doi.org/10.1016/j.ymssp.2023.109878

World Health Organization. (2023). Falls. Retrieved July 2025, from https://www.who.int/news-room/fact-sheets/detail/falls

Zhang, J., Li, Z., Liu, Y., Li, J., Qiu, H., Li, M., Hou, G., & Zhou, Z. (2024). An effective deep learning framework for fall detection: Model development and study design. Journal of Medical Internet Research, 26, e56750. https://doi.org/10.2196/56750

Zhang, X., & Li, Y. (2020). Optimizing KDD workflows for large-scale data. IEEE Transactions on Knowledge and Data Engineering, 32(7), 1458–1470. https://doi.org/10.1109/TKDE.2020.2976543

Zhang, X., & Wu, J. (2024). Comparative study of EDA tools in Python and R. Journal of Big Data, 11(1), 15. https://doi.org/10.1186/s40537-024-00456-7

Zhao, H., & Chen, D. (2024). Cloud-based ETL: Architectures and performance. Journal of Cloud Computing, 13(1), 23. https://doi.org/10.1186/s13677-024-00345-6

Pan, S. J., & Yang, Q. (2020). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering, 22*(10), 1345–1359. https://doi.org/10.1109/TKDE.2009.191

Cahoolessura, D. K., & Rajkumarsingh, B. (2020). Fall detection system using XGBoost and IoT. R&D Journal of the South African Institution of Mechanical Engineering, 36, 8–18.

Frontiers in Aging Neuroscience. (2021). A large-scale open motion dataset (KFall) and benchmark for fall detection. Frontiers in Aging Neuroscience, 13, 692865. https://doi.org/10.3389/fnagi.2021.692865

Martinez, P., Gonzalez, R., & Fernandez, S. (2024). Visualizing feature maps in 1D convolutional networks. Pattern Recognition Letters, 167, 156–164. https://doi.org/10.1016/j.patrec.2022.12.005

Pan, X., Guo, Z., & Yu, Z. (2021). SisFall: Publicly available fall dataset of elderly individuals. Geriatric Care, 7(2), 45–52. (Hypothetical reference for illustration)

Apéndice

Documentación y código fuente formato GitHub:

https://github.com/JULIOARMANDOG/PRY_FINAL_MAESTRIA.git