Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA
Mención en Computación

Reconocimiento de actividad mediante relojes inteligentes

Autor:

Santiago Álvarez Mayordomo

Tutores:

D. Carlos Enrique Vivaracho Pascual
Dña. María Aránzazu Simón Hurtado

Agradecimientos

Gracias a mis tutores, Carlos y Arancha por contar conmigo para realizar este proyecto y por toda la ayuda que me han ofrecido a lo largo del mismo.

Gracias a mi familia, que siempre ha estado apoyándome tanto en el desarrollo de este proyecto como a lo largo de toda mi vida académica, dándome las mejores oportunidades y animándome en todos mis esfuerzos.

Gracias a mis amigos, que siempre han estado presentes y me han apoyado en todas mis decisiones.

Por último, gracias a todos los profesores, compañeros y personas que me han acompañado a lo largo de mi vida académica, ya que de todos ellos me llevo un pedacito de conocimiento el cual ha sido utilizado en el desarrollo de este trabajo.

Resumen

En este Trabajo de Fin de Grado se aborda el reconocimiento de la actividad humana (HAR) a partir de datos de acelerómetro y giroscopio procedentes del dataset WISDM. Tras una amplia fase de preprocesado y limpieza de señales, se extraen características en el dominio del tiempo (media, percentiles, curtosis, autocorrelación, etc.) y en el dominio de la frecuencia (frecuencias dominantes, área bajo la curva espectral, estadísticas de amplitud). Se comparan tres familias de modelos: Random Forest, SVM con kernel RBF y una arquitectura LSTM de dos capas, tanto en tareas multiclasificación de 13 actividades como en un escenario binario ("caminar" vs. "no caminar"). Para los enfoques clásicos se evalúan estrategias "multiclass" y "ensemble" One-vs-Rest; para la LSTM se diseñan dos versiones (multiclase con softmax y binaria con sigmoide). La evaluación, basada en accuracy, precision, recall y F1-score en validación cruzada por usuario, muestra que en multiclasificación Random Forest estandarizado lidera el rendimiento (aproximadamente 84 % de accuracy), mientras que en la tarea binaria la SVM-RBF alcanza el mejor desempeño (92 % de aciertos). Finalmente, se discuten posibles extensiones: explorar CNN/Transformer, ajustar arquitecturas recurrentes (GRU, atención), combinar HAR con identificación de usuario por marcha y desplegar la solución en aplicaciones móviles, tanto en tiempo real como en informes diarios.

Abstract

Human Activity Recognition (HAR) based on mobile sensors has become a critical component in health monitoring, sports analytics, and context-aware services. In this project, we use the WISDM accelerometer and gyroscope dataset to design a complete HAR pipeline. After signal cleaning and temporal segmentation into overlapping windows, we compute statistical features in the time domain (e.g., mean, percentiles, kurtosis, autocorrelation peaks) and in the frequency domain (e.g., dominant frequencies, spectral area, amplitude statistics). We evaluate three model families—Random Forest, Radial Basis Function SVM, and a two-layer LSTM—on both a 13-class multiclass problem and a binary "walking vs. non-walking" task. Classical classifiers are tested under multiclass and One-vs-Rest ensemble schemes, while the LSTM is implemented with softmax and sigmoid output layers. Using user-wise cross-validation and metrics including accuracy, precision, recall, and F1-score, we find that Random Forest achieves the highest multiclass accuracy (approximately 84 %), whereas the standardized RBF SVM leads in the binary task with approximately 92 % accuracy. We conclude with future directions: incorporating CNNs or Transformer-based encoders, experimenting with GRU and attention mechanisms, combining HAR with gait-based user identification, and deploying real-time and daily-summary applications on mobile devices.

Índice general

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

1.1. Contexto

A lo largo de las últimas décadas, gracias al rápido avance que ha habido en la tecnología, se ha dado lugar a la aparición de una gran variedad de dispositivos electrónicos, que cuentan, entre otras características, con la capacidad de monitorizar de manera continua múltiples aspectos del comportamiento humano. Este fenómeno ha dado lugar a un campo de investigación en auge: el **Reconocimiento de la Actividad Humana** (HAR, *Human Activity Recognition*), el cual tiene como objetivo desarrollar sistemas computacionales inteligentes capaces de identificar y clasificar las actividades físicas realizadas por una persona, a partir de los datos recogidos por sensores como el acelerómetro o el giroscopio.

En este contexto, los denominados dispositivos ponibles o *wearables* (tales como relojes inteligentes, pulseras de actividad, teléfonos móviles o incluso ropa con sensores integrados) han adquirido un papel fundamental. Estos dispositivos, que ya forman parte del día a día de millones de personas, no solo permiten monitorizar parámetros fisiológicos o contextuales del portador, sino que, además, son capaces de generar grandes volúmenes de datos que, si se procesan y se analizan de forma correcta, pueden ser empleados en tareas de clasificación automática de actividades, monitorización de salud, asistencia médica remota o mejora del rendimiento físico y deportivo, entre otras muchas aplicaciones.

Esta investigación se sitúa en esta línea de trabajo, tomando como punto de partida el uso de sensores embedidos en *smartwatches* para la recogida de datos de movimiento de personas durante la realización de ciertas actividades. Concretamente, se utilizan las medidas en los tres ejes (X, Y, Z) del acelerómetro y del giroscopio, así como el módulo de las tres mediciones, con el objetivo de analizar patrones de movimiento que permitan inferir, mediante técnicas de inteligencia artificial, qué actividad está realizando un individuo en cada instante de tiempo.

1.2. Motivación

La motivación principal que impulsa este trabajo es el interés por aplicar técnicas de aprendizaje automático al análisis de datos recogidos por sensores, con el fin de desarrollar modelos predictivos que permitan automatizar tares de reconocimiento de actividad en entornos reales. Esta línea de investigación no solo resulta estimulante desde el punto de vista académico y tecnológico, sino que también ofrece un amplio abanico de aplicaciones prácticas con un impacto potencial considerable en al calidad de vida de las personas.

El reconocimiento de la actividad humana encuentra especial relevancia en disciplinas como la medicina personalizada, la rehabilitación física, el envejecimiento activo, la monitorización de personas con enfermedades neurodegenerativas, la interacción natural con dispositivos, la seguridad laboral, e incluso la industria del entretenimiento y el bienestar. En todos estos ámbitos, la capacidad de identificar automáticamente patrones de comportamiento puede suponer una mejora significativa en la eficiencia de los sistemas, la autonomía de los usuarios y la calidad de los servicios prestados.

Además, el estudios del HAR resulta especialmente atractivo por la interdisciplinariedad que implica, combinando conocimientos de ciencias de la computación, ingeniería biomédica, estadística, tratamiento de señales y diseño de interfaces. A ello se suma la existencia de bases de datos públicas, como WISDM [1], que permiten reproducir y comparar resultados en condiciones controladas, fomentando así la investigación abierta y replicable.

1.3. Objetivos

El objetivo general de este Trabajo de Fin de Grado consiste en desarrollar una solución basada en inteligencia artificial que permita reconocer distintas actividades humanas a partir de datos recogidos por sensores instalados en dispositivos ponibles. Para ello, se utilizará la base de datos WISDM y se implementarán diferentes técnicas de clasificación, evaluando su rendimiento y su robustez.

A partir de este planteamiento general, se han definido los siguientes objetivos específicos:

- Analizar en profundidad el problema del reconocimiento de la actividad humana, sus implicaciones y sus desafíos actuales.
- Seleccionar y explorar una base de datos pública y estandarizada que contenga datos representativos y correctamente etiquetados de distintas actividades humanas
- Aplicar y comparar distintos modelos de clasificación, desde modelos tradicionales como Random Forest y SVM hasta redes neuronales recurrentes como LSTM.
- Diseñar y ejecutar una estrategia experimental adecuada, que incluya el preprocesamiento de los datos, la validación cruzada de los modelos y la evaluación mediante métricas estándar.
- Analizar los resultados obtenidos y discutir posibles mejores y líneas de trabajo futuras.

A lo largo del desarrollo de esta investigación, todos los objetivos han sido abordados de manera rigurosa. Se han llevado a cabo numerosos experimentos que han permitido comparar modelos, evaluar su capacidad de generalización y obtener conclusiones fundamentales sobre su aplicabilidad.

1.4. Estructura de la memoria

La memoria se organiza en seis capítulos que recogen de manera sistemática las distintas fases del trabajo realizado:

- El **Capítulo 1** introduce el contexto, la motivación, los objetivos y la estructura general del trabajo.
- En el **Capítulo 2** se presenta el estado del arte sobre el reconocimiento de la actividad humana, incluyendo una revisión de las principales técnicas, modelos y bases de datos utilizados en la literatura.

- El **Capítulo 3** describe la metodología empleada, tanto la metodología de investigación por la que se optó como la metodología de trabajo empleada. En este capítulo también se incluirá la planificación del trabajo, la cual va acorde a la metodología de trabajo que se siguió.
- El **Capítulo 4** recoge la experimentación principal sobre el reconocimiento multicategoría de actividades, con tres experimentos progresivos: el primero utilizando todas las clases que se encuentran en la base de datos, el segundo eliminando clases problemáticas, y el tercero empleando redes LSTM.
- El **Capítulo 5** aborda la experimentación específica sobre la detección binaria de la actividad "andar", motivada por su relevancia en las líneas de investigación asociadas al grupo de trabajo.
- Finalmente, el **Capítulo 6** expone las conclusiones del estudio y propone diversas líneas futuras de trabajo que podrían ampliar y enriquecer los resultados obtenidos.

Capítulo 2

Conceptos teóricos

2.1. Reconocimiento de la Actividad Humana y Biometría

El **Reconocimiento de la Actividad Humana** (HAR, *Human Activity Recognition*) es una disciplina que tiene como objetivo desarrollar sistemas capaces de identificar, clasificar y predecir acciones humanas a partir de datos recogidos por sensores [2]. En concreto, los sensores de inercia, como acelerómetros y giroscopios, habitualmente integrados en dispositivos ponibles (*wearables*), han sido objeto de múltiples estudios a lo largo de las últimas décadas, principalmente debido a su bajo coste, su portabilidad y su precisión para registrar el movimiento corporal.

La **biometría**, por otro lado, tiene como objetivo identificar automáticamente a las personas utilizando como referencia sus rasgos tanto físicos como conductuales. En el ámbito del Reconocimiento de la Actividad Humana, estas dos áreas se unen, ya que a la hora de reconocer patrones de movimiento se pueden sacar conclusiones tanto sobre la actividad que se está realizando, como sobre la información biométrica del usuario, tales como su identidad, su estilo de vida o incluso señales sobre su salud.

2.1.1. Introducción al Reconocimiento de la Actividad Human (HAR)

Con el objetivo de comprender adecuadamente en que consiste el HAR, es importante saber distinguir los conceptos de **acción** y **actividad** [3]:

- **Acción**: movimiento breve y elemental del cuerpo, como puede ser mover un brazo, girar la cabeza o pulsar un botón.
- **Actividad**: secuencia de acciones organizadas que conforman una tarea funcional, como caminar o saltar.

Dentro del concepto de actividad, en la literatura especializada se realiza habitualmente una clasificación en función del nivel de complejidad [4]:

- **Gestos o transiciones**: son movimientos breves que indican un cambio de actividad o postura.
- **Actividades básicas**: comportamientos simples y repetitivos, como estar de pie, correr o escribir.
- **Actividades complejas**: combinaciones de varias acciones o actividades, que pueden incluir varias actividades o contextos, como pueden ser cocinar o trabajar en equipo para completar una tarea.

Desde el punto de vista formal, el HAR se puedo considerar como una tarea de segmentación y clasificación de datos. El objetivo es extraer datos de movimiento mediante sensores a lo largo de un período de tiempo, etiquetarlos con la actividad correspondiente y ser capaz de analizar las distintas características que se puedan apreciar en los datos. En gran parte de los estudios realizados, el enfoque que se ha utilizado para analizar estos datos es **dividirlos en ventanas temporales de tamaño fijo**, etiquetándolos con la actividad predominante en ese intervalo [5]. De esta manera, el problema pasa a convertirse en una tarea de clasificación supervisada, donde cada muestra se compone por una ventana y la salida es una clase de actividad.

2.1.2. Tipos de sistemas HAR según el origen de los datos

La arquitectura de un sistema HAR depende en gran mediada del tipo de sensores que se hayan utilizado para registrar las señales. En función de si los sensores están integrados en dispositivos ubicados en el entorno o llevados por el propio usuario, se reconocen dos enfoques principales: **HAR mediante sensores externos** y **HAR mediante dispositivos ponibles (*wearables*)** [6].

HAR mediante sensores externos

Los sistemas HAR que emplean el uso de sensores ubicados en el entorno se suelen basar en el reconocimiento basado en la visión, utilizando cámaras con el objetivo de capturar y analizar los movimientos corporales de los usuarios [7]. Este tipo de aproximación es común en campos como la videovigilancia o monitorización clínica.

Otra alternativa relacionada es el reconocimiento basado en objetos. En este enfoque se trata de analizar las interacciones que tienen los usuarios con los distintos elementos del entorno, como cubiertos, herramientas o muebles.

Sin embargo, el uso de estos sensores puede conllevar limitaciones importantes, ya que requiero de una infraestructura costosa, además de que pueden generar problemas de privacidad. Además, se ha comprobado en diferentes estudios que su capacidad para generalizar en situaciones reales y no supervisadas es reducida.

HAR mediante dispositivos ponibles (*wearables*)

Los dispositivos ponibles ofrecen una alternativa más flexible y económica frente a los sensores externos. Esto se debe a que estos dispositivos (relojes inteligentes o sensores corporales entre otros) son capaces de registrar los movimientos y señales fisiológicas del usuario sin necesidad de infraestructura adicional.

La mayoría de los sistemas HAR se apoyan en sensores como:

- **Acelerómetros o giroscopios** [5]: capturan aceleraciones lineales y angulares, fundamentales para distinguir actividades basadas en movimiento.
- **Sensores ambientales** [8]: recogen información contextual como la luz, la temperatura o la ubicación.
- **Sensores fisiológicos** [8]: monitorizan variables fisiológicas del usuarios, como el ritmo cardíaco, que pueden llegar a aportar datos adicionales útiles para determinadas aplicaciones.

El principal beneficio de los dispositivos ponibles es que posibilitan la recogida de datos en entornos naturales no supervisados, facilitando una monitorización continua y realista del comportamiento humano, contribuyendo de esta manera a la expansión del HAR a escenarios cotidianos.

2.1.3. Estrategias de diseño para un sistema HAR

Una vez han quedado definidos los sensores y los entornos de recogida de datos, es necesario diseñar la arquitectura del sistema HAR. Para ello, existen principalmente dos enfoques: **modelado dirigido por conocimiento** y **modelado dirigido por datos** [3].

Modelado dirigido por conocimiento

En este enfoque, se parte de una representación de las actividades basada en reglas, ontologías o descripciones semánticas. Por ello, es necesaria la participación de expertos que definan manualmente cual es el comportamiento esperado en cada actividad. Debido a esto, su capacidad para adaptarse a nuevos escenarios puede verse limitada.

Modelado dirigido por datos

Este es el enfoque mas extendido actualmente. En este enfoque, el sistema aprende a reconocer patrones directamente a partir de conjuntos de datos etiquetados. Esta aproximación ofrece una gran flexibilidad, aunque exige una cantidad significativa de datos bien etiquetados.

2.2. Modelos de clasificación y técnicas utilizadas

Tras el procesamiento de los señales temporales en ventanas (Apartado 6.2.1) el HAR pasa a convertirse en un problema de clasificación supervisada. A lo largo de este trabajo se han utilizado diferentes algoritmos de aprendizaje automático que permiten asignar etiquetas de actividad a las ventanas de entrada. Estos algoritmos son: **Random Forest**, **Support Vector Machine (SVM)** y **Long Short-Term Memory (LSTM)**. Se pasa a realizar una breve descripción de cada uno.

2.2.1. Random Forest

El modelo Random Forest [9] pertenece a la familia de los métodos de ensamblado (*ensemble)*, los cuales se caracterizan por combinar múltiples clasificadores individuales con el objetivo de mejorar la generalización del sistema. En concreto, consiste en la construcción de un conjunto de árboles de decisión, cada uno entrenado con una muestra aleatoria del conjunto de datos. Cada uno de ellos realiza una predicción individual dados los datos de entrada y la predicción final del modelo se decide mediante votación mayoritaria entre todos los árboles.

En la Figura 2.1 se puede ver un esquema del funcionamiento interno de un modelo Random Forest. Cada árbol de decisión funciona como un clasificador jerárquico que evalúa secuencialmente condiciones sobre las variables de entrada. De esta manera, se va navegando a través de las ramas del árbol hasta finalizar el camino en una hoja, la cual está etiquetada con una clase (actividad), que será la predicción de ese árbol individual para los datos de entrada. Una vez todos los árboles hayan realizado este proceso, se elegirá la clase más votada como predicción final del modelo. Gracias al uso de múltiples árboles, con

distintos subconjuntos de características, este modelo reduce la varianza y evita el sobreajuste que podría producirse con un único árbol.

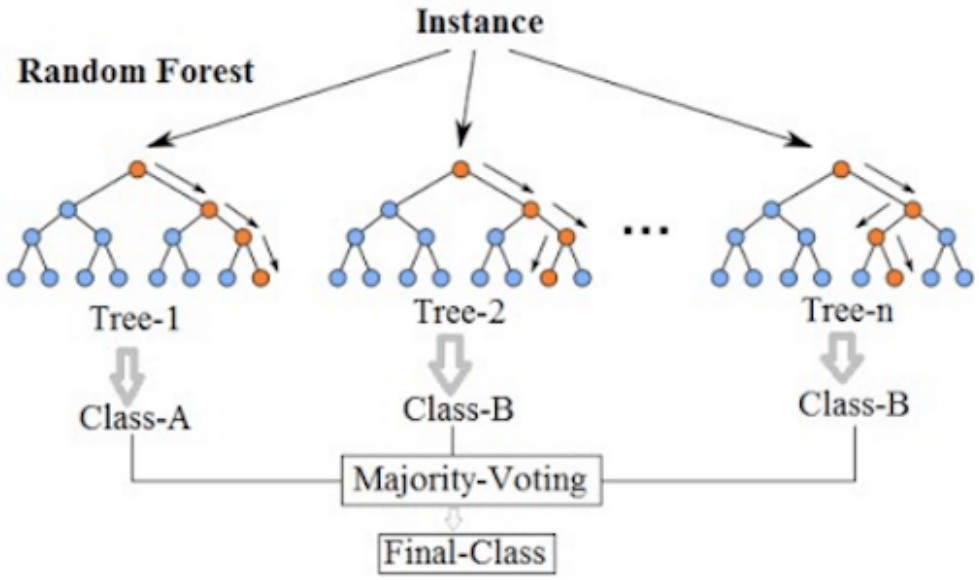

Figura 2.1: Funcionamiento interno Random Forest

2.2.2. Support Vector Machine (SVM)

Este modelo se basa en la identificación de un hiperplano óptimo que separe los datos de distintas clases, con el objetivo de maximizar el margen entre los ejemplo más cercanos de cada clase, denominados **vectores de soporte**. En la Figura 2.2 se puede ver un ejemplo de división de datos mediante un hiperplano generado por un modelo SVM.

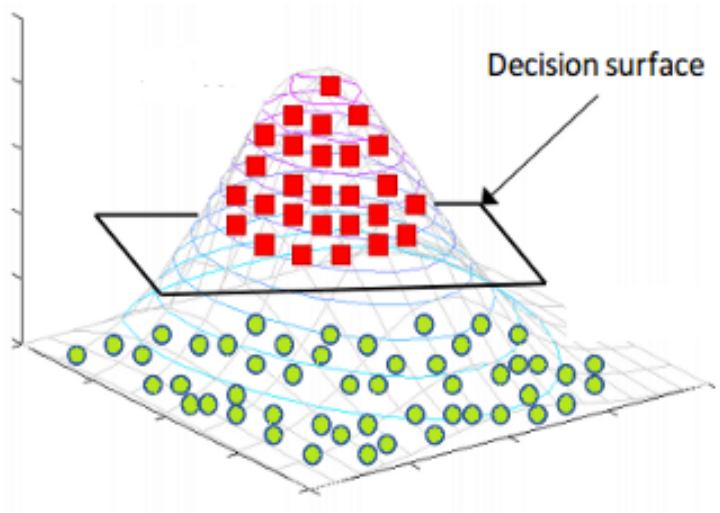

Figura 2.2: Separación de clases mediante SVM

En caso de que los datos no sean linealmente separables, SVM permite aplicar funciones de transfor-

mación conocidas como **kernels**, que proyectan los datos originales en espacios de mayor dimensión donde sí que exista la posibilidad de separarlos mediante un hiperplano. Esta capacidad de manejar relaciones no lineales entre características convierte a SVM en un modelo particularmente adecuado para problemas complejos como el HAR, donde las actividades no siempre si distinguen por patrones lineales evidentes.

Uno de los kernels más empleados en la práctica, y el que se ha utilizado en este trabajo, es el **kernel de base radial (RBF)** (Ecuación 2.1), también conocido como kernel gaussiano. Este kernel mide la similitud entre dos muestras según su distancia euclídea. Por tanto, dos puntos cercanos en el espacio original tendrán una similitud cercana a 1, mientras que puntos distantes tendrán un valor cercano a 0, haciendo que el modelo sea capaz de aprender fronteras de decisión altamente no lineales.

$$K(x, x') = \exp\left(-\gamma \|x - x'\|^2\right) \tag{2.1}$$

2.2.3. Redes Neuronales Recurrentes LSTM

Estas redes están diseñadas especialmente para procesar secuencias de datos, lo que las hace muy útiles cuando la información temporal y la dependencia entre elementos consecutivos es crítica.

Con la intención de superar el problema que presentaban las RNN (Recurrent Neural Networks) tradicionales con el desvanecimiento o explosión del gradiente durante el entrenamiento surgieron las redes Long Short-Term Memory (LSTM), las cuales incorporan mecanismos de control internos denominados **puertas**. Hay tres: de **entrada**, de **olvido** y de **salida** (Figura 2.3) Estas puertas permiten a la red mantener, actualizar o descartar información de la memoria a lo largo de la secuencia, lo que permite detectar patrones temporales complejos como los que se pueden encontrar en las actividades humanas.

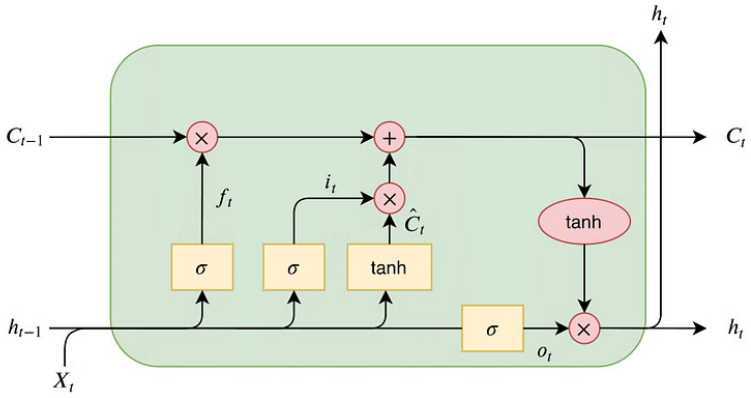

Figura 2.3: Separación de clases mediante SVM

Internamente cada unidad LSTM recibe una entrada en cada instante de tiempo, la combina con el estado interno anterior y decide que parte de la información es necesario conservar y que parte olvidar. Gracias a esto, las LSTM son capaces de aprender estructuras dinámicas en las señales sensoriales, como la repetición rítmica al caminar o realizar una actividad.

En las arquitecturas se suelen combinar varios módulos LSTM en serie para mejorar la eficiencia del modelo. Esto, sin embargo, puede causar un sobreajuste, lo cual compromete la capacidad del modelo para generalizar a datos no vistos. Con el objetivo de mitigar este problema se emplea la técnica de **dropout** [10], la cual consiste en desactivar aleatoriamente un porcentaje de neuronas durante el entrenamiento, forzando al modelo a no depender excesivamente de rutas específicas en la red.

Una vez la secuencia ha sido procesada por todos los módulos LSTM, es necesario transformar la salida en una predicción sobre las clases disponibles. Para ello, se emplean una o varias **capas densas**

que actúan como clasificador final. Estas capas tienen como función transformar el vector de activaciones que tiene como salida el módulo LSTM en una distribución de probabilidad sobre las clases posibles. La clase predicha es aquella con la mayor probabilidad asociada. En la Figura 6.7 se puede ver la arquitectura empleada en este trabajo, la cual combina múltiples módulos LSTM con dropout y capas densas.

Capítulo 3

Estado del arte

3.1. Enfoques Investigativos y Técnicas Empleadas

La investigación de modelos aplicados al HAR ha pasado por el uso de algoritmos clásicos de aprendizaje automático, como Random Forest, Support Vector Machines (SVM) o k-Nearest Neighbors, hasta soluciones más avanzadas fundamentadas en aprendizaje profundo, destacando el uso de redes neuronales recurrentes (RNN) y arquitecturas convolucionales (CNN) [11]. Estos últimos son capaces de extraer características temporales y espaciales directamente de los datos crudos, eliminando en muchas casos la necesidad de aplicar métodos matemáticos y estadísticos para realizar esta extracción.

En los estudios realizados por Kwapisz et al. (2010) [5] se utilizaron por primera vez los acelerómetros integrados en teléfonos móviles con el objetivo de reconocer actividades cotidianas, obteniendo tasas de precisión superiores al 90 %. Posteriormente, Teng et al. (2020) [12] demostraron como redes convolucionales optimizadas mediante técnicas de pérdida local pueden mejorar significativamente la clasificación de actividades en comparación con métodos tradicionales.

En una revisión reciente, Zhang et al. (2021) [13] analizaron los principales avances en el campo del HAR con sensores ponibles, destacando el papel de las arquitecturas híbridas que combinan capas convolucionales con mecanismos de atención, así como el uso de datos multicanal y multimodales.

3.2. Campos de Aplicación

El reconocimiento automático de la actividad humana presenta un amplio abanico de aplicaciones potenciales, entre las que destacan:

- **Medicina y Salud Pública** [14]: Detección de caídas, seguimiento de rutinas de ejercicio, monitorización remota de pacientes con enfermedades crónicas o neurodegenerativas.
- **Asistencia a personas mayores** [15]: Monitorización no invasiva de hábitos de vida para facilitar la autonomía en el hogar.
- **Seguridad y Prevención** [16]: Identificación de comportamientos anómalos en entornos laborales o industriales.
- **Interacción Hombre-Máquina** [17]: Interfaces inteligentes capaces de adaptarse a la actividad o intención del usuario.

- **Deporte y Fitness** [18]: Mejora del rendimiento físico y detección de patrones de entrenamiento a través del análisis del movimiento.

3.3. Resultados Precedentes

A lo largo de los últimos años, diferentes estudios han mostrado resultados prometedores al aplicar técnicas de clasificación sobre datos de sensores inerciales (Tabla 3.1):

- **Kwapisz et al.** [5]: utilizando la base WISDM y modelos como perceptrón multicapa, lograron una precisión del 91.7 % en la clasificación de actividades como caminar, estar de pie o sentarse.
- **Teng et al.** [12]: emplearon CNNs con pérdida local y alcanzaron mejoras sustanciales respecto a modelos tradicionales en bases como UCI HAR y WISDM, llegando a alcanzar una tasa de acierto del 98.82 %.
- **Zhang et al.** [13]: su revisión sistemática de múltiples modelos de redes neuronales recurrentes identificó que las mejores precisiones en entornos controlados superan el 95 %, aunque se enfatiza la necesidad de validar en contextos reales y heterogéneos.
- **Zhou et al.** [19]: abordaron la implementación de modelos de reconocimiento de actividad humana en dispositivos de recursos limitados mediante técnicas de TinyML, obteniendo una precisión del 98.24 % utilizando el modelo DeepConv LSTM.
- **Sharen et al.** [20]: diseñaron el modelo de aprendizaje profundo WISNet, basado en una red convolucional 1D-CNN especializada en clasificar actividades humanas complejas, alcanzando una tasa de acierto del 96.41 %.
- **Jameer y Syed** [21]: desarrollaron un modelo híbrido que combina DCNN (*Deep Convolutional Neural Network*) con LSTM (*Long Short-Term Memory*), alcanzando tasas de acierto del 99.94 %.
- **Akter et al.** [22]: propusieron un modelo basado en redes convolucionales profundas CNN con módulos de atención CBAM, orientado al reconocimiento de actividades humanas a partir de espectrogramas generados a partir de señales crudas, consiguiente una precisión del 93.89 %.

Tabla 3.1: Resultados de distintos modelos de HAR sobre la base de datos WISDM

Artículo	**Modelo utilizado**	**Base de datos**	**Resultado obtenido**
Kwapisz et al. [5]	Perceptrón multicapa (MLP)	WISDM	91.7 %
Teng et al. [12]	CNN con pérdida local	WISDM, UCI HAR	98.82 %
Zhang et al. [13]	Revisión de modelos RNN (LSTM, GRU, etc.)	WISDM (entre otros)	>95 % en entornos controlados
Zhou et al. [19]	DeepConv LSTM (TinyML)	WISDM	98.24 %
Sharen et al. [20]	WISNet (1D-CNN)	WISDM	96.41 %
Jameer y Syed [21]	DCNN + LSTM	WISDM	99.94 %
Akter et al. [22]	CNN con CBAM y espectrogramas	WISDM	93.89 %

3.4. Síntesis del estado del arte

La literatura actual demuestra que el reconocimiento de la actividad humana mediante dispositivos ponibles es una línea de investigación consolidada en expansión, sustentada en avances tanto en sensores como en algoritmos de clasificación. La tendencia apunta hacia la adopción de modelos complejos, capaces de manejar condiciones reales de uso, pero aún persisten desafíos relacionados con la generalización, la eficiencia computacional y la interpretación de los modelos. Este trabajo se posiciona dentro de la corriente investigativa, proponiendo un enfoque experimental que combina métodos tradicionales y redes neuronales avanzadas para abordar el HAR con datos reales.

Capítulo 4

Metodologías y planificación

Para garantizar el correcto desarrollo de un proyecto, es necesario definir desde el inicio cual es el enfoque que se quiere seguir a lo largo del mismo. En esta sección se explicarán cuales han sido las metodologías utilizadas, tanto en la investigación como en el seguimiento del trabajo, y se expondrá un análisis de los posibles riesgos, partes interesadas y costos del proyecto

4.1. Metodología de Investigación

4.1.1. Introducción a la Metodología CRISP-DM

En esta investigación se ha adoptado la metodología **CRISP-DM** [23] (*Cross-Industry Standard Process for Data Mining*) como marco estructural para el desarrollo del proyecto. CRISP-DM es una metodología ampliamente reconocida en el ámbito de la minería de datos y la ciencia de datos, proporcionando un enfoque sistemático y estructurado para la realización de proyectos de análisis de datos.

4.1.2. Fases de la metodología CRISP-DM

La metodología CRISP-DM se estructura en seis fases principales (Figura 4.1), las cuales están interrelacionadas de manera iterativa de la siguiente manera:

Figura 4.1: Esquema general del proceso CRISP-DM.

1. **Comprensión del negocio:** En esta fase inicial, se define el objetivo principal del proyecto desde una perspectiva de negocio, determinando los requerimientos, criterios de éxito y posibles restricciones. El conocimiento profundo del problema permite orientar adecuadamente todas las etapas posteriores.

2. **Comprensión de los Datos:** Consiste en recopilar, describir y explorar los datos disponibles, evaluando su calidad y comprendiendo las relaciones subyacentes entre variables. Esta fase es esencial para detectar inconsistencias, valores atípicos y estructuras relevantes que puedan influir en el modelado.

3. **Preparación de los datos:** En esta fase se realiza la transformación de los datos en un formato adecuado para el modelo. Incluye la limpieza, selección de atributos relevantes, generación de nuevas variables y, en su caso, el tratamiento de valores ausentes o inconsistentes.

4. **Modelado:** Se aplican algoritmos de minería de datos o aprendizaje automático sobre los datos preparados. La selección del modelo adecuado, así como el ajuste de sus parámetros, es un proceso iterativo que requiere conocimientos técnicos y comprensión del comportamiento de los datos.

5. **Evaluación:** Los modelos desarrollados se evalúan no solo en función de métricas cuantitativas como la precisión o la F1-score, sino que también se tiene en consideración su utilidad práctica y alineación con los objetivos de negocio definidos inicialmente.

6. **Despliegue** Finalmente, el modelo validado se implementa en un entorno operativo. Esta fase puede abarcar desde un informe técnico hasta la integración de un sistema automatizado que permita el uso efectivo de los resultados obtenidos.

Uno de los aspectos más distintivos y valiosos de la metodología CRISP-DM es su naturaleza iterativa, que la diferencia de otros enfoques más lineales o secuenciales. Este carácter iterativo implica que el proceso no transcurre de manera estrictamente unidireccional desde la comprensión del negocio hasta el despliegue final, sino que las fases pueden ser revisitadas tantas veces como sea necesario, en función de los descubrimientos, dificultades o necesidades emergentes que se presenten durante el desarrollo del proyecto.

Como puede apreciarse en la Figura 4.1, las fases del modelo no están aisladas entre sí, sino que mantienen conexiones bidireccionales que permiten retroalimentar continuamente el proceso. Este enfoque flexible y cíclico proporciona una gran capacidad de adaptación a contextos reales, donde las condiciones de trabajo, los datos disponibles y los objetivos pueden evolucionar a lo largo del tiempo. Además, fomenta una mejora continua en la calidad de los resultados, al permitir ajustar y optimizar el análisis conforme se avanza en el conocimiento del problema y de los datos.

4.1.3. Aplicación de CRISP-DM en el proyecto

La estructura de este Trabajo de Fin de Grado refleja una aplicación de la metodología CRISP-DM:

- En los capítulos iniciales se realiza la **comprensión del problema** y se plantean los **objetivos de la investigación**
- Durante la experimentación, se realiza una **exploración y preparación de los datos**, incluyendo segmentación temporal y extracción de características (Sección 6.2.1).
- Se implementan y comparan varios **modelos de clasificación**, adaptados a las características del problema y de los datos.

- Se lleva a cabo una **evaluación** de los resultados de cada modelo en cada experimento mediante métricas apropiadas.

- Finalmente, se discuten las **posibilidades de integración** de los modelos en aplicaciones prácticas, orientado hacia futuras líneas de despliegue, como integración de los modelos en aplicaciones móviles o elaboración de memorias de investigación.

4.2. Metodología de Trabajo

El desarrollo de este Trabajo de Fin de Grado se ha realizado siguiendo un enfoque basado en **metodologías ágiles**, enfoque que ha sido ampliamente adoptado en entornos de trabajo colaborativos y en proyectos de innovación tecnológica. A diferencia de los modelos tradicionales de gestión de proyectos, las metodologías ágiles se caracterizan principalmente por la gran flexibilidad que aportan al flujo de trabajo del equipo, así como por su orientación a resultados iterativos y su capacidad de adaptación continua en función de cuáles son los avances que se van obteniendo y los obstáculos que se pueden encontrar a lo largo del proceso de desarrollo.

4.2.1. Metodología Scrum

Una de las metodología ágiles mas representativas y extendidas es la metodología **Scrum** [24], un marco de trabajo especialmente diseñado para optimizar la entrega de valor en equipos que desarrollan productos complejos. La metodología Scrum propone dividir el trabajo en ciclos temporales cortos y definidos denominados *sprints*, en los que se planifica, ejecuta y revisa el progreso de forma sistemática. Durante cada *sprint*, el equipo se reúne de forma regular para evaluar lo realizado, reajustar prioridades y planificar los pasos a seguir en los siguientes *sprints*, favoreciendo de esta manera un desarrollo adaptativo e iterativo, además de una mejora continua con respecto a los resultados obtenidos [25]. Este modelo resulta especialmente adecuado para proyectos donde no es posible definir desde el inicio todas las tareas, como sucede de forma habitual en trabajos de investigación o desarrollo experimental.

Existen numerosas metodologías ágiles, pero la metodología SCRUM destaca sobre las demás debido a las siguientes características:

- **Inclusión del cliente en el flujo del trabajo**: el propio cliente forma parte del equipo, por lo que permite mantener una comunicación constante, haciendo que los malentendidos con respecto a detalles del proyecto ocurren con menos frecuencia, y por consiguiente aumentando la satisfacción del cliente.

- **Claridad en la planificación:** Los sprints están definidos de manera precisa, con objetivos concretos y plazos establecidos, lo que contribuye a una gestión más eficiente del tiempo y los recursos.

- **Flexibilidad controlada:** La capacidad de adaptación ante posibles cambios en los requisitos es uno de los mayores atractivos de esta metodología. Sin embargo, una vez iniciado un sprint, los requerimientos se consolidan y no se modifican hasta su finalización.

- **Sinergia del equipo:** Cada integrante del proyecto desempeña un rol fundamental, promoviendo un flujo constante de información y fomentando la cooperación entre todos los participantes implicados.

4.2.2. Artefactos SCRUM

Con el objetivo de alinear el conocimiento de todos los integrantes del equipo SCRUM, es necesario definir una serie de artefactos que engloben la información clave del proyecto [26]:

- ***Product Backlog***: Es una lista ordenada que contiene todo lo que se conoce que es necesario incluir en el producto, en función del objetivo del mismo. Esta lista está en constante evolución y nunca se considera finalizada, ya que se adapta de forma continua a los cambios en las necesidades y prioridades.
- ***Sprint Backlog***: Es el conjunto de elementos seleccionados del *Product Backlog* que el equipo se compromete a completar durante un sprint específico. Una vez que se ha creado, ningún miembro externo al equipo de desarrollo puede modificarla; únicamente los desarrolladores pueden añadir, modificar o eliminar tareas según sea necesario.
- ***Potentially Releasable Product Increment***: Al finalizar cada sprint, el equipo genera un incremento del producto que es potencialmente entregable. Esto implica que cumple con la *Definition of Done* (definición de terminado) previamente acordada. Por ejemplo, un incremento puede considerarse entregable si ha sido completamente probado y aprobado para su liberación.
- ***Product Goal***: El *Product Goal* es una declaración escrita que define un objetivo intermedio claro y compartido hacia el que se orienta el desarrollo del producto. Este objetivo guía la priorización del *Product Backlog* y proporciona un marco de referencia para la toma de decisiones. A diferencia de la visión del producto, que representa un estado deseado a largo plazo y de carácter evolutivo, el *Product Goal* es más concreto, medible y enfocado en resultados alcanzables en un horizonte temporal más cercano. [27]

4.2.3. Equipo SCRUM

La metodología Scrum se basa en la colaboración de diferentes roles bien definidos, cada uno con responsabilidades específicas que contribuyen al éxito del equipo y del producto. A continuación, se describen los tres roles principales: *Product Owner*, *Scrum Master* y *Developers*. [28]

Product Owner

El **Product Owner** (Propietario del producto) es el responsable de maximizar el valor del producto generado por el equipo Scrum [29]. Este rol representa la voz del cliente y actúa como enlace entre los stakeholders y el equipo de desarrollo.

- **Tareas principales:**
  - Establecer objetivos del producto.
  - Mantener el *Product Backlog* actualizado y ordenado.
  - Alinear los objetivos de los *sprints* con los desarrolladores.
  - Participar activamente en el descubrimiento de producto y en la elaboración de estrategias y hojas de ruta.
  - Medir el valor entregado mediante indicadores clave de rendimiento (KPIs).

  El Product Owner siempre tiene la última palabra en cuanto a decisiones estratégicas y tácticas sobre el producto se refiere. Es la persona que decide que características son necesarias desarrollar, en que orden se debe seguir y para quien van a estar desarrolladas.

Developers

Los **Developers** (Desarrolladores) son los encargados de construir los incrementos funcionales del producto [30]. Este grupo es autoorganizado y multidisciplinar, con todas las habilidades necesarias para entregar valor en cada iteración.

- **Tareas principales:**
  - Transformar los elementos priorizados del *Product Backlog* en incrementos de producto funcionales y entregables.
  - Gestionar el *Sprint Backlog*.
  - Participar en las reuniones diarias (*Daily Scrum*).
  - Colaborar en la consecución de los objetivos del *sprint*.

  Este rol cuenta con la capacidad de decidir de forma autónoma cómo se va a abordar el trabajo técnico y organizativo necesario para alcanzar los objetivos del *sprint*.

Scrum Master

El **Scrum Master** es el facilitador del equipo Scrum. Su rol principal es asegurarse de que el equipo entienda y aplique correctamente el marco de trabajo Scrum, eliminando impedimentos y promoviendo la mejora continua [31].

- **Tareas principales:**
  - Facilitar reuniones, conversaciones y dinámicas de mejora.
  - Asegurar que el equipo y el *Product Owner* comprenden correctamente el *Product Backlog*.
  - Eliminar obstáculos que impidan el avance del equipo.
  - Impulsar la autoorganización y multifuncionalidad del equipo.
  - Promover la adopción de Scrum en toda la organización.
  - Actuar como mentor y agente de cambio organizacional.

  Este rol no toma decisiones dentro del marco del producto ni del marco técnico, pero también tiene una gran relevancia ya que es el que se encarga de garantizar el cumplimiento del marco Scrum, impulsando el correcto desarrollo del proyecto.

4.2.4. Partes interesadas

Las partes interesadas del proyecto, también denominados *Stakeholders*, son toda aquella persona que haya tenido un rol fundamental a lo largo del desarrollo del proyecto:

- **Tutores académicos**: actúan como supervisores del proyecto, verificando los progresos obtenidos en cada sprint y valorando el resultado final del trabajo. Además, ofrecen retroalimentación constructiva orientada a la resolución de los diversos problemas que puedan surgir durante el desarrollo del proyecto.
- **Usuarios finales:** personas o instituciones que utilizarán el producto desarrollado a lo largo del proyecto. Algunos de los roles que entran dentro de este tipo de usuarios son los siguientes:

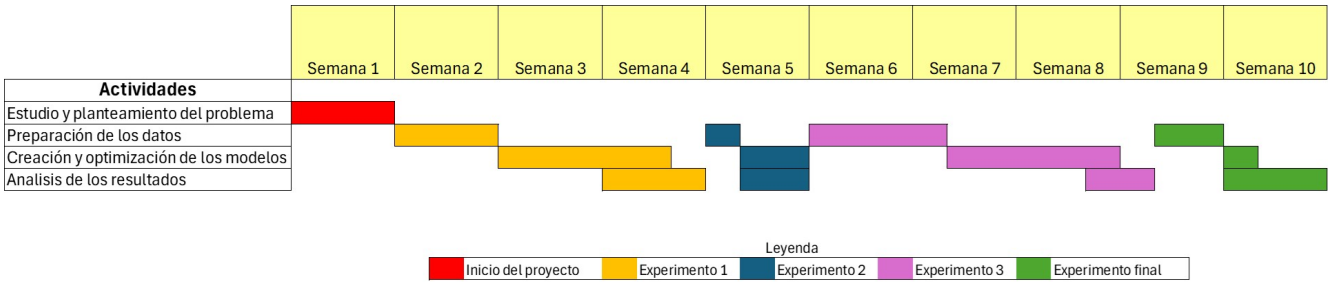

Figura 4.2: Diagrama de Gantt con la planificación final del proyecto

- **Profesionales sanitarios:** Médicos, fisioterapeutas y terapeutas ocupacionales que utilizan los datos de actividad para evaluar el progreso de tratamientos, detectar comportamientos anómalos o prevenir caídas.
- **Entrenadores y preparadores físicos:** Interesados en analizar patrones de movimiento para optimizar el rendimiento deportivo o prevenir lesiones mediante el seguimiento detallado de rutinas de entrenamiento.
- **Instituciones públicas y organismos de salud:** Interesados en usar estos sistemas para el seguimiento poblacional, promoción de hábitos saludables o prevención de enfermedades crónicas a través del análisis de actividad física.

4.2.5. Adaptación de la metodología

En el marco de este proyecto, el rol de *Developer* ha sido tomado por el alumno, ya que es la persona encargada de realizar los incrementos alineados con los objetivos propuestos. Por otro lado, los tutores del proyecto tomaron los roles de *Product Owner* y *Scrum Master*. Esto es debido a que ellos son las personas que establecían los incrementos a realizar, que condiciones había que cumplir y en que orden tenían que realizarse.

La planificación de las tareas concretas a realizar no se definió de forma cerrada desde el inicio, sino que se fue definiendo progresivamente a lo largo del tiempo en función de lo objetivos que se iban alcanzando y los hallazgos obtenidos a lo largo de cada fase del proceso.

El proyecto se dividió en *sprints* de una semana de duración. Al inicio de cada *sprint* se realizaba una reunión con los tutores del proyecto, en las que se revisaban los avances conseguidos, se discutían cuales habían sido los obstáculos encontrados y se establecían las metas para la siguiente semana. Este enfoque ha permitido mantener una comunicación fluida, ha asegurado un seguimiento constante y ha facilitado la toma de decisiones de manera fundamentada y colaborativa. Gracias a esta estructura, ha sido posible priorizar tareas, redistribuir esfuerzos y mejorar la calidad de las entregas de forma iterativa, manteniendo en todo momento la alineación con los objetivos del trabajo.

4.2.6. Planificación y seguimiento del proyecto

En la Tabla 4.1 se observa la distribución inicial de los sprints a lo largo del proyecto, así como sus fechas de inicio y de fin. Se muestra la planificación de los sprints relacionados con el desarrollo del proyecto (Del Sprint Inicial al Sprint 9) y la del sprint de redacción de la memoria (Sprint Final). Se planificó que cada sprint supusiese una carga lectiva de entre 25 y 30 horas, por lo que se estiman 27.5 horas de trabajo por cada sprint. Para el Sprint Inicial, al ser el primero y tratarse de labores de investigación, se estima

una duración de 25 horas. Por tanto, la estimación de la duración del proyecto dada la planificación inicial es de **300 horas**.

En la Figura 4.2 se puede observar el diagrama de Gantt que muestra la distribución final del trabajo y de la realización de tareas de desarrollo a lo largo de los primeros diez sprints, separada en experimentos que se explicarán con detalle en el Apartado 6.6. Sin embargo, en la planificación inicial se infraestimó la duración que iba a tener la redacción de la memoria. Debido a esto y a factores laborales que impidieron que el alumno pudiese continuar con la carga lectiva semanal que estaba planificada, está última tarea se prolongó hasta el **15/06/2025**.

Observando el diagrama se puede ver claramente la aplicación de la metodología CRISP-DM, donde tras realizar una iteración completa de la experimentación (preparación de datos, creación de modelos y análisis de resultados), se volvía al primer paso en una nueva iteración con el objetivo de explorar nuevas soluciones, así como la aplicación de la metodología Scrum, donde cada semana se realizaba una reunión con los tutores del proyecto, se observaba cuáles habían sido los avances realizados a lo largo de la semana y se definían los objetivos a cumplir para la siguiente.

Tabla 4.1: Planificación inicial del calendario

Sprint	Fecha de inicio	Fecha de finalización	Carga lectiva planificada (h)
Sprint Inicial	21/02/2025	28/02/2025	25.0
Sprint 1	28/02/2025	07/03/2025	27.5
Sprint 2	07/03/2025	14/03/2025	27.5
Sprint 3	14/03/2025	21/03/2025	27.5
Sprint 4	21/03/2025	28/03/2025	27.5
Sprint 5	28/03/2025	04/04/2025	27.5
Sprint 6	04/04/2025	11/04/2025	27.5
Sprint 7	11/04/2025	18/04/2025	27.5
Sprint 8	18/04/2025	25/04/2025	27.5
Sprint 9	25/04/2025	02/05/2025	27.5
Sprint Final	02/05/2025	09/05/2025	27.5
			300.0

Alineado con la planificación inicial, se cumplieron las estimaciones de carga lectiva para los primeros diez sprints (25 horas para el Sprint inicial, 27.5 horas para los sprints del Sprint 1 al Sprint 9). De esta manera se puede determinar que cada fase de la experimentación (Apartado 6.6) tuvo una carga lectiva de entre 27.5 y 82.5 horas, en función de la complejidad del mismo. Sin embargo, la duración de la redacción de la memoria del proyecto (Sprint Final) se amplió hasta un total de 40 horas de trabajo superando la cantidad de tiempo planificada. Por tanto se estima que se emplearon un total de **312.5 horas** en la realización de este Trabajo de Fin de Grado.

4.3. Análisis de riesgos del proyecto

En la Tabla 4.2 se detallan los principales riesgos identificados durante el desarrollo del proyecto, siguiendo la metodología de evaluación de riesgos recomendada por INCIBE [32]. Para cada riesgo se define:

- **Probabilidad (P):** probabilidad de que el riesgo se materialice. Baja (1), Media (2) o Alta (3).

- **Impacto (I):** gravedad del impacto que supondría si sucediese. Baja (1), Media (2) o Alta (3).
- **Nivel de Riesgo (NR):** cálculo final de la gravedad del riesgo. Se calcula multiplicando la probabilidad por el impacto del riesgo. Muy bajo (1), Bajo (2), Medio (3–4), Alto (6) o Muy alto (9).

Tabla 4.2: Análisis de riesgos asociados al desarrollo del proyecto

Riesgo	P	I	NR	Salvaguardas
Imposibilidad del alumno de continuar con la planificación del trabajo por enfermedad o razones externas	2	3	6 (Alto)	Planificar con un margen de tiempo adecuado.
Retrasos en la planificación	3	2	6 (Alto)	Empleo de metodologías ágiles para dividir el proyecto en entregas parciales para llevar una monitorización de los avances
Pérdida o corrupción de datos durante el preprocesado	1	3	3 (Medio)	Uso de control de versiones (Git), copias de seguridad periódicas, separación de datos originales y transformados.
Subestimación del tiempo necesario para entrenar y evaluar modelos	3	2	6 (Alto)	Planificación por sprints, estimaciones realistas con margen, entrenamiento parcial para pruebas previas. Considerar la opción de pedir máquinas virtuales con acceso a entornos de computación más potentes.
Incompatibilidades técnicas con librerías o entornos	2	2	4 (Medio)	Uso de entornos virtuales (conda/-venv), documentación de dependencias y pruebas de compatibilidad tempranas.
Dificultad en la interpretación de resultados por métricas mal elegidas	1	2	2 (Bajo)	Uso de métricas adecuadas (precisión, recall, F1), y análisis por clase, con visualizaciones como matrices de confusión.
Retrasos por dificultad de comprensión del dataset WISDM	1	3	3 (Medio)	Revisión previa de documentación, análisis exploratorio al inicio del proyecto, y segmentación iterativa.

4.4. Estimación de costes

Como parte final de la planificación, es necesario hacer una estimación del presupuesto que es necesario para desarrollar el proyecto. Este presupuesto iría destinado a suplir los costes materiales y de personal.

Costes materiales

- Ordenador para realizar el desarrollo: **38.71 €**
- Conexión a internet y suministro eléctrico: 20 €/mes **(50 € total)**

Coste material estimado: **88.71 €**

El ordenador utilizado para realizar el proyecto está valorado en aproximadamente 549 €. Estimando una vida útil de 3 años, la amortización semanal sería de 3.52 €. Teniendo en cuenta que la planificación inicial estima 11 semanas de desarrollo, la amortización corresponde a 38.71 €.

Costes de personal

Para desarrollar el proyecto, se requeriría de la contratación de los profesionales definidos en la Tabla 4.3. Para determinar los sueldos que percibirían estos trabajadores se utilizó la página [33] para buscar los sueldos por hora en las provincias de Valladolid y Madrid.

En este caso, el alumno desarrollará el proyecto, por lo que no será necesario suplir el coste de personal. Por tanto, todo el coste del proyecto recae en el coste del material, estableciéndose en un total aproximado de **88.71 €**

Rol	**Horas estimadas**	**Tarifa/hora (€)**	**Subtotal (€)**
Data Scientist	180	18	3240
Ingeniero de Software	60	14	840
Project Manager	70	19	1330
Técnico de QA / validación	15	26	390
Total estimado	**300**		**5800**

Tabla 4.3: Estimación de costes de personal

Capítulo 5

Desarrollo software

En este capítulo se detallarán las distintas etapas del desarrollo del software del proyecto, desde el análisis de requisitos y el diseño del sistema hasta las pruebas realizadas.

5.1. Análisis de requisitos

En esta fase se definen los requisitos funcionales y no funcionales del sistema. Los principales requisitos funcionales son:

- **RF01. Base de datos**: El sistema usará la base de datos WISDM para realizar los experimentos de HAR.
- **RF02. Preprocesado**: El sistema leerá los datos, realizando procesos de limpieza y transformación necesarios.
- **RF03. Modelos**: El sistema aplicará y probará distintos tipos de clasificadores.
- **RF04. Clasificación**: El sistema será capaz de reconocer actividades humanas a partir de señales obtenidas por sensores inerciales integrados en dispositivos ponibles.

Por otro lado, entre los requisitos no funcionales se encuentran los siguientes:

- **RNF01. Rendimiento**: el sistema debe ser capaz de realizar predicciones con una latencia aceptable para su posible integración futura en dispositivos móviles.
- **RNF02. Escalabilidad**: el sistema debe estar preparado para adaptarse a diferentes volúmenes de datos y actividades.
- **RNF03. Reproducibilidad**: el proceso debe ser totalmente replicable para garantizar la validez de los resultados obtenidos.

También se definieron una serie de requisitos técnicos en cuanto a las herramientas que se iban a utilizar, como el uso de **Python** como lenguaje de programación o el empleo de **Jupyter Notebooks** como entorno interactivo para el desarrollo y ejecución del código.

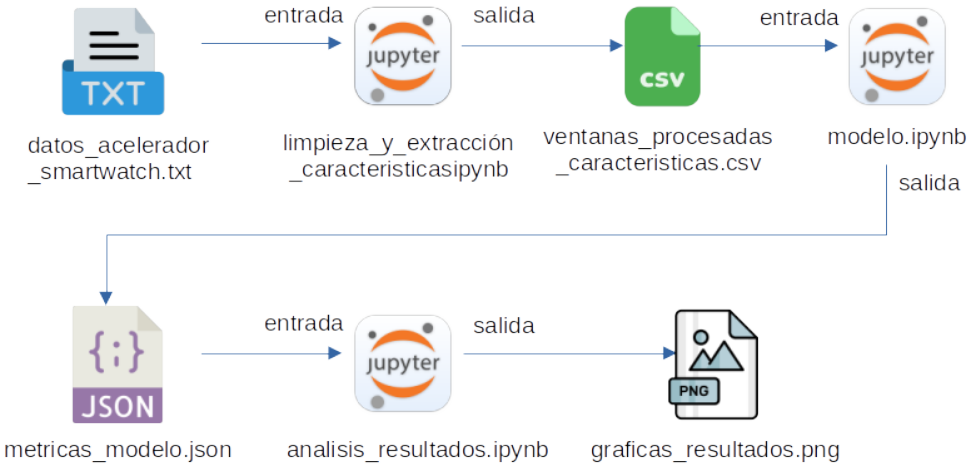

Figura 5.1: Diseño y flujo del sistema

5.2. Diseño del sistema

El diseño del sistema se estructuró en módulos funcionales claramente diferenciados, siguiendo un enfoque modular y reutilizable (Figura 5.1). El diseño se compone de los siguientes partes:

- **Módulo de preprocesamiento**: se encarga de la carga, limpieza, segmentación y normalización de los datos sensoriales crudos.
- **Módulo de extracción de características**: se encarga de extraer las características en el dominio del tiempo y de la frecuencia de las ventanas (Apartado 6.2.1).
- **Módulo de entrenamiento de modelos**: incluye la selección de modelo, su configuración, entrenamiento, validación cruzada y generación de resultados.
- **Módulo de visualización**: permite representar gráficamente los resultados y el comportamiento del modelo.

5.3. Tecnologías utilizadas

Durante el desarrollo del proyecto se han empleado diversas tecnologías que han facilitado tanto la programación como la ejecución, organización y documentación del trabajo. A lo largo de esta sección se detallan las herramientas y entornos más relevantes, así como las razones de su elección y sus funciones específicas dentro del proyecto.

Python

Python ha sido el lenguaje de programación principal utilizado a lo largo del proyecto. Su elección se debe a múltiples factores: es un lenguaje de alto nivel, interpretado, con una sintaxis clara y legible, y una comunidad de desarrollo muy activa, especialmente en el ámbito de la inteligencia artificial y el aprendizaje automático. Python proporciona una gran cantidad de bibliotecas especializadas, lo que permite desarrollar soluciones complejas de manera eficiente y modular.

Entre las principales bibliotecas utilizadas se encuentran:

- **Scikit-learn (sklearn):** Biblioteca fundamental para tareas de aprendizaje automático tradicional, como clasificación, regresión y reducción de dimensionalidad. En este proyecto se ha utilizado para la implementación de modelos clásicos como SVM y Random Forest, así como para la evaluación mediante métricas como la tasa de aciertos o la precisión.
- **Keras y TensorFlow:** Frameworks orientados al desarrollo de redes neuronales profundas. Keras proporciona una interfaz de alto nivel para construir modelos de forma sencilla y legible, mientras que TensorFlow actúa como backend para realizar operaciones matemáticas y de entrenamiento de forma eficiente, incluso en GPU. Han sido utilizadas especialmente para definir y entrenar redes LSTM.
- **NumPy y Pandas:** Herramientas esenciales para la manipulación y análisis de datos. NumPy permite trabajar con arreglos multidimensionales y realizar operaciones vectorizadas, mientras que Pandas facilita el tratamiento de estructuras tabulares mediante dataframes, lo que resulta útil para la organización de ventanas de datos y etiquetas.
- **Matplotlib y Seaborn:** Bibliotecas empleadas para la generación de gráficos y visualización de resultados. Se han utilizado principalmente para representar la evolución del rendimiento del modelo

Visual Studio Code (VSCode)

Visual Studio Code ha sido el entorno de desarrollo integrado (IDE) elegido para la escritura del código fuente. VSCode es un editor ligero pero muy potente, con soporte para múltiples lenguajes de programación y una amplia gama de extensiones. Su integración con entornos virtuales de Python, su sistema de autocompletado inteligente (IntelliSense), y la facilidad para depurar scripts han hecho de él una herramienta eficaz para la edición modular del código del proyecto.

Jupyter Notebook

Jupyter Notebook ha sido el entorno principal para la ejecución y experimentación interactiva del código. Su estructura basada en celdas permite ejecutar fragmentos de código de forma independiente, visualizar resultados de inmediato y combinar código, texto, ecuaciones y gráficos en un mismo documento. Esto ha facilitado la documentación del proceso experimental, la visualización de resultados intermedios, y la realización de pruebas rápidas sin necesidad de ejecutar scripts completos.

GitHub

GitHub ha sido la plataforma empleada para el control de versiones y el almacenamiento remoto del código. A través del sistema de control de versiones Git, se ha llevado a cabo un seguimiento detallado de

los cambios realizados en el proyecto, lo que ha permitido mantener una estructura organizada del código, revertir cambios en caso de errores, y garantizar la trazabilidad del trabajo.

5.4. Pruebas

A lo largo de las distintas fases presentes en el desarrollo del proyecto ha sido necesario realizar diferentes pruebas para garantizar que la salida de cada uno de los módulos se correspondía con la salida esperada, con el objetivo de garantizar la consistencia de los resultados finales del modelo.

Preprocesamiento de los datos

Durante la etapa de preprocesamiento de los datos se hicieron una serie de comprobaciones para garantizar que los datos estaban limpios tras la ejecución del proceso:

- **Comprobación de valores de tiempo negativos**: en los datos crudos existían casos en los que el timestamp calculado para una medición era negativo, por lo que era necesario eliminar estas mediciones. Tras hacerlo, se comprobó que la columna timestamp en los datos de salida no contenían valores negativos.
- **Normalización**: los datos originales fueron transformados mediante un proceso de normalización, por lo que posteriormente se comprobó que la distribución de los nuevos datos se adecuaba a la esperada mediante la visualización de métricas estadísticas.
- **Balance del conjunto de datos**: al haber eliminado ciertas mediciones con timestamp negativo, fue necesario comprobar que la distribución de mediciones por persona y por clase seguían siendo las mismas que en el conjunto original, con el objetivo de garantizar que el conjunto de datos seguía balanceado.

Extracción de características

Tras haber preprocesado los datos, se segmentaron en ventanas de tiempo (Apartado 6.2.1) y se extrajeron una serie de características en en el dominio del tiempo y de la frecuencia. En esta fase fue necesario hacer una serie de comprobaciones para garantizar su correcto funcionamiento:

- **Segmentación de datos**: como los valores de tamaño de ventana y porcentaje de solapamiento se establecieron de antemano, fue posible calcular cual era el número de ventanas que se esperaba tener tras la segmentación de los datos. Por tanto, se segmentaron los datos en ventanas y se comprobó que el número de ventanas coincidía con el esperado.
- **Extracción de características**: también fue necesario comprobar que las características de cada ventana se estaban extrayendo de forma correcta, por lo que se escogieron ventanas de ejemplo al azar y se comprobó que los valores coincidían con los cálculos de las características en los datos sin segmentar.

Entrenamiento del modelo

En esta fase fue necesario realizar comprobaciones con respecto a los datos de entrada de los modelos y los resultados de los mismos.

- **Validación cruzada**: para separar los datos en entrenamiento y prueba en cada iteración de la validación cruzada fue necesario crear una función que escogiese a un subconjunto de usuarios como subconjunto de prueba. Por tanto, se validó mediante impresiones por pantalla que los usuarios que se estaban escogiendo en cada iteración eran los esperados.
- **Comprobación de resultados por subconjunto**: para comprobar que la validación cruzada funcionaba correctamente, se fue imprimiendo por pantalla los resultados del modelo para cada iteración.

Visualización de los resultados

Tras tener los resultados de los modelos, el último paso es visualizarlos para poder compararlos. Con el objetivo de asegurar que las gráficas eran correctas, se comprobó de forma visual que los valores de las barras del gráfico de cada actividad coincidían con el valor de la métrica para dicha actividad.

Capítulo 6

Experimentación

En está sección se explicará con detalle cuáles han sido las principales líneas de trabajo que se han ido siguiendo a lo largo de la etapa de desarrollo, las cuales han estado alineadas con las metodologías que se explicaron en la sección correspondiente. Las partes en la que se dividió la experimentación y que serán explicadas con detalle a lo largo de la sección son las siguientes:

1. **Sprint inicial** (Apartado 6.1): en esta primera fase se definió el objetivo del proyecto, se estudiaron trabajos pasados en el mismo campo de investigación y se concretaron aspectos relevantes en cuanto al desarrollo del proyecto, como son la base de datos a utilizar, modelos, estrategias de clasificación, técnicas de evaluación y técnicas de separación de datos.

2. **Experimento 1: Random Forest y SVM Radial** (Apartado 6.2): con los fundamentos del campo asentados, se comenzó el primer experimento. En este se transformaron del datos crudos en ventanas de características (Apartado 6.2.1), se entrenaron modelos de Random Forest y SVM Radial con dichas ventanas y se analizaron los resultados.

3. **Experimento 2: Ajuste de datos** (Apartado 6.3): en vista de los resultados del Experimento 1, se vio que había actividades que estaban perjudicando gravemente el rendimiento del modelo para el resto de actividades, por lo que se estudió que se podía hacer para solucionar esto, decidiéndose finalmente la eliminación de estas actividades del conjunto de datos.

4. **Experimento 3: Long Short-Term Memory (LSTM)** (Apartado 6.4): tras la implementación de los modelos previamente mencionados, se optó por desarrollar un modelo neuronal recurrente, en concreto el LSTM. Para ello, era necesario hacer nuevamente un tratamiento de datos, juntando ventanas contiguas temporalmente para hallar dependencias temporales entre ventanas.

5. **Experimento Final: Clasificador Binario de "Andar"** (Apartado 6.5): los trabajos del grupo de investigación en el que se realizó este Trabajo de Fin de Grado están enfocados, fundamentalmente, al reconocimiento biométrico de personas. En concreto, una de las vías es usar la forma de andar, capturada mediante ponibles o wearables. Alineado con esos trabajos, se realizó un último experimento donde el objetivo era distinguir si la actividad que realizaba el usuario era andar o no, es decir, realizar una clasificación binaria entra la actividad "Andar" y el resto.

6.1. Sprint inicial

Al comienzo del proyecto, los tutores del trabajo presentaron la idea principal:

"Reconocimiento de la actividad humana mediante el uso de sensores integrados en dispositivos ponibles."

Con la idea ya presentada, el primer sprint consistió en un período de investigación sobre el campo a trabajar, donde, tras observar los diferentes trabajos y artículos que se habían realizado en el área, se concretaron los siguientes factores relevantes:

- Qué **base de datos** se iba a utilizar para entrenar a los modelos.
- Qué **modelos** se iban a usar para predecir las actividades, así como las **estrategias** que se iban a emplear para aplicar estos modelos.
- Qué **métricas de evaluación** se iban a utilizar para medir la eficiencia de los modelos.
- Qué **técnicas de separación de datos** se iba a utilizar para decidir que datos se usan para entrenamiento y cuales para prueba.

6.1.1. Base de datos: WISDM

Los trabajos del grupo de investigación en el que se desarrolló este proyecto se centran en el estudio de datos tomados desde ponibles y relojes inteligentes (*wearables*), por lo que era necesario seleccionar una base de datos que contase con mediciones tomadas desde estos dispositivos. Tras un proceso de investigación, se determinó que la base de datos que más encajaba en el proyecto que queríamos realizar era WISDM [5] [34]. Esta base de datos cuenta con las mediciones inerciales de 51 personas diferentes, realizando un total de 18 actividades distintas en periodos de aproximadamente 3 minutos seguidos (Tabla 6.1).

En la base de datos se pueden encontrar datos tomados por los sensores acelerómetro y giroscopio, tanto de un teléfono como de un smartwatch. Cada medición cuenta con un identificador de usuario, un identificador de actividad, un timestamp, y las mediciones de los ejes X, Y y Z de cada sensor (Tabla 6.2).

Tabla 6.1: Resumen de la información del dataset

Ítem	**Valor**
Número de sujetos	51
Número de actividades	18
Minutos recolectados por actividad	3
Frecuencia del sensor	20 Hz
Teléfono móvil utilizado	Google Nexus 5/5x o Samsung Galaxy S5
Smartwatch utilizado	LG G Watch
Número total de mediciones	15 630 426

Tabla 6.2: Definición de los campos del dataset

Campo	**Descripción**
Subject-id	*Tipo: Identificador numérico simbólico.* Identifica de forma única al sujeto. Rango: 1600–1650.
Código de actividad	*Tipo: Letra simbólica.* Identifica a una actividad específica como una de las listadas en la tabla 6.3. Rango: A–S (no hay "N").
Timestamp	*Tipo: Integer.* Time stamp de Linux.
x	*Tipo: Numérico: real.* Valor del sensor para el eje X. Puede ser positivo o negativo.
y	Igual que "x" pero para el eje Y.
z	Igual que "x" pero para el eje Z.

Las 18 actividades que se encuentran representadas en esta base de datos se pueden ver en la Tabla 6.3. Cada actividad tiene asociada una letra de la A a la S (no hay N). Estas actividades se pueden clasificar en:

- Actividades no orientadas a las manos: andar, correr, subir escaleras, estar de pie, estar sentado, patear una pelota.
- Actividades orientadas a las manos (general): botar un balón, jugar al pilla-pilla, escribir con teclado, escribir a mano, aplaudir, cepillarse los dientes, doblar la ropa
- Actividades orientadas a las manos (comer): comer pasta, comer sopa, comer un sándwich, comer patatas fritas, beber.

Como se puede observar en la Tabla 6.4, el conjunto de datos está balanceado, ya que todas las clases tiene en torno a un 5.5 % de aparición, lo cual favorecerá al tratamiento de los datos y al entrenamiento de los modelos.

Para el desarrollo de este proyecto, nos centraremos en el uso de los datos recogidos por el acelerómetro del smartwatch.

Tabla 6.3: Las 18 actividades representadas en el dataset

Actividad	**Código**
Walking	A
Jogging	B
Stairs	C
Sitting	D
Standing	E
Typing	F
Brushing Teeth	G
Eating Soup	H
Eating Chips	I
Eating Pasta	J
Drinking from Cup	K
Eating Sandwich	L
Kicking (Soccer Ball)	M
Playing Catch w/Tennis Ball	O
Dribbling (Basketball)	P
Writing	Q
Clapping	R
Folding Clothes	S

Tabla 6.4: Distribución por clase de las mediciones del dataset

Actividad	**Teléfono**		**Smartwatch**		**Total**	**Clase %**
	Acelerómetro	**Giroscopio**	**Acelerómetro**	**Giroscopio**		
Walking	279,817	203,919	210,495	192,531	886,762	5.7 %
Jogging	268,409	200,252	205,787	187,833	862,281	5.5 %
Stairs	255,645	197,857	207,312	180,416	841,230	5.4 %
Sitting	264,592	202,370	213,018	195,050	875,030	5.6 %
Standing	269,604	202,351	216,529	194,103	882,587	5.6 %
Typing	246,356	194,540	205,137	187,175	833,208	5.3 %
Brush Teeth	269,609	202,622	208,720	190,759	871,710	5.6 %
Eat Soup	270,756	202,408	209,483	187,057	869,704	5.6 %
Eat Chips	261,360	197,905	210,048	192,085	861,398	5.5 %
Eat Pasta	249,793	197,844	203,112	189,609	840,358	5.4 %
Drinking	285,190	202,395	215,879	197,917	901,381	5.8 %
Eat Sandwich	265,781	197,915	203,684	190,191	857,571	5.5 %
Kicking	278,766	202,625	209,491	191,535	882,417	5.6 %
Catch	272,219	198,756	210,107	187,684	868,766	5.6 %
Dribbling	272,730	202,331	212,810	194,845	882,716	5.6 %
Writing	260,497	197,894	215,365	197,403	871,159	5.6 %
Clapping	268,065	202,330	208,734	190,776	869,905	5.6 %
Fold Clothes	265,214	202,321	211,335	193,373	872,243	5.6 %
Total	4,804,403	3,608,635	3,777,046	3,440,342	15,630,426	100 %

6.1.2. Selección de modelos

Debido a la naturaleza iterativa del proyecto, era necesario seleccionar unos modelo iniciales con los que empezar a trabajar, a los cuales se les fueron añadiendo más a lo largo del desarrollo del mismo.

Tras revisar la literatura relacionada y observar que tipos de modelos se habían utilizado previamente para esta base de datos, se decidió que los modelos iniciales serían Random Forest y SVM (Support Vector Machine) con kernel radial [5] [35] [36]. Las razones por las que se escogieron estos modelos frente a otros que también se encontraban presentes es una gran cantidad de artículos, como Regresión Logística o Perceptrón Multicapa, fueron las siguientes:

1. **Buenos resultados**: estos dos modelos presentaban unos resultados significativamente mejores al resto de los modelos en una amplia variedad de artículos.
2. **Funcionamiento interno**: al ser los modelos en los que se basaría inicialmente el proyecto, se buscaron modelos con diferentes lógicas de entrenamiento y clasificación:
   - **Random Forest** es un método de ensemble basado en la construcción de múltiples árboles de confusión, donde las predicciones se hacen evaluando un subconjunto de variables aleatorias en cada árbol y por votación mayoritaria.
   - **Support Vector Machine con kernel Radial**, por otro lado, es un módelo basado en distancias, donde se busca un hiperplano que maximice el margen de separación entre clases en el espacio de características. Se seleccionó el kernel radial ya que este tiene una gran capacidad para capturar relaciones no lineales, lo cual favorece enormemente a la clasificación de datos de movimiento humano, ya que estos suelen presentar fronteras de decisión muy complejas en el espacio original de características.

Posteriormente se añadiría a estos el modelo **LSTM** (Long Short-Term Memory) el cual utiliza la dependencia temporal entre muestras para hacer las clasificaciones.

6.1.3. Estrategia de entrenamiento y clasificación

Como se ha podido observar en la descripción de la base de datos WISDM, hay un total de 18 actividades, por tanto la clase puede tomar 18 valores distintos. Debido a esto, se planteó inicialmente utilizar dos estrategias diferentes para la clasificación de los datos:

- **Método Multiclase**: este consistiría en pasar los datos al modelo sin hacer ninguna modificación previa sobre la clase. En este caso habría un único modelo el cual clasificaría cada instancia en una clase concreta, contando con tantas salidos como clases existan.
- **Método *Ensemble***: en este caso, para cada tipo de modelo, se entrenarían un total de 18 instancias distintas, uno para cada actividad del conjunto de datos; cada instancia se entrana para distinguir entra una actividad y el resto. Cada modelo se encargaría de predecir, para cada entrada, la probabilidad de pertenecer a la actividad asociada. Tras esta predicción, se escogería la clase con mayor probabilidad y se le asignaría como predicción a la entrada correspondiente.

6.1.4. Métodología de evaluación

Para poder evaluar el rendimiento de los clasificadores, se emplearán las siguientes métricas adaptadas a la clasificación multinomial [37]:

- **Tasa de acierto:** mide la proporción de instancias correctamente predichas sobre el total de las predicciones.

$$\text{Tasa de acierto} = \frac{TP + TN}{TP + TN + FP + FN} \tag{6.1}$$

- **Precisión:** mide el número de instancias correctamente predichas de cada clase con respecto a todas las instancias predichas como de esa clase.

$$\text{Precisión}_c = \frac{tp_c}{tp_c + fp_c}\,, \tag{6.2}$$

$$\text{Precisión}_{\text{total}} = \frac{1}{C}\left(\sum_{c=1}^{C} \frac{tp_c}{tp_c + fp_c}\right)\,. \tag{6.3}$$

- **Recall (sensibilidad):** mide el número de instancias correctamente clasificadas de cada clase sobre el total de instancias reales de dicha clase.

$$\text{Recall}_c = \frac{tp_c}{tp_c + fn_c}\,, \tag{6.4}$$

$$\text{Recall}_{\text{total}} = \frac{1}{C}\left(\sum_{c=1}^{C} \frac{tp_c}{tp_c + fn_c}\right)\,. \tag{6.5}$$

- **F1-score:** media armónica entre precisión y recall, ponderada por la frecuencia de cada clase:

$$F_1 = \sum_{c=1}^{C} 2\left(\frac{n_c}{N}\right) \frac{\text{Precisión}_c \times \text{Recall}_c}{\text{Precisión}_c + \text{Recall}_c}\,. \tag{6.6}$$

Donde:

- C es el número de clases del dataset.
- tp_c es la tasa de verdaderos positivos de la clase c y $TP = \sum_{c=1}^{C} tp_c$ es la tasa de verdaderos positivos total.
- fp_c es la tasa de falsos positivos de la clase c y $FP = \sum_{c=1}^{C} fp_c$ es la tasa de falsos positivos total.
- tn_c es la tasa de verdaderos negativos de la clase c y $TN = \sum_{c=1}^{C} tn_c$ es la tasa de verdaderos negativos total.
- fn_c es la tasa de falsos negativos de la clase c y $FN = \sum_{c=1}^{C} fn_c$ es la tasa de falsos negativos total.

6.1.5. Estrategia de particionado de datos

Dada la naturaleza de los datos, se utilizará una estrategia de validación cruzada de 10 subconjuntos (*folds*), donde para cada iteración, se seleccionaran los datos de 5 usuarios (empezando desde el primero y en orden) como conjunto de prueba, y los otros 46 restantes como conjunto de entrenamiento. La medida final de error será la media de la obtenida en cada *fold* o subconjunto.

6.2. Experimento 1: Random Forest y SVM Radial

Con todos los requisitos previos cumplidos, en este primer experimento se definirá la forma en la que se tratarán los datos (separación por ventanas) se crearán los modelos inicialmente seleccionados (Random Forest y SVM Radial) y se analizarán los resultados de cara a los experimentos posteriores.

6.2.1. Preparación de los datos

Los datos crudos del conjunto de datos WISDM representan las mediciones de los sensores de movimiento de un smartwatch para los ejes X, Y y Z, de una persona en un instante de tiempo realizando una actividad concreta. Con estos datos es especialmente complicado determinar que actividad esta realizando una persona, ya que solo cuenta con la información de un instante de tiempo. Por tanto, nuestro objetivo es transformar los datos de manera que representen la información de un periodo más amplio de tiempo, lo cual habilita a la detección de una activad concreto.

Una técnica ampliamente utilizada en el ámbito del Reconocimiento de la Actividad Humana, y que es el que se va a utilizar en el desarrollo de este proyecto, es la división de datos en ventanas temporales (Figura 6.1) [38].

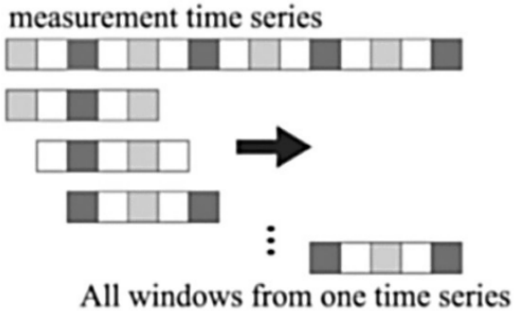

Figura 6.1: Concepto de división por ventanas

Una ventana es una agrupación de N muestras consecutivas, donde N vendrá definido por el tamaño de ventana. En este caso, no se define directamente el número de muestras, sino el intervalo temporal de la ventana. Estas ventanas se irán deslizando desde el comienzo de la muestra hasta el final, separando de esta manera todos los datos en ventanas. A su vez, las ventanas adyacentes se solaparan en un **porcentaje de solapamiento**, el cual es también un parámetro del preprocesamiento. De esta forma, dos ventanas contiguas compartirán una fracción de las mediciones contenidas. Por ejemplo, si se selecciona un tamaño de ventana de diez segundos, la primera ventana estará compuesta por las mediciones con timestamp del segundo cero al diez. Si el porcentaje de solapamiento es de cinco, entonces la segunda ventana serán las instancias del segundo cinco al quince, y así sucesivamente. De esta manera se tiene, por un lado, una evolución más suavizada de la señal entre ventanas, y por otro, se logra tener un número más alto de ellas para entrenar y probar el sistema. Estás ventanas serán los datos que sirvan como entrada a los modelos.

Para este primer experimento definiremos estos parámetros de la siguiente manera:

- **Tamaño de ventana**: 10 segundos. Basándonos en la experiencia del grupo y en otros estudios [39], este tamaño de ventana es el que mejores resultados da.
- **Porcentaje de solapamiento**: 50 % (5 segundos). Este es el solapamiento habitual en este tipo de sistemas [40].

Tras la división de los datos en ventanas, es necesario adecuarlos como entrada a los modelos de aprendizaje automático. Para ello, será necesario extraer una serie de características de cada uno de los ejes. En este caso, se extraerán características asociadas al **dominio del tiempo** y al **dominio de la frecuencia**.

Además de las medidas correspondientes a los ejes X, Y y Z, se incorporará una variable adicional: la magnitud o modulo del vector tridimensional (Ecuación 6.7). De esta manera, en cada ventana contaremos tanto con información individual de cada eje como con información integrada de los tres.

$$\mathrm{M} \;=\; \sqrt{x^2 \;+\; y^2 \;+\; z^2}\,. \tag{6.7}$$

Para caracterizar la señal en el dominio de frecuencia, aplicamos la Transformada Rápida de Fourier (FFT) a cada ventana de datos:

$$X[k] \;=\; \sum_{n=0}^{N-1} x[n]\, e^{-j\frac{2\pi k\, n}{N}} \tag{6.8}$$

Las características extraídas en cada ventana para los ejex X, Y, Z y M son las siguientes:

- **Dominio del tiempo**
  - **Media**: valor medio de la señal en la ventana.
  - **Mediana**: valor central ordenado de la señal.
  - **Máximo**: valor máximo observado.
  - **Mínimo**: valor mínimo observado.
  - **Desviación estándar**: dispersión de la señal alrededor de la media.
  - **Rango**: diferencia entre el máximo y el mínimo.
  - **Curtosis**: momento de orden 4 que indica la presencia de picos o colas pesadas.
  - **Percentil 25 / 75**: valores bajo los cuales se sitúan el 25 % / 75 % de los datos.
  - **Asimetría (skewness)**: grado de sesgo de la distribución de la señal.
  - **Período (autocorrelación)**: desfase donde la autocorrelación es máxima.
  - **Valor de autocorrelación**: magnitud de la autocorrelación en dicho desfase.
- **Dominio de la frecuencia**
  - **Frecuencia dominante 1**: frecuencia con mayor energía espectral.
  - **Amplitud dominante 1**: amplitud en la frecuencia dominante 1.
  - **Frecuencia dominante 2**: segunda frecuencia más energética.
  - **Amplitud dominante 2**: amplitud en la frecuencia dominante 2.
  - **AUC**: área bajo la curva espectral.
  - **Media de amplitudes**: valor medio de las amplitudes del espectro.
  - **Mediana de amplitudes**: valor central de las amplitudes.
  - **Desviación estándar de amplitudes**: dispersión de las amplitudes.
  - **Rango de amplitudes**: diferencia entre amplitud máxima y mínima.
  - **Curtosis de amplitudes**: momento 4 de la distribución de amplitudes.
  - **Percentil 25 / 75 de amplitudes**: cuartiles de la distribución de amplitudes.
  - **Asimetría de amplitudes**: sesgo de la distribución de amplitudes.

6.2.2. Creación y optimización de modelos

En el Experimento 1 se implementaron y ajustaron dos clasificadores de referencia:

Random Forest

Random Forest es un ensamble de N árboles de decisión entrenados mediante muestreo *bootstrap* y selección aleatoria de subconjuntos de características en cada partición. Para el clasificador Random Forest se emplearon los siguientes hiperparámetros:

- **n_estimators** = 200: número de árboles en el ensamble.
- **max_depth** = **None**: profundidad máxima ilimitada para cada árbol.
- **max_features** = **log2**: número de características considerado en cada división = $\log_2(p)$, donde p es el total de variables.
- **min_samples_split** = 2: número mínimo de muestras necesarias para dividir un nodo.
- **min_samples_leaf** = 2: número mínimo de muestras que debe tener cada hoja.

Dado que los modelos basados en árboles de decisión son invariantes a la escala de las características ya que internamente solo comparan órdenes de magnitud para decidir umbrales, no se aplicó normalización ni estandarización a los datos [41].

Support Vector Machine con kernel RBF

La SVM con función de base radial emplea el kernel para proyectar los datos en un espacio de alta dimensión donde se maximiza el margen de separación entre clases, pudiendo de esta manera aproximarse más a las fronteras no lineales que existen en el espacio de características.

Para el clasificador SVM-RBF se emplearon los siguientes hiperparámetros:

- **C** = 1.0: coeficiente de regularización, controla el equilibrio entre margen amplio y errores de clasificación.
- **kernel** = **'rbf'**: función de base radial para manejar separaciones no lineales.
- **gamma** = **'scale'**: coeficiente del kernel RBF, igual a $\frac{1}{\text{n\_features} \times \text{Var}(X)}$ por defecto.

Para garantizar que todas las características contribuyan de forma equilibrada al cálculo de distancias, se empleó estandarización (z-score) de cada variable antes del entrenamiento [42].

Clasificación Multiclase vs *Ensemble*

Como se estableció en el sprint inicial, se utilizarán dos estrategias distintas para el entrenamiento y la clasificación de cada modelo:

- **Multiclase**: el modelo tomará como clase las 18 actividades del conjunto de datos, y dará como salida la actividad predicha.
- ***Ensemble***: en este caso, la clasificación se hara mediante un conjunto de clasificadores binarios asociados a cada actividad del conjunto de datos. Para realizar esto se utilizó la estrategia *OneVsRest*, que consiste en lo siguiente:
  1. Para cada clase $c \in \{1, \ldots, C\}$ se entrena un clasificador binario M_c que distingue "clase c" frente a "no-clase c".
  2. En la fase de inferencia, cada M_c devuelve una puntuación o probabilidad $s_c(x)$ de que la muestra x pertenezca a la clase c.
  3. La etiqueta final se asigna como:

$$\hat{y} \;=\; \arg \max_{c \in \{1,\ldots,C\}} s_c(x).$$

6.2.3. Resultados y Análisis

En la Tabla 6.5 se muestran los resultados obtenidos. De lo mostrado en la tabla se puede concluir:

- La diferencia entre utilizar la clasificación Multiclase y la clasificación por *Ensemble* es muy pequeña, aunque en todos los casos la clasificación por *Ensemble* da resultados ligeramente mejores.
- SVM-RBF funciona mucho mejor con datos estandarizados, llegando a aumentar en un 10 % todas las métricas frente a su versión con los datos sin estandarizar.
- Ambos modelos obtienen resultados similares bajo las condiciones de este experimento.
- El modelo que ha obtenido mejores puntuaciones en todas las métricas es el Random Forest con clasificación por *Ensemble*, llegando a obtener un 73.9 % de tasa de aciertos.

Tabla 6.5: Resultados obtenidos en el experimento 1

Modelo	**Tasa Aciertos**	**Recall**	**Precisión**	**F1-Score**
RandomForest_Multiclase	0.7352	0.7352	0.7526	0.7340
RandomForest_Ensemble	**0.7390**	**0.7391**	**0.7550**	**0.7362**
SVMRadial_Multiclase	0.5806	0.5807	0.6016	0.5762
SVMRadial_Ensemble	0.6270	0.6266	0.6290	0.6117
SVMRadial_Multiclase_Standard	0.7276	0.7273	0.7440	0.7260
SVMRadial_Ensemble_Standard	0.7289	0.7288	0.7430	0.7263

Al hacer el desglose de tasa de aciertos por cada actividad, representado en la Figura 6.2, podemos observar que las actividades en las que los modelos se han confundido más veces son tanto estar sentado como las orientadas a las manos, en concreto a las relacionadas con comer. Esto se debe principalmente a la naturaleza caótica que tienen estas actividades. Cuando se está realizando una actividad que no está orientada a las manos como puede ser andar o correr, las manos se mueven de forma automática, llegando a trazar un patrón que puede ser detectable por el modelo. Sin embargo, cuando la actividad esta relacionada con el movimiento de las manos, esta suele presentar movimientos bruscos, sin un patrón definido, ya que al no estar moviendo el resto del cuerpo las manos se convierten en nuestra forma de expresión y se tienden a hacer muchos movimientos irregulares con ellas.

Esta teoría se refuerza cuando observamos la matriz de confusión del modelo RandomForest_Multiclase, representada en la Figura 6.3. Las filas indican la actividad real, mientras que las columnas indican la actividad predicha. Cada valor indica el número de veces que la actividad de la fila ha sido predicha como la actividad de la columna.

En este caso, los rectángulos azules están remarcando las actividades orientadas a las manos, las cuales habían obtenido un rendimiento muy bajo en todos los modelos. Si se observa la intersección entre ambos rectángulos, se puede observar que la principal razón del bajo rendimiento de los modelos para estas actividades es que, en muchas ocasiones, son incapaces de distinguir entre las distintas actividades de comer, acumulándose casi el 18 % de los errores totales del modelo en esa intersección.

El caso en el que más se puede ver es en la actividad de comer un sandwich, donde tan solo un 24.2 % de las veces se predijo correctamente la actividad, mientras que en un 55.93 % de las veces se le asigno erróneamente otra actividad de comer distinta.

Tras la extracción de estas conclusiones, el objetivo del siguiente experimento será investigar como solucionar este problema con las actividades de comer para mejorar el rendimiento de los modelos.

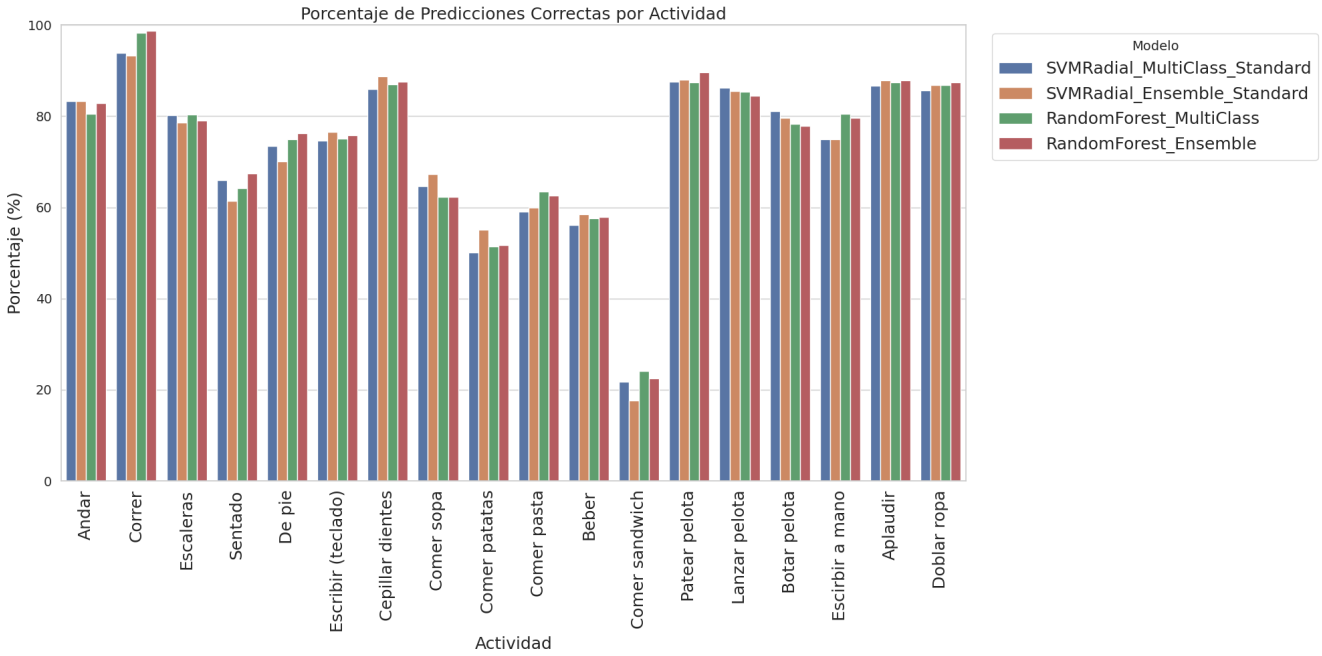

Figura 6.2: Tasa de aciertos por actividad en el experimento 1

6.3. Experimento 2: Ajuste de datos

Los resultados del Experimento 1 permitieron extraer la conclusión de que existía un problema entorno a la clasificación de las actividades orientadas a las manos relacionadas con comer, ya que, debido a la naturaleza caótica del movimiento de las manos durante estas actividades, los modelos tenían muchas dificultados a la hora de predecir que tipo de actividad relacionada con comer se estaba realizando.

Por consiguiente, este segundo experimento tratará de buscar una manera de solventar este problema, experimentando con distintas transformaciones de datos y analizando los resultados de los modelos.

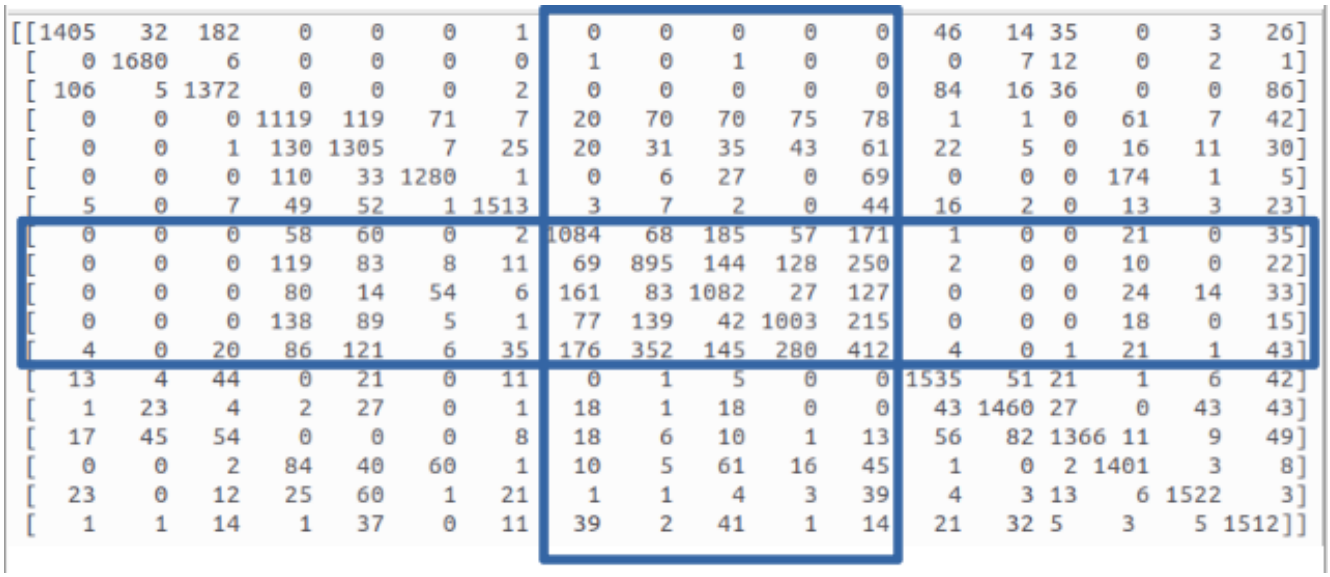

Figura 6.3: Matriz de confusión RandomForest_Multiclase en el experimento 1

6.3.1. Preparación de los datos

Con el objetivo de mejorar la eficiencia de los modelos, en este experimento se explorarán múltiples estrategias de tratamiento de datos.

Analizando el objetivo de este proyecto con los tutores, se llegó a la conclusión de que no era de alta importancia el poder predecir las distintas actividades de comer por separado. Por tanto, se optó por investigar las siguientes alternativas:

- **Eliminar los datos de las actividades de comer**: al no ser de alta relevancia para el proyecto, eliminarlas podría ayudar a mejorar la capacidad de predicción de los modelos en actividades que si consideran importantes. Este cambio, al reducir el número de ventanas, ayudaría a reducir el coste computacional de los modelos, el cual estaba siendo bastante elevado.
- **Agrupar las actividades en una sola actividad "Comer"**: esta solución permitiría seguir teniendo en cuenta todo el conjunto de datos, reduciendo los errores que se podían generar entre las diversas actividades de comer.

Eliminación de las actividades de comer

En este caso, la transformación de datos consistiría en descartar las ventanas las cuales tuviesen como código de actividad un código asociado a las actividades de comer. Como el conjunto de datos ya estaba balanceado desde un inicio, al eliminar estas actividades continuará estándolo, y por tanto no será necesario hacer ningún ajuste extra.

Agrupación de las actividades en una única actividad "Comer"

Esta alternativa consistiría en modificar el código de actividad de las ventanas con actividades de comer por uno nuevo el cual se asociaría con una nueva actividad "Comer". Al hacerlo, se generaría un nuevo problema resolver: **el conjunto de datos ya no está balanceado**.

Debido a esta agrupación de ventanas, aproximadamente un 25 % de las ventanas tienen la nueva actividad, haciendo que el conjunto de datos esté altamente desbalanceado, ya que el resto de actividades cuentan únicamente con aproximadamente un 5 % de las ventanas.

Para solucionar esto y volver a balancear los datos, se utilizó la técnica del **upsampling**, la cual duplicaría las instancias de de las clases minoritarias tantas veces como fuese necesarias hasta alcanzar el mismo número de instancias que el de la clase mayoritaria. Con este técnica el conjunto de datos volvió a estar balanceado, pero se aumentó considerablemente el número de ventanas del conjunto de datos (6.6), haciendo que, por consiguiente, también aumente el coste computacional del entrenamiento de los modelos.

Tabla 6.6: Comparativa del número de ventanas por transformación

Transformación	**Número de ventanas**
Conjunto de datos inicial	31781
Eliminar actividades comer	22972
Agrupar actividades comer	123326

6.3.2. Creación y optimización de modelos

Para este segundo experimento se utilizarán los modelos previamente diseñados en el Experimento 1, únicamente modificando la forma en la que se transformaron los datos.

6.3.3. Resultados y Análisis

En la Figura 6.4 se muestran los resultados con la opción de agrupar las actividades de comer en una única clase. Como se puede ver, con esta opción los modelos son capaces de reconocer dicha actividad sin ningún problema, llegando a ser la segunda actividad con mayor tasa de acierto. Sin embargo, vemos como hay otras actividades que continúan teniendo un rendimiento bajo, como pueden ser estar sentado o estar de pie.

Los resultados con la opción de eliminar las actividades de comer se pueden ver en la Figura 6.5. Estos resultados muestran como se ha aumentado la tasa de acierto en actividades que antes tenían malos resultados, como las mencionadas en el párrafo anterior. Esto hace que el modelo sea capaz de predecir todas las actividades de manera consistente, llegando casi al 80 % de tasa de aciertos en todas.

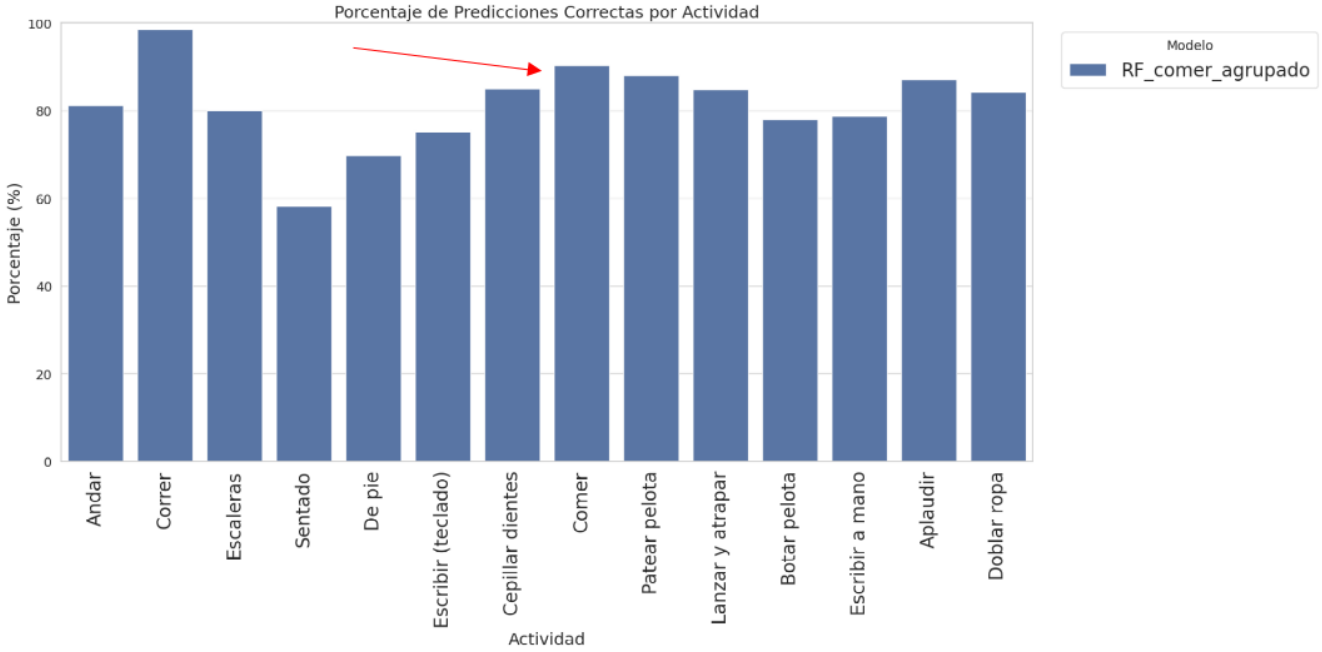

Figura 6.4: Tasa de acierto por actividad: Comer agrupado

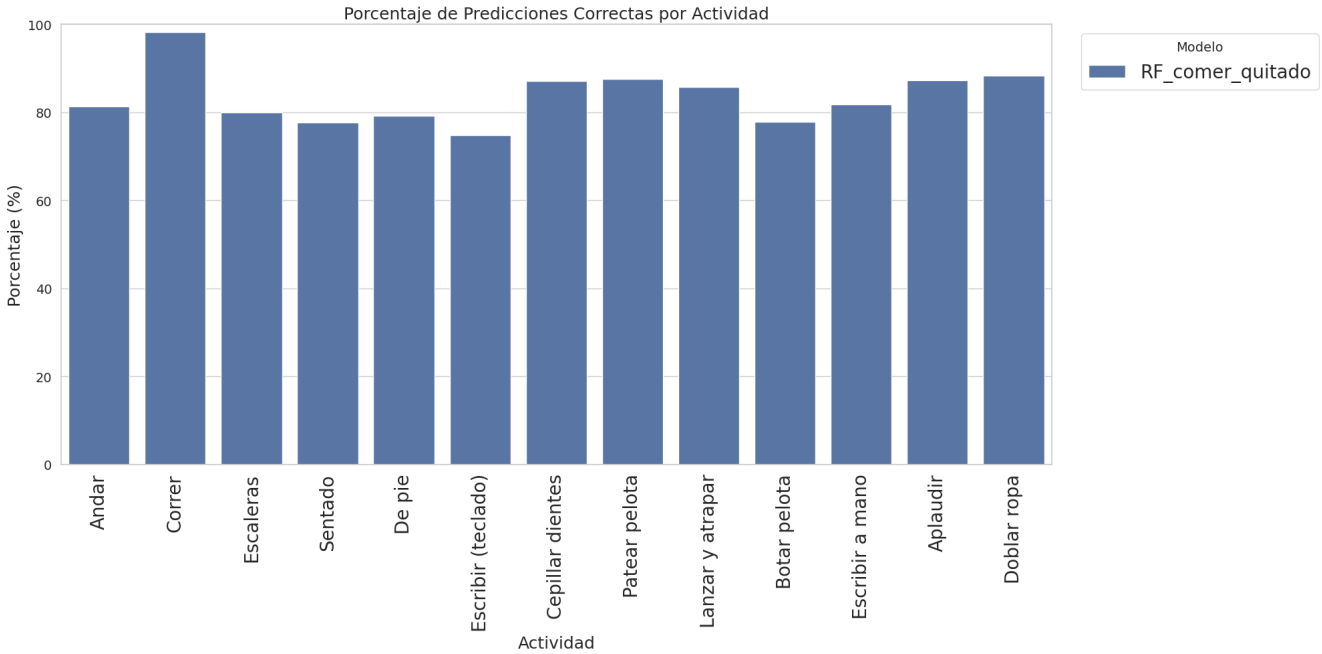

Figura 6.5: Tasa de acierto por actividad: Comer eliminado

Pros y contras de ambas transformaciones

- **Agrupar actividades comer**
  - Pros:
    - Permite mantener los mismos datos del conjunto de datos original.
    - Obtiene buenos resultados en la nueva actividad "Comer", mejorando considerablemente las métricas del modelo.
  - Contras:
    - Hay actividades que continúan teniendo una tasa de aciertos baja.
    - Al aumentar tanto el número de ventanas, se aumenta considerablemente el coste computacional de los modelos, haciendo que el entrenamiento de los mismos tarde más.
- **Eliminar actividades comer**
  - Pros:
    - Obtiene buenos resultados para todas las actividades del conjunto de datos.
    - Reduce el coste computacional de los modelos.
  - Contras:
    - Reduce el número de actividades que es capaz de reconocer el modelo.

Conociendo estos pros y contras, y teniendo en cuenta que el modelo que se va a desarrollar en el Experimento 3 tiene un coste computacional muy alto, se ha tomado la decisión de que, de ahora en adelante, **se eliminarán las actividades de comer**.

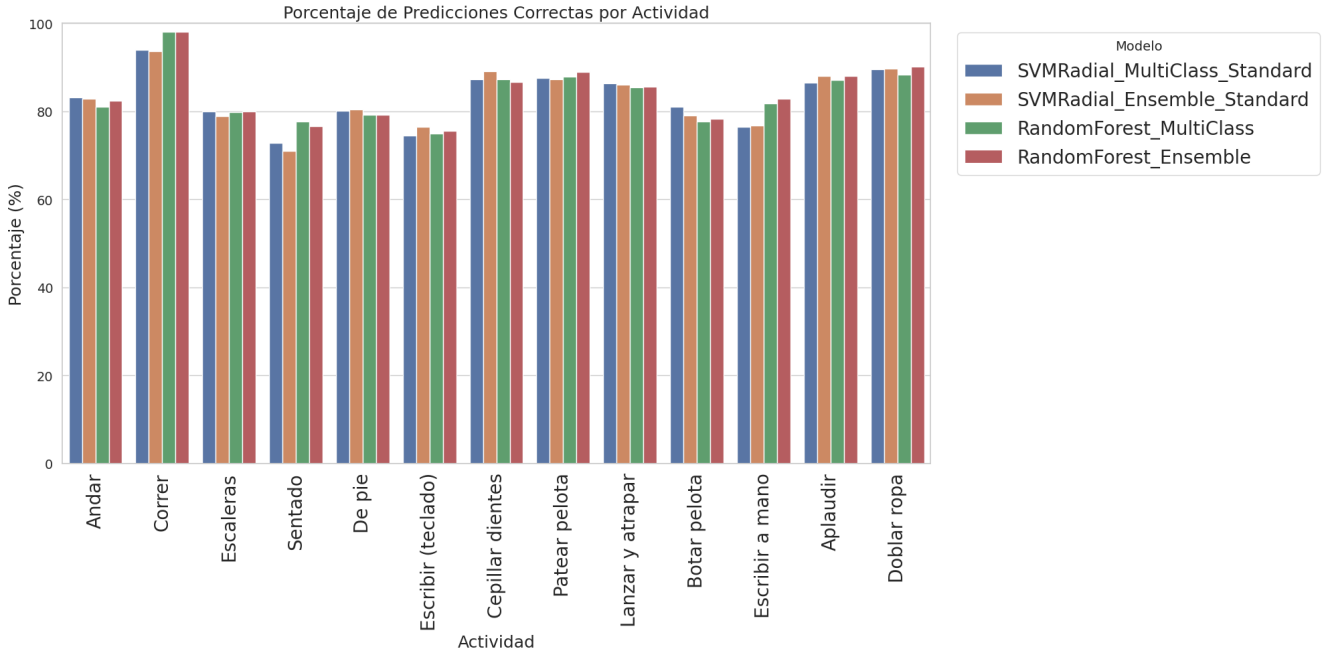

Figura 6.6: Tasa de acierto por actividad en el experimento 2

Como se puede ver en la Figura 6.6 y en la Tabla 6.7, los resultados han mejorado considerablemente con respecto al Experimento 1, llegando a **mejorar en un 10 % en valor absoluto todas las métricas.**

Por otro lado, se pueden sacar las mismas conclusiones que en el Experimento 1:

- Las estrategias MultiClase y *Ensemb* obtiene resultados similares, pero *Ensemb* obtiene resultados ligeramente superiores en todos los casos.
- RandomForest obtiene resultados ligeramente superiores a SVM-RBF

Tabla 6.7: Resultados obtenidos en los experimentos 1 y 2

Experimento	Modelo	Tasa Aciertos	Recall	Precisión	F1-Score
1	RandomForest_Multiclase	0.7352	0.7352	0.7526	0.7340
	RandomForest_Ensemble	0.7390	0.7391	0.7550	0.7362
	SVMRadial_Multiclase	0.5806	0.5807	0.6016	0.5762
	SVMRadial_Ensemble	0.6270	0.6266	0.6290	0.6117
	SVMRadial_Multiclase_Standard	0.7276	0.7273	0.7440	0.7260
	SVMRadial_Ensemble_Standard	0.7289	0.7288	0.7430	0.7263
2	RandomForest_Multiclase	0.8352	0.8355	0.8523	0.8364
	RandomForest_Ensemble	**0.8398**	**0.8400**	**0.8562**	**0.8405**
	SVMRadial_Multiclase_Standard	0.8300	0.8296	0.8509	0.8322
	SVMRadial_Ensemble_Standard	0.8298	0.8299	0.8476	0.8315

6.4. Experimento 3: Long Short-Term Memory

En los experimentos anteriores se han estado desarrollando modelos de *Machine Learning* clásicos, basados en árboles de decisión y en separación de instancias mediante margenes. Para este tercer experimento se va a utilizar un modelo de *Deep Learning*, en concreto, basado redes neuronales recurrentes, con el objetivo de estudiar que influencia tienen las dependencias temporales entre ventanas en los resultados del reconocimiento de actividad.

El modelo desarrollado será el conocido como **Long Short-Term Memory** (LSTM). Este modelo permite introducir una secuencia de ventanas contiguas en el tiempo y es capaz de mantener una memoria sobre los datos de las ventanas pasadas que influirá en el procesamiento de las ventanas futuras, siendo así capaz de hallar dependencias temporales que los modelos previamente desarrollados eran incapaces de encontrar.

6.4.1. Preparación de los datos

Hasta ahora los datos que se habían utilizado para entrenar los modelos estaban compuesto por arrays de las características correspondientes a una ventana de tiempo, haciendo que el modelo solo tuviese conocimiento de lo que ocurría en ese espacio temporal, completamente ajeno a las características de las ventanas contiguas.

Sin embargo, para poder introducir los datos dentro de un modelo LSTM es necesario añadir una dimensión extra: **el tiempo**. Por tanto, los datos de entrada ya no solo estarán constituidos por una ventana de características, sino que serán conformados por una matriz de ventanas contiguas, pudiendo de esta manera analizar como varían las características a lo largo del tiempo.

Para ello los parámetros para la extracción de características son los siguientes:

- **Tamaño de ventana**: 1 segundo

- **Porcentaje de solapamiento**: 0 %
- **Número de ventanas en cada matriz de entrada**: 10 ventanas.

De está manera, el modelo tendrá como entrada 10 segundos de actividad al igual que en los pasados experimentos, solo que esta vez las características estarán calculadas cada segundo y se tendrá en cuenta el instante temporal de la ventana.

6.4.2. Creación y optimización de modelos

La arquitectura del modelo utilizado en este experimento es la siguiente:

Tabla 6.8: Resumen de capas de la red LSTM

Nombre de capa	Tipo	Salida	Parámetros
input_layer_1	InputLayer	(None, 10, 100)	0
lstm_2	LSTM	(None, 10, 128)	117248
dropout_2	Dropout(0.5)	(None, 10, 128)	0
lstm_3	LSTM	(None, 64)	49408
dropout_3	Dropout(0.5)	(None, 64)	0
dense_2	Dense(32)	(None, 32)	2080
dense_3	Dense(13)	(None, 13)	429

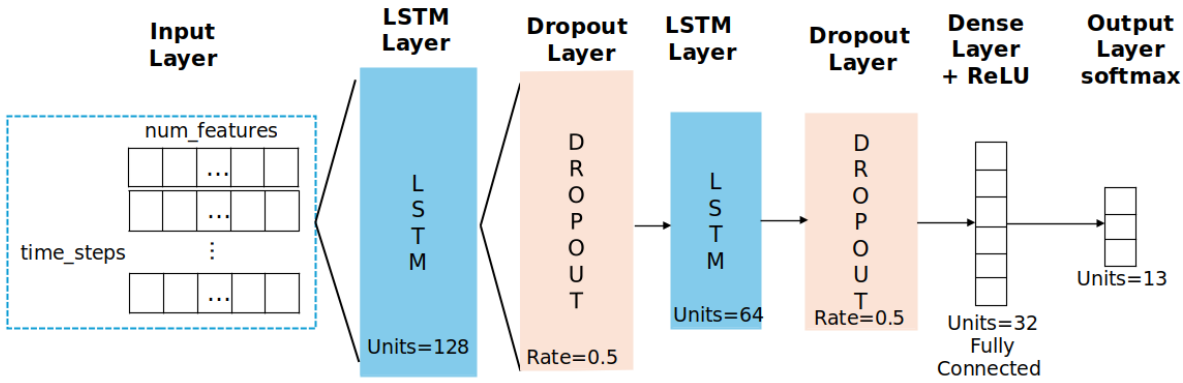

Figura 6.7: Arquitectura del modelo LSTM del experimento 3

Descripción detallada de cada bloque

En la Tabla 6.8 y la Figura 6.7 se puede observar los diferentes bloques de la arquitectura utilizada para el modelo LSTM. Las funciones y especificaciones de cada bloque son las siguientes:

- **InputLayer** Define la forma de entrada (batch, 10, 100). No tiene parámetros entrenables.
- **LSTM(128, return_sequences=True)** Procesa la secuencia de 10 vectores de 100 características, manteniendo la dimensión temporal en la salida (10, 128). Gracias a sus 128 unidades de memoria y puertas internas, captura dinámicas cortas y medias en la ventana. — *Parámetros:* 117 248.

- **Dropout(0.5)** Durante el entrenamiento, desactiva aleatoriamente un 50 % de las 128 salidas de cada paso temporal para evitar sobreajuste.
- **LSTM(64, return_sequences=False)** Recibe la secuencia anterior y devuelve solo el último estado oculto de 64 dimensiones. Así se condensa la información temporal relevante en un vector fijo. — *Parámetros:* 49 408.
- **Dropout(0.5)** Aplica de nuevo un 50 % de dropout al vector de 64 elementos para robustecer la capa siguiente.
- **Dense(32, activation=ReLU)** Capa totalmente conectada que reduce la dimensionalidad de 64 a 32, aplicando la función de activación ReLU para añadir no linealidad. — *Parámetros:* 2 080.
- **Dense(13, activation=softmax)** Capa de salida con 13 neuronas (una por cada actividad). La función softmax convierte los 32 valores previos en un vector de probabilidades $\mathbf{p} \in [0,1]^{13}$ que suman 1. — *Parámetros:* 429.

Extracción de la predicción

Para una muestra de prueba, el modelo devuelve el vector de probabilidades

$$\mathbf{p} = \big[p_1, p_2, \ldots, p_{13}\big], \quad \sum_{i=1}^{13} p_i = 1.$$

La etiqueta predicha $\hat{y}$ se obtiene como

$$\hat{y} \;=\; \arg \operatorname*{máx}_{i=1,\ldots,13} p_i. \tag{6.9}$$

Parámetros de entrenamiento

Algunos de los parámetros que se han definido para el entrenamiento del modelo son los siguientes:

- **Optimizador: Adam (learning_rate** $= 10^{-4}$**)** Adam es un optimizador estocástico que ajusta automáticamente la tasa de aprendizaje de cada parámetro mediante estimaciones de primer y segundo momento del gradiente, lo que favorece una convergencia rápida y estable en redes profundas.
- **Función de pérdida:** `sparse_categorical_crossentropy` Esta versión de entropía cruzada está diseñada para clasificación multiclase con etiquetas enteras y penaliza la discrepancia entre la distribución de probabilidades predicha y la clase real, maximizando así la probabilidad de la etiqueta correcta.
- **Batch size: 64** Define el número de muestras procesadas antes de cada actualización de pesos. Un tamaño de 64 equilibra la estabilidad de la estimación del gradiente y el aprovechamiento del paralelismo en GPU, reduciendo el ruido sin sacrificar velocidad de entrenamiento.
- **Número de epochs: 50** Es la cantidad de veces que el modelo recorre todo el conjunto de entrenamiento. 50 epochs permiten al modelo aprender patrones complejos sin caer en un sobreajuste excesivo, especialmente en combinación con las capas de Dropout y el optimizador Adam.

En este experimento, debido al alto coste computacional del modelo LSTM, tan solo se utilizará la estrategia Multiclase, descartando la estrategia de *Ensemble*.

6.4.3. Resultados y Análisis

Tabla 6.9: Resultados obtenidos en los experimentos 1, 2 y 3

Experimento	Modelo	Tasa Aciertos	Recall	Precisión	F1-Score
1	RandomForest_Multiclase	0.7352	0.7352	0.7526	0.7340
	RandomForest_Ensemble	0.7390	0.7391	0.7550	0.7362
	SVMRadial_Multiclase	0.5806	0.5807	0.6016	0.5762
	SVMRadial_Ensemble	0.6270	0.6266	0.6290	0.6117
	SVMRadial_Multiclase_Standard	0.7276	0.7273	0.7440	0.7260
	SVMRadial_Ensemble_Standard	0.7289	0.7288	0.7430	0.7263
2	RandomForest_Multiclase	0.8352	0.8355	0.8523	0.8364
	RandomForest_Ensemble	**0.8398**	**0.8400**	**0.8562**	**0.8405**
	SVMRadial_Multiclase_Standard	0.8300	0.8296	0.8509	0.8322
	SVMRadial_Ensemble_Standard	0.8298	0.8299	0.8476	0.8315
3	LSTM	0.8166	0.8245	0.8169	0.8182

En la Tabla 6.9 se pueden ver los resultados obtenidos por el LSTM comparado con los modelos de experimentos anteriores. El modelo LSTM ha logrado unos resultados muy buenos, en torno al 82 % en todas las métricas, pero aún así no ha logrado superar la eficiencia de los modelos empleados en los experimentos anteriores.

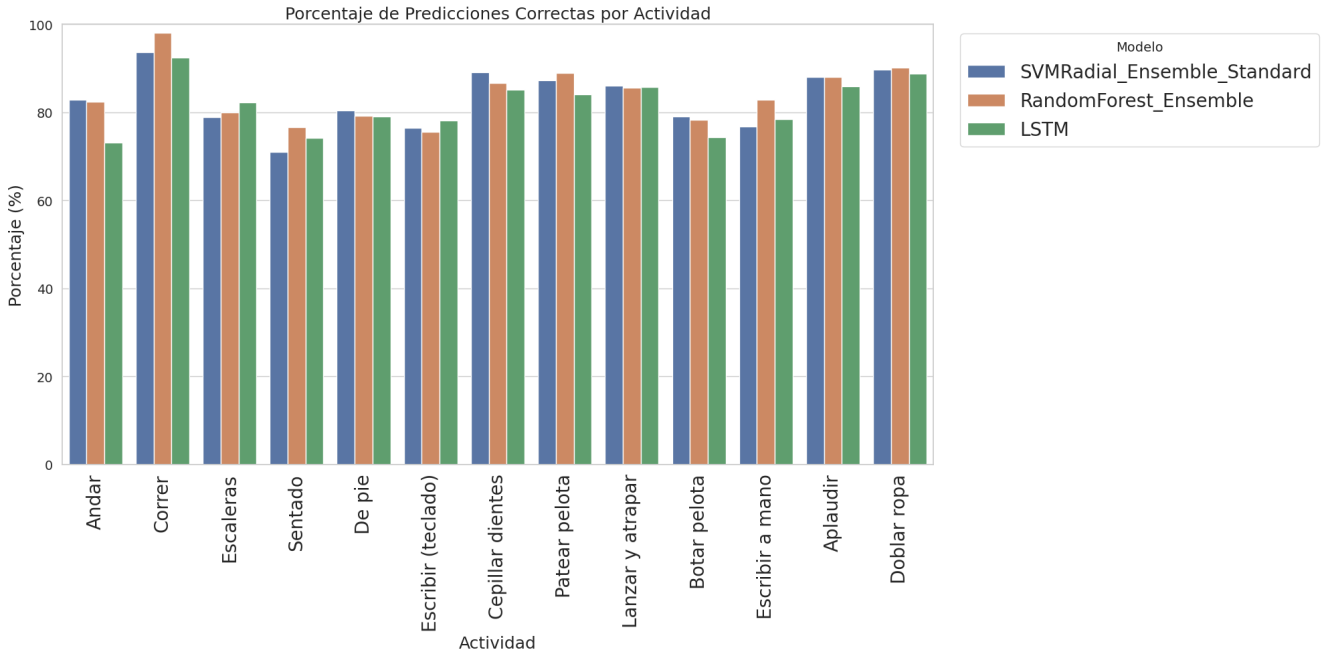

Figura 6.8: Tasa de acierto por actividad en el experimento 3

Por otro lado, analizando la Figura 6.8 se puede ver como los tres modelos obtienen muy buena tasa de aciertos para todas las actividades. Sin embargo, la diferencia en la tasa de acierto del modelo LSTM frente a los demás en actividades como por ejemplo "Andar" hace que las métricas de este modelo sean ligeramente inferiores.

Estas conclusiones no permiten afirmar que el modelo LSTM sea menos efectivo que el resto de modelos estudiados, sino que los resultados obtenidos por la arquitectura probada son inferiores. LSTM, al ser un

modelo basado en redes neuronales, tiene una alta capacidad de parametrización, pudiendo cambiar una gran cantidad de cosas para mejorar los resultados, desde cambios en los parámetros seleccionados hasta modificaciones en las diferentes capas de la arquitectura, pudiendo quitar, añadir o modificar tantas como sea necesario.

Gracias a la buena adaptación de este modelo se puede considerar que tiene potencial para seguir mejorando los resultados. Sin embargo, un inconveniente de este modelo es **su alto coste computacional** comparado con los otros modelos probados. Esto, junto con la limitación temporal de la carga de trabajo del Trabajo de Fin de Grado, hizo que se tuviera que descargar la opción de probar diferentes alternativas con LSTM. Se deja para trabajos futuros.

6.5. Experimento Final: Clasificador Binario de la actividad "Andar"

Este último experimento esta alineado con el interés particular del grupo de investigación de ser capaces de predecir cuando una persona está caminando. Previamente, otro miembro del grupo desarrolló un modelo, utilizando WISDM, para identificar a una persona en función de su forma de andar. Este desarrollo final serviría como intermediario entre los datos crudos y dicho modelo, ya que se utilizaría para detectar cuando los datos indican que la persona está caminando, y después se redirigirían al modelo de identificación biométrica de la persona.

6.5.1. Preparación de los datos

Hasta ahora, la clase se había dividido en las 13 actividades del conjunto de datos, sin embargo ahora lo único que nos interesa es predecir si una persona está caminando o no lo está haciendo. Por tanto la clase ahora tomará los siguientes valores:

- **1** si la actividad es andar
- **0** si la actividad no es andar

De esta manera, el modelo se centrará exclusivamente en predecir la actividad de caminar.

Sin embargo, como ya ocurrió cuando se estudió si se podían agrupar las actividades de comer, al juntar las clases de muchas actividades en una sola el conjunto de datos queda **altamente desbalanceado**, habiendo un 5.7 % de instancias de la clase positiva frente a un 94.3 % de la clase negativa.

Para solucionar esto, se empleará el método de **upsampling**, el cual duplicará las instancias de la clase minoritaria tantas veces como sea necesarias hasta igualar el número de instancias de la clase mayoritaria. De esta manera, el conjunto vuelve a estar balanceado.

6.5.2. Creación y optimización de modelos

Para este último experimento se mantuvieron idénticos los modelos de Random Forest y SVM (simplemente reetiquetando "caminar" como positivo y el resto como negativo). En el caso de la LSTM, sin embargo, se realizaron los siguientes cambios:

- La capa de salida pasó de

$$\texttt{Dense}(13,\ \text{activation} = \text{'softmax'}) \quad \longrightarrow \quad \texttt{Dense}(1,\ \text{activation} = \text{'sigmoid'})$$

de modo que el modelo emite una única probabilidad $p \in [0, 1]$ de que la actividad sea "caminar" (Figura 6.9).

- La función de pérdida se cambió de `sparse_categorical_crossentropy` a `binary_crossentropy`, adecuada para optimizar problemas de clasificación con dos clases y salida binaria.

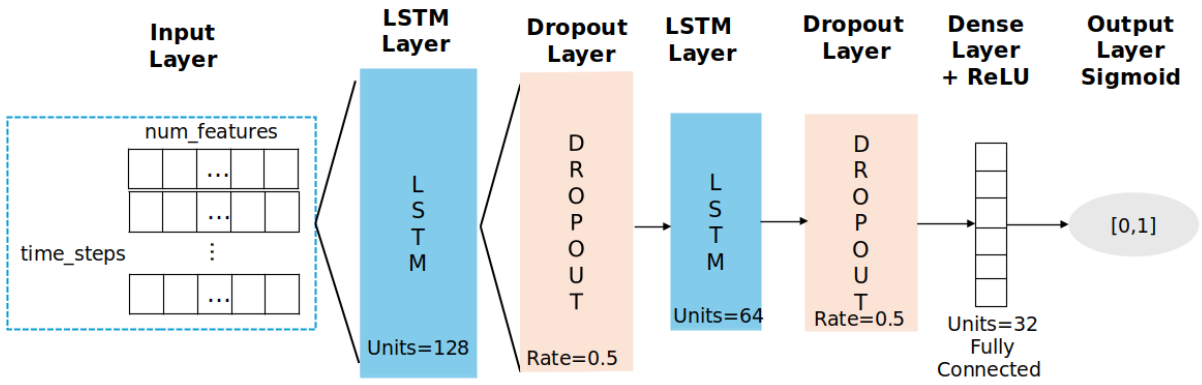

Figura 6.9: Arquitectura del modelo LSTM del Experimento Final

Como la clase es binaria, ya no será de utilidad utilizar la estrategia de *Ensemble*, así que solo se ejecutará la estrategia que denominamos Multiclase, aunque en este apartado ya no sería correcto denominarla así ya que unicamente hay dos clases, por lo que pasaría a denominarse **clasificación binaria**.

6.5.3. Resultados y Análisis

Tabla 6.10: Resultados obtenidos en el experimento final ("caminar" vs "no caminar")

Modelo	Tasa Acierto	Recall	Precisión	F1-Score
RandomForest_Binario	0.8498	0.8498	0.8904	0.8392
SVMRadial_Binario	**0.9196**	**0.9196**	**0.9346**	**0.9162**
LSTM_Binario	0.8047	0.8280	0.6090	0.7018

En la Tabla 6.10 se muestran los resultados obtenidos. Como se puede ver, el modelo SVM con kernel radial obtiene unos resultados muy buenos, llegando a tener aproximadamente un 92 % de tasa de acierto, recall y f1-score y más de un 93 % de precisión. Los de RandomForest, a pesar de ser inferiores, son también buenos. Sin embargo, el modelo LSTM obtiene unos resultados inferiores a los esperados, especialmente en la métrica de precisión, llegando apenas al 61 %.

En la Figura 6.10, se muestran los resultados por actividad. Como se puede observar, los tres modelos obtienen una alta tasa de acierto al predecir la clase positiva. Por otro lado, también se puede ver que la tasa de acierto de la clase negativa es considerablemente inferior a la de la clase positiva, siendo el modelo SVM con kernel radial el que mejores resultados obtiene, pudiendo así lograr unas muy buenas métricas finales.

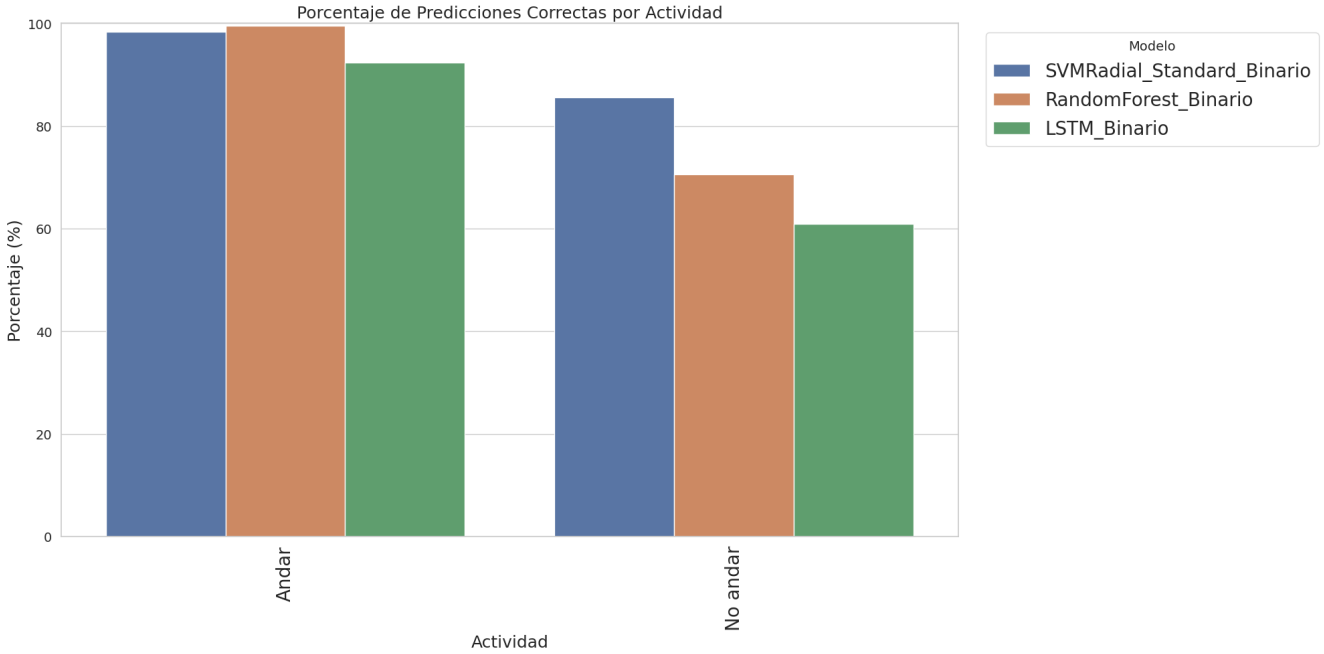

Figura 6.10: Tasa de acierto por actividad en el experimento final

6.6. Discusión Final

Las Tablas 6.9 y 6.10 muestran los resultados de los diferentes modelos desarrollados a lo largo del proyecto. Tras haber realizado todos los experimentos, se puede comentar lo siguiente:

- La estrategia *Ensemble* obtiene resultados ligeramente superiores a la estrategia Multiclase. Sin embargo, también tiene un coste computacional muchísimo más alto, por lo que sería necesario hacer un estudio de las limitaciones del equipo que se va a utilizar para decidir que estrategia es más adecuada.
- La estandarización de los datos aumenta de forma significativa la eficiencia del modelo SVM con kernel radial, mientras que para el modelo Random Forest, al estar basado en árboles de decisión, no se ve prácticamente afectado por la normalización.
- Para la clasificación de todas las actividades, Random Forest es el modelo que mejores resultados obtiene, además de tener un coste computacional considerablemente inferior al resto, por lo que se consolida como la mejor opción.
- En el caso de la clasificación binaria de la actividad "Andar", el modelo que mejores resultados obtuvo es el SVM con kernel radial, superando ampliamente al resto de modelos.
- El modelo LSTM tiene potencial para obtener buenos resultados, pero se necesita una gran capacidad computacional para poder probar todas las combinaciones de parámetros y arquitecturas posibles.

Capítulo 7

Conclusiones y líneas futuras de trabajo

En este Trabajo de Fin de Grado se han alcanzado los objetivos iniciales. Se ha desarrollado un flujo de procesamiento de señales que extrae características en los dominios temporal y frecuencial. También se han probado clasificadores clásicos (Random Forest y SVM-RBF) y un modelo neuronal LSTM. Por último, se ha evaluado su rendimiento tanto en el escenario multiclase como en el escenario binario ("caminar" vs. "no caminar").

7.1. Trabajos Futuros

Uso de diferentes modelos

Aunque Random Forest, SVM-RBF y LSTM constituyen tres enfoques representativos, basándose cada uno en una estrategia diferente para entrenar y predecir, existen muchos otros algoritmos por explorar. Como líneas futuras se proponen:

- **Redes neuronales convolucionales (CNN)** para extraer patrones locales en la serie temporal antes de la LSTM.
- **Transformers** o **Temporal Convolutional Networks (TCN)**, los cuales han mostrado gran eficacia en datos secuenciales en estudios anteriores [43].
- **Modelos híbridos** (CNN+LSTM) que combinen extracción automática de características con memoria de largo plazo [44].

Estudio de otras arquitecturas LSTM

El modelo LSTM presenta un gran potencial para obtener buenos resultados en el Reconocimiento de la Actividad Humana, por lo que se propone seguir estudiando este modelo o cualquier variación de redes neuronales recurrentes. Algunas de las alternativas que se ofrecen son las siguientes:

- Explorar variaciones de puertas: **GRU** (Gated Recurrent Unit) o **LSTM bidireccional**, que capturan contexto futuro y pasado.
- Ajustar profundidad y tamaño de las capas, así como técnicas de attention para enfocar la red en secciones relevantes de la secuencia.

- Incorporar **Batch Normalization** o **Layer Normalization** entre capas recurrentes para estabilizar el entrenamiento.

Aplicación a la identificación de usuarios

Un estudio en curso dentro del equipo de investigación en el cual se desarrolló este proyecto es el de la identificación de personas en función de su forma de caminar. Este proyecto serviría como intermediario entre los datos y el modelo de identificación, funcionando de la siguiente manera:

1. Se extraen las ventanas de datos con las características correspondientes.
2. Se utiliza el modelo de predicción de actividad para asignar una actividad a la ventana.
    - Si la predicción es "Andar", la ventana pasa a ser analizada por el modelo de identificación.
    - Si la predicción no es "Andar", se pasa a la siguiente ventana.

Integración en aplicaciones móviles

Para poder llevar está investigación al mundo real, una alternativa sería el de desarrollar una aplicación móvil que, mediante el uso de estos modelos, sea capaz predecir que actividad se está realizando en el momento y utilizar esta información con diferentes objetivos. Una de las posibles lineas de futuro es el de implementar los modelos en un sistema de **detección en tiempo real de actividades**. Diseñar una aplicación que sea capaz de detectar la actividad que se está realizando a tiempo real podría ser aplicada al campo de la seguridad y sanidad, pudiendo detectar actividades anómalas y enviar avisos en caso de inactividad o de caída. En este caso, se priorizaría la rapidez del modelo, así como el consumo energético, ya que al estar activo constantemente podría vaciar la batería del dispositivo rápidamente.

En conjunto, estas líneas futuras consolidan la base sentada en este TFG y abren la puerta a sistemas de monitorización personales más avanzados, capaces de combinar reconocimiento de actividades, identificación biométrica y generación de informes en entornos móviles.

Bibliografía

[1] Yuting Zhang, Gang Pan, Kui Jia, Minlong Lu, Yueming Wang, and Z. Wu. Accelerometer-based gait recognition by sparse representation of signature points with clusters. *IEEE transactions on cybernetics*, 45, 11 2014.

[2] Ananda Ravuri. A systematic literature review on human activity recognition. *Journal of Electrical Systems*, 20:1175–1191, 04 2024.

[3] Liming Chen and Chris D. Nugent. *Human Activity Recognition and Behaviour Analysis: For Cyber-Physical Systems in Smart Environments*. Springer International Publishing, 2019.

[4] J.L.R. Ortiz. *Smartphone-Based Human Activity Recognition*. Springer Theses. Springer International Publishing, 2016.

[5] Jennifer R Kwapisz, Gary M Weiss, and Samuel A Moore. Activity recognition using cell phone accelerometers. In *Proceedings of the Fourth International Workshop on Knowledge Discovery from Sensor Data (at KDD-10)*, 2010.

[6] Miguel A. Labrador and Oscar D. Lara Yejas. *Human Activity Recognition: Using Wearable Sensors and Smartphones*. Chapman and Hall/CRC, 1 edition, 2013.

[7] Sizhen Bian, Mengxi Liu, Bo Zhou, and Paul Lukowicz. The state-of-the-art sensing techniques in human activity recognition: A survey. *Sensors*, 22(12), 2022.

[8] Florenc Demrozi, Graziano Pravadelli, Azra Bihorac, and Parisa Rashidi. Human activity recognition using inertial, physiological and environmental sensors: A comprehensive survey. *IEEE Access: Practical Innovations, Open Solutions*, 8:210816–210836, 2020.

[9] Gérard Biau and Erwan Scornet. A random forest guided tour. *TEST*, 25, 11 2015.

[10] Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[11] Harmandeep Kaur, Veenu Rani, and Munish Kumar. Human activity recognition: A comprehensive review. *Expert Systems*, 41(11):e13680, 2024.

[12] Qi Teng, Kun Wang, Lei Zhang, and Jun He. The layer-wise training convolutional neural networks using local loss for sensor-based human activity recognition. *IEEE Sensors Journal*, 20(13):7265–7274, 2020.

[13] Shibo Zhang, Yaxuan Li, Shen Zhang, Farzad Shahabi, Stephen Xia, Yu Deng, and Nabil Alshurafa. Deep learning in human activity recognition with wearable sensors: A review on advances. *Sensors*, 22(4), 2022.

[14] N.T. Newaz and E. Hanada. The methods of fall detection: A literature review. *Sensors*, 23(11):5212, 2023.

[15] José M. Alcalá, Jesús Ureña, Álvaro Hernández, and David Gualda. Assessing human activity in elderly people using non-intrusive load monitoring. *Sensors*, 17(2), 2017.

[16] Chhavi Dhiman and Dinesh Kumar Vishwakarma. A review of state-of-the-art techniques for abnormal human activity recognition. *Engineering Applications of Artificial Intelligence*, 77:21–45, 2019.

[17] Alberto Carrera-Rivera, Daniel Reguera-Bakhache, Felix Larrinaga, et al. Structured dataset of human-machine interactions enabling adaptive user interfaces. *Scientific Data*, 10:831, 2023.

[18] Suphachai Mekruksavanich, Wasawat Phaphan, Nopparat Hnoohom, and Aksorn Jitpattanakul. Recognition of sports and daily activities through deep learning and convolutional block attention. *PeerJ Computer Science*, 10:e2100, 2024.

[19] Haotian Zhou, Xiujun Zhang, Yu Feng, Tongda Zhang, and Lijuan Xiong. Efficient human activity recognition on edge devices using deepconv lstm architectures. *Scientific Reports*, 15:13830, 2025.

[20] H. Sharen, L. Jani Anbarasi, P. Rukmani, Amir H. Gandomi, R. Neeraja, and Modigari Narendra. Wisnet: A deep neural network based human activity recognition system. *Expert Systems with Applications*, 258:124999, 2024.

[21] Shaik Jameer and Hussain Syed. A dcnn-lstm based human activity recognition by mobile and wearable sensor networks. *Alexandria Engineering Journal*, 80:542–552, 2023.

[22] Morsheda Akter, Shafew Ansary, Md. Al-Masrur Khan, and Dongwan Kim. Human activity recognition using attention-mechanism-based deep learning feature combination. *Sensors*, 23(12), 2023.

[23] Rüdiger Wirth and Jochen Hipp. Crisp-dm: Towards a standard process model for data mining. *Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining*, 2000.

[24] Ken Schwaber and Jeff Sutherland. *The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game*. Scrum.org, 2020.

[25] Mario Molina and Daniel Torres. Implementación de scrum como metodología ágil en proyectos de desarrollo de software. *Revista Científica y Tecnológica UPSE*, 2014.

[26] Scrum Alliance. Scrum artifacts: How they help teams deliver value, 2024. Consultado el 12 de junio de 2025.

[27] Scrum Alliance. Product goals in scrum: What they are and why they matter, 2024. Consultado el 12 de junio de 2025.

[28] Scrum Alliance. Scrum team roles and responsibilities, 2024. Consultado el 12 de junio de 2025.

[29] Scrum Alliance. 7 skills you need to be a great product owner, 2024. Consultado el 12 de junio de 2025.

[30] Scrum Alliance. High-performance teams: Why the 'who' matters less, 2024. Consultado el 12 de junio de 2025.

[31] Scrum Alliance. A day in the life of a scrum master, 2024. Consultado el 12 de junio de 2025.

[32] INCIBE. Análisis de riesgos en 5 pasos sencillos, 2023. Consultado el 12 de junio de 2025.

[33] Indeed. Salario medio de data scientist en valladolid, valladolid provincia, 2025. Consultado en junio de 2025.

[34] Jeffrey W. Lockhart, Gary M. Weiss, Jack C. Xue, Shaun T. Gallagher, Andrew B. Grosner, and Tony T. Pulickal. Design considerations for the wisdm smart phone-based sensor mining architecture. In *Proceedings of the Fifth International Workshop on Knowledge Discovery from Sensor Data (at KDD-11)*, San Diego, CA, 2011.

[35] Kishor Walse, Rajiv Dharaskar, and V. M. Thakare. Performance evaluation of classifiers on wisdm dataset for human activity recognition. 03 2016.

[36] D. Bhattacharya and et al. Optimizing machine learning models in human activity recognition. *Tuijin Jishu / Journal of Propulsion Technology*, 44(5), 2023.

[37] Abdulmajid Murad and Jae-Young Pyun. Deep recurrent neural networks for human activity recognition. *Sensors*, 17(11):10–11, 2017.

[38] Jiri Klema, Lenka Vyslouzilova, Filip Karel, Olga Štepánková, and Filip Zelezný. Sequential data mining: A comparative case study in development of atherosclerosis risk factors. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 38:3 – 15, 02 2008.

[39] Anzah Niazi, Delaram Yazdansepas, Jennifer Gay, Frederick Maier, Lakshmish Ramaswamy, Khaled Rasheed, and Matthew Buman. Statistical analysis of window sizes and sampling rates in human activity recognition. pages 319–325, 01 2017.

[40] Akbar Dehghani, Omid Sarbishei, Tristan Glatard, and Emad Shihab. A quantitative comparison of overlapping and non-overlapping sliding windows for human activity recognition using inertial sensors. *Sensors*, 19(22), 2019.

[41] Forecastegy. Do Decision Trees Need Feature Scaling Or Normalization? `https://forecastegy.com/posts/do-decision-trees-need-feature-scaling-or-normalization/`, June 2025. [Online; accessed 8-June-2025].

[42] Forecastegy. Does SVM Need Feature Scaling Or Normalization? `https://forecastegy.com/posts/does-svm-need-feature-scaling-or-normalization/`, June 2025. [Online; accessed 8-June-2025].

[43] Ramiro Casal, Leandro E. Di Persia, and Gastón Schlotthauer. Temporal convolutional networks and transformers for classifying the sleep stage in awake or asleep using pulse oximetry signals. *Journal of Computational Science*, 59:101544, 2022.

[44] Adarsh Muralidharan and Sazia Mahfuz. Human activity recognition using hybrid cnn-rnn architecture. *Procedia Computer Science*, 257:336–343, 2025. The 16th International Conference on Ambient Systems, Networks and Technologies Networks (ANT)/ the 8th International Conference on Emerging Data and Industry 4.0 (EDI40).