Universitat Oberta de Catalunya

uoc.edu

Detección de caídas con reloj inteligente (*smartwatch*) para personas mayores.

Arnau Tienda Tejedo
Máster en Ciencia de Datos
Ciencia de datos aplicada a la salud

Tutora: Susana Pérez Álvarez
Profesora responsable: Àngels Rius Gavidia

10/01/2021

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Detección de caídas con reloj inteligente (smartwatch) para personas mayores*
Nombre del autor:	*Arnau Tienda Tejedo*
Nombre del consultor/a:	*Susana Pérez Álvarez*
Nombre del PRA:	*Àngels Rius Gavidia*
Fecha de entrega (mm/aaaa):	01/2021
Titulación:	*Máster en Ciencia de datos*
Área del Trabajo Final:	*Ciencia de datos aplicada a la salud*
Idioma del trabajo:	*Castellano*
Palabras clave	*PPG, Fall Detection, Atrial Fibrillation*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

El tanto por ciento de personas mayores de 65 años en España aumenta año tras año. Gran parte de este colectivo vive solo en sus casas, hecho que hace que las familias se preocupen por su estado de salud al no recibir actualizaciones continuas por parte de, por ejemplo, un cuidador.

La finalidad de este proyecto es analizar la viabilidad de crear un reloj inteligente con dos funcionalidades: detección de caídas, y análisis de fibrilación auricular, el tipo de arritmia más común y la que provoca más infartos de miocardio.

A nivel de datos de acelerómetro, se han podido encontrar suficientes datos etiquetados, es decir, lecturas de acelerómetro en las que se indican la actividad que se está realizando. Además, es factible crear datos propios a partir del reloj inteligente realizando acciones y capturándolas, como para llegar a tener un buen modelo que clasifique caídas.

Se ha realizado el entrenamiento de 4 métodos de aprendizaje supervisado, y se ha conseguido obtener una detección de caídas con una sensibilidad muy alta, clasificando de manera correcta hasta un 96% de las caídas.

Por otro lado, no se han podido encontrar suficientes datos de PPG (fotopletismografía) de personas sanas y personas con arritmias. La generación de estos datos a partir de nuestro reloj tampoco es fácil al tener que contar con

gente con esta patología. Por ello, resulta más difícil crear un modelo que pueda predecir fibrilación auricular, por falta de datos.

Abstract (in English, 250 words or less):

The percentage of people over 65 years old in Spain increases year after year. A significant part of this group lives alone in their homes, a fact that makes families worry about their health condition as they do not receive continuous feedback from, for example, a caregiver.

The purpose of this project is to analyze the viability of creating an smartwatch with two functions: detection of falls, and analysis of atrial fibrillation, the most common type of arrhythmia and the one that causes more strokes.

Regarding accelerometer data, it has been possible to find enough labeled data, that is, accelerometer readings that indicate the activity being performed. In addition, it is feasible to create your own data from the smartwatch by performing actions and capturing them, as to get a good model that classifies falls.

We have trained 4 supervised learning methods, and we have managed to obtain a fall detection with a very high sensitivity, classifying correctly up to 96% of the falls.

On the other hand, it has not been possible to find enough PPG (photoplethysmography) data of healthy people and people with arrhythmias. The generation of this data from our watch is not easy either, since we have to count on people with this pathology. Therefore, it is more difficult to create a model that can predict atrial fibrillation, due to lack of data.

Índice

Lista de figuras

1. Introducción

1.1. Contexto y justificación del Trabajo

1.1.1. Introducción

Existe un gran número de preguntas y cuestiones sobre cómo se debe controlar el estado de salud de las personas que viven solas o son dependientes. El tener la capacidad de monitorizar las constantes vitales de cada individuo permitiría el poder hacer un seguimiento continuo sobre el desarrollo del estado de salud de estos.

En el caso de los domicilios particulares, la principal necesidad es el control de la persona a tiempo real. Al tratarse de una persona mayor, en muchos casos dependiente, puede haber complicaciones de salud y al no darse cuenta, puede no haber la posibilidad de recurrir a la asistencia médica necesaria.

El número de personas mayores de 65 años en España y su previsión para los próximos años es el siguiente:

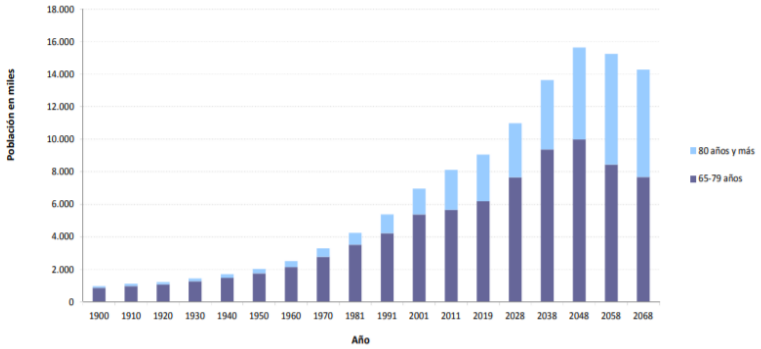


Figura 1: Evolución de la población de 65 años y más en España. 1900 – 2068 (CSIC 2018)

Como puede observarse, hay un aumento considerable año tras año de la población de más de 65 años, por lo que podemos asegurar que tendremos más usuarios potenciales a medida que avanza el tiempo.

La distribución según su género es la siguiente:

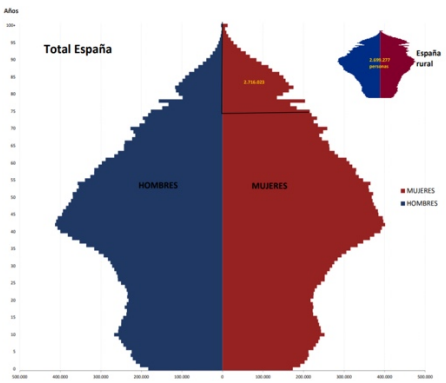


Figura 2: Pirámides de población de España y de la España rural (comparación) (CSIC s.f.)

Los últimos datos que se tienen sobre el número de usuarios de tele asistencia es el siguiente: 769.336 personas. Es decir, un 8,89% del total de personas mayores. Además, se trata de un servicio altamente feminizado dado que el 76% son mujeres y con un elevado porcentaje de personas usuarias mayores de 80 años (67%). (Gobierno de España 2015).

Detección de caídas

Las caídas en gente mayor son un problema de salud importante. Existen dos tipos de métodos que pueden servir para evitar su impacto.

Por un lado, se pueden implementar métodos preventivos a partir de la lectura de constantes vitales y análisis de comportamiento junto con datos de acelerómetro (para tener en cuenta, por ejemplo, el equilibrio de esa persona) que tratados de manera correcta pueden llegar a predecir cuando se va a producir una caída. Se trataría de un método predictivo.

Por otro lado, los métodos detectivos permiten reconocer una caída en el momento de producirse, de modo que el tiempo de respuesta entre la caída y la acción siguiente (llamar al 112, acudir en su ayuda, etc) sea lo más rápido posible. **En este trabajo se explorará está segunda rama, la de los métodos detectivos.**

Detección de arrítmias

El *smartwatch* utilizado cuenta con un sensor de fotoplestimografía. Este tipo de sensores mide el cambio de volumen de sangre a través de un fotoemisor y un fotodetector, midiendo diferencias entre la luz emitida y la luz recibida. A partir de estos datos, se puede realizar un estudio sobre varios signos vitales del individuo (saturación de oxígeno en sangre, ratio de respiración, latidos por minuto, fibrilación auricular…). (Allen 2007)

Este proyecto se centrará en la fibrilación articular. Se trata de un tipo de arritmia que comúnmente puede causar ictus (Kirstin, y otros 2020). Adicionalmente, puede no tener símptomas aparentes. Por ello, se estudiará como poder detectarlo a partir de la lectura del sensor de fotoplestimografía (PPG).

1.1.2. Motivación personal

A nivel de motivación personal, se trata de un proyecto que da continuidad a un proyecto que llevo desarrollando desde hace meses en una *start-up* creada con compañeros de diferentes disciplinas. Se pretende crear un servicio para que las familias tengan tranquilidad sobre el estado de salud de sus seres queridos a partir de este dispositivo “wearable”.

Este trabajo se basa únicamente en el análisis de datos, pero estos datos parten del trabajo que están realizando mis compañeros, por ejemplo, con

la programación del reloj para que se realice la lectura de los datos del acelerómetro en crudo.

El hecho de que sea un proyecto completo y con un objetivo final (poder comercializar este servicio) agrega mucha importancia a que este trabajo salga bien y tenga valor.

Además, trabajar para mejorar la vida de las personas mayores, y que sus familiares tengan más tranquilidad sobre su estado de salud aporta un valor añadido a este proyecto.

1.2. Objetivos del Trabajo

El objetivo principal del trabajo es realizar un sistema de monitorización que sea capaz de detectar caídas con una gran precisión, evitando al máximo el número de falsos positivos en actividades diarias. Es decir, el sistema tendrá que alertar cuando el individuo que lleve el reloj se caiga, y no alertar si no ha habido ninguna caída.

Como objetivo secundario, se estudiará la viabilidad de detectar FA (Fibrilación Auricular) a partir de los datos de PPG del reloj.

1.3. Enfoque y método seguido

A nivel general, el proyecto seguirá la metodología CRISP-DM. Se trata de un método muy maduro para proyectos de Data Science, el cual consta de 6 fases (IBM s.f.).

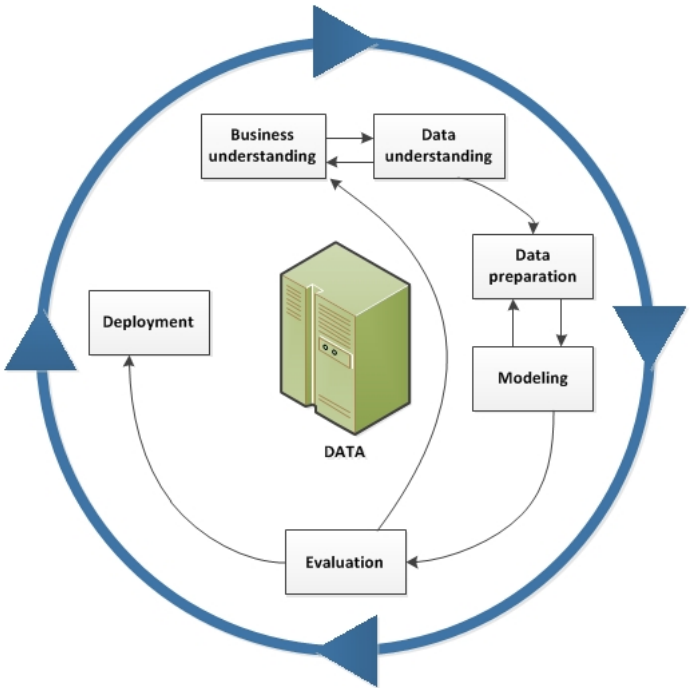


Figura 3: Metodología CRISP-DM

Business understanding
Para empezar, se estudiará bibliografía sobre detección de caídas y FA.

Data understanding
Para realizar la detección de caídas, se utilizarán datos etiquetados públicos sobre lecturas de acelerómetros situados en la muñeca del usuario, con tal de parecerse al máximo a las lecturas que puede generar nuestro *smartwatch*.

También se estudiará la viabilidad de realizar una detección de fibrilación auricular a partir de PPG, mediante la búsqueda de estudios científicos relacionados, y la de conjuntos de datos etiquetados con los datos de personas con este tipo de arritmias.

Data preparation
En el caso de no encontrarse datos de suficiente calidad, se realizará la creación de un conjunto de datos propio, utilizando diferentes sujetos y registrando acciones diarias de todo tipo, y modalidades distintas de caída. Si ya los tenemos, se realizará la carga y limpieza de datos.

Modeling
Se crearán y evaluarán varios modelos que serán entrenados por los datos conseguidos en apartados anteriores.

Evaluation
Se evaluarán los modelos y los resultados obtenidos para ver con cual se obtiene una clasificación de más calidad.

Deployment
Finalmente, se lanzará una versión estable del código, con el mejor modelo obtenido para predecir caídas y FA.

Toda la programación se realizará con Python, al ser el lenguaje más extendido a nivel de *Data Science*, y disponer de varias librerías que facilitan el trabajo. Adicionalmente, para el control de versiones se utilizará la plataforma Github.

A nivel de entorno de desarrollo, se utilizará Spyder de Anaconda. Se trata de un IDE (*Integrated Development Environment*) muy completo que dispone de explorador de variables y ejecución por lotes de código entre otras funcionalidades.

1.4. Planificación del Trabajo

La planificación de este trabajo se basa en los entregable que se proponen para cada PAC desde el aula de la asignatura. Se trata de los siguientes:

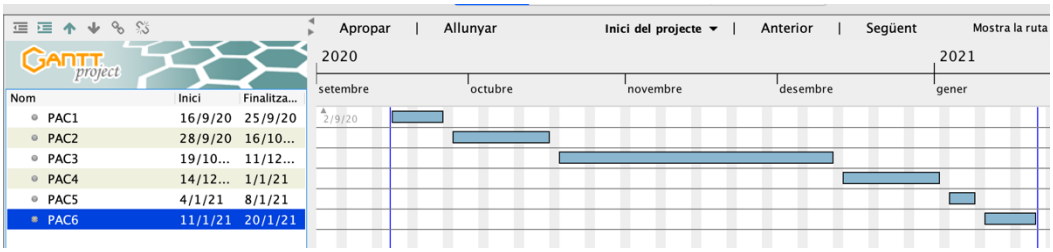


Figura 4: Diagrama de gantt del proyecto

Definición de objetivos y alcance

Se realiza una propuesta de como será el trabajo final. Se definen los objetivos, la motivación del proyecto, la metodología que se seguirá y la planificación. Es una parte importante del proyecto ya que marcará la pauta para el alcance del proyecto.

Estado del arte

Se realiza un estudio del estado del arte. En nuestro caso, tenemos que realizar dos estudios, uno sobre detección de caídas, y el otro sobre detección de FA mediante PPG. Así pues, podemos desglosar las siguientes tareas:

- Familiarización con la detección de caídas mediante lectura de datos de acelerómetro (X,Y,Z).
- Familiarización con la detección de Fibrilación Articular mediante datos PPG.
- Búsqueda de dispositivos que realicen funciones similares, y estudio sobre sus funcionalidades.

Diseño e implementación

Este apartado constituye gran parte del proyecto, ya que contiene el diseño, desarrollo e implementación del producto. La dividiremos de la siguiente manera:

Preparación de datos:

- Búsqueda de conjuntos de datos de lecturas de acelerómetro y PPG etiquetadas.
- Carga de datos.
- Exploración de datos.
- Limpieza de datos
- Integración (en el caso de haber más de un conjunto de datos de fuentes diferentes)
- Análisis descriptivo de los datos.

Modelado (detección de caídas)

- Selección de características.

- Selección de la ventana de tiempo a analizar.
- Selección de posibles modelos de clasificación.
- Construcción de los modelos
- Evaluación de la calidad de los modelos

Redacción de la memoria

En este punto se dispondrá de los resultados de todo el proyecto, por lo que el siguiente paso será el redactado de la memoria. Se realizará de forma que siga el hilo de las tareas que se han ido haciendo, y documentando en cada punto el por qué de cada decisión tomada.

Presentación y defensa pública

Finalmente, se entregará y presentará el proyecto ante un tribunal.

1.5. Breve sumario de productos obtenidos

El producto final será un código que dados unos segundos de datos de acelerómetro proporcionados por un *smartwatch* sea capaz de indicar si se trata de una caída o si se trata de cualquier otra actividad del día a día.

Adicionalmente, se presentará un estudio sobre como utiliza el PPG proporcionado por el reloj para poder llegar a predecir arritmias de tipo fibrilación auricular.

1.6. Breve descripción de los otros capítulos de la memoria

- **Estado del arte**: análisis de otros proyectos similares y su resolución.
- **Diseño e implementación**: preparación de los datos usados, y construcción y evaluación de los modelos de clasificación.
- **Resultados**: análisis de medidas utilizadas para calificar la calidad de los modelos.
- **Conclusiones**: revisión de objetivos del proyecto y aprendizajes obtenidos tras su realización.
- **Glosario:** Acrónimos y siglas utilizadas en la memoria.
- **Bibliografía:** documentos utilizados durante el proyecto.

2. Estado del arte

2.1. Introducción

A nivel de investigaciones realizadas sobre detección de caídas, existen básicamente tres tipos de datos que permiten clasificar este tipo de eventos:

- **Acelerómetro**, datos recogidos, por ejemplo, en una pulsera o cinturón colocados en el individuo.
- **Sensores ambiente**, como podría ser detección por infrarrojos o sensores de vibración.
- **Dispositivos de visión**, como cámaras de vigilancia.

Los datos de los que se dispondrán a partir de nuestro reloj inteligente serán datos de acelerómetro, por lo que nos interesa buscar conjuntos de datos que contengan esta tipología.

Para seguir la nomenclatura que se utiliza en las publicaciones en inglés, se usarán las siglas ADL (Activities of Daily Life) para hablar de datos de acelerómetro correspondientes a actividades del día a día.

Conjunto de datos	Sensores utilizados	Modelos
UMAFall	4 sensores (acelerómetros y giroscopios) en 4 posiciones (tobillo, cadera, pecho y muñeca).	Umbral.
UCI Dataset	3 sensores (acelerómetro, giroscopio y magnetómetro MTx de Xsens colocados en 6 posiciones: tobillo, cadera, pecho, muñeca, muslo y cabeza.	KNN (K-Nearest Neighbors), SVM (Support Vector Machines), BDM (Bayesian Decision Making), DTW (Dynamic, ANN.
SisFall Dataset	1 unidad de sensores situada en la cadera, que consta de 2 acelerómetros y un giroscopio.	Únicamente se crea el dataset, no se analiza.
UNIVPM Dataset	1 unidad de sensores situada en la cadera, que consta de 1 acelerómetro, un magnetómetro, y 1 giroscopio.	Umbral.
UP-Fall	3 sensores (acelerómetro, giroscopio y magnetómetro colocados en 6 posiciones: tobillo, cadera, pecho, muñeca, bolsillo del pantalón y cuello.	RF (Random Forest), SVM (Support Vector Machines), MLP (Multi-Layer Perceptron), KNN (K-Nearest Neighbors).

Tabla 1: Resumen tratamiento de datos de cada investigación

Estos son los estudios más relevantes que se pueden encontrar a nivel de detección de caídas. Son interesantes ya que cada uno utiliza sensores diferentes, en distintas posiciones del cuerpo con tal de obtener un mejor modelado para la detección de caídas. Adicionalmente, se usan varios modelos de aprendizaje automático con tal de obtener la mayor precisión posible.

En relación a la detección de fibrilación auricular, cada vez más son los estudios que intentan definir una manera de detectarla a partir de algún método no invasivo.

En los siguientes apartados se darán más detalles sobre los estudios que se han tenido en cuenta para el desarrollo de este proyecto.

2.2. Detección de caídas

UMAFall (Eduardo, Ramon y Garcia 2017)

Conjunto de datos creado por investigadores de la Universidad de Málaga en el que se registran los datos de acelerómetro en 4 posiciones distintas del cuerpo (muñeca, tobillo, cintura y pecho). Su objetivo era descubrir en qué posición del cuerpo se debe colocar el sensor para obtener la máxima precisión.

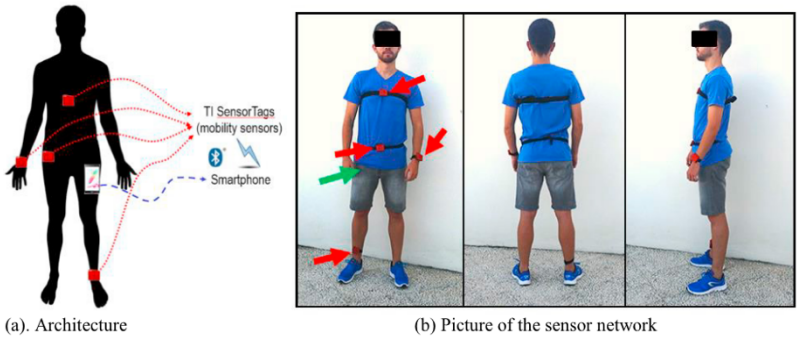


Figura 5: Colocación de los sensores en UMAFall

Un total de 17 usuarios realizando 8 actividades diarias y 3 tipos de caídas. Cada uno de los movimientos fue replicado 3 veces por persona.

SisFall (Angela, Jose David y Bonilla 2017)

Investigadores de la universidad de Antiquia, en Medellín desarrollaron este conjunto de datos. Se trata de 23 jóvenes adultos realizando 19 ADLs y 15 caídas distintas. Consiguieron llegar al 96% de precisión para la detección de caídas.

Cabe destacar que las medidas se tomaron con un dispositivo propio, y situado en un cinturón, es decir, sobre la cadera. La posición en la cadera permite poder trabajar solo con dos ejes, ya que uno de ellos es siempre perpendicular al cuerpo, por lo que se puede llegar a conseguir una precisión muy alta en comparación a un dispositivo en la muñeca.

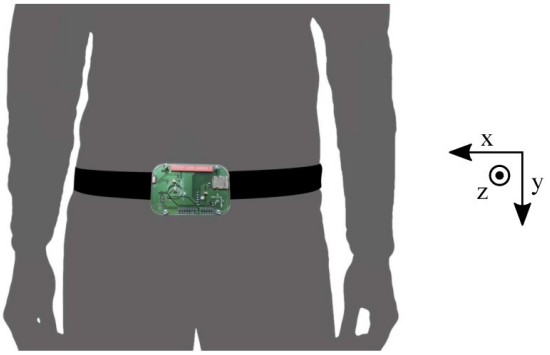


Figura 6: Colocación del sensor de SisFall

UCI (Ã–zdemir y Billur 2014)

En este caso, se creó un conjunto de datos a partir de 17 voluntarios, que realizaron 20 caídas y 16 ADLs, repitiendo cada una de ellas 5 veces. Cada sujeto tenía 6 sensores situados en partes distintas del cuerpo (cabeza, pecho, cadera, muñeca, muslo y tobillo).

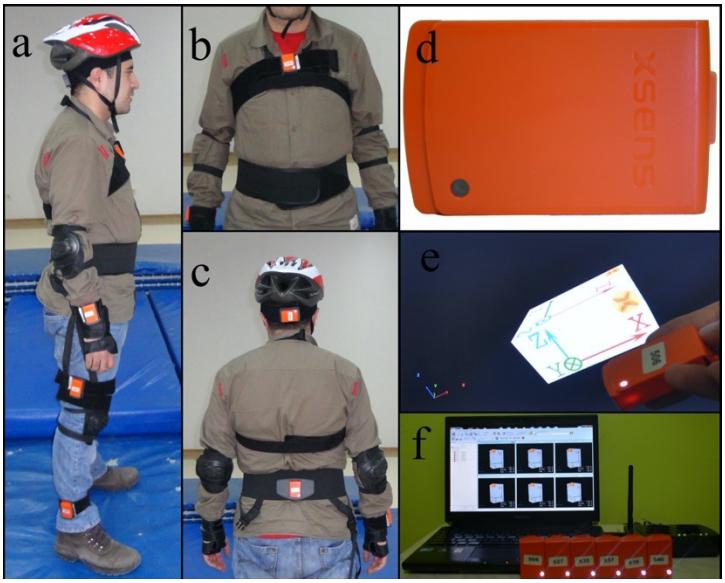


Figura 7: Colocación de los sensores para el conjunto de datos de UCI

Este estudio es uno de los más completos, y uno de los que obtiene mejores resultados, teniendo en cuenta el número de sensores en distintas partes del cuerpo y el gran número de datos que contempla (17 voluntarios, 20 caídas, 17 ADLs y cada acción repetida 5 veces). Llega a obtener precisiones del 99,91% al realizar combinaciones entre datos de distintas posiciones del cuerpo.

UP-Fall (Villaseñor, y otros 2019)

El objetivo de este proyecto es crear un conjunto de datos que permita aplicar algoritmos de aprendizaje automático para ser capaz de predecir caídas a partir de distintas señales de entrada, como podrían ser los datos de un acelerómetro situado en una parte determinada del cuerpo, o cámaras en partes concretas de casa.

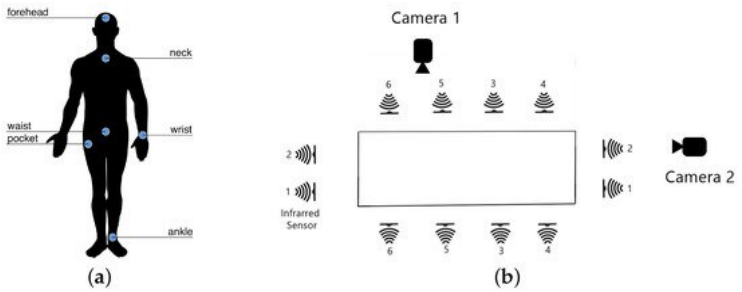


Figura 8: Colocación de los sensores

A partir de los datos recopilados, se realizó una comparación de los resultados obtenidos a partir de varios modelos, y diferentes datos utilizados para cada modelo, combinando datos de acelerómetro en diferentes partes del cuerpo, datos visuales de cámaras, y lecturas de infrarrojos.

2.3. Detección de fibrilación auricular

Cada vez más son los estudios que intentan definir una manera de detectar la fibrilación auricular a partir de algún método no invasivo. A continuación, se citan y resumen algunos de los estudios que han llegado más lejos en este campo.

Stanford y Samsung (Yichen, y otros 2018)

La universidad de Stanford y Samsung se unen para crear un algoritmo que detecta FA a partir de señal PPG con un 95% de AUC[1].
Dispone de un total de 510.566 registros de PPG de 81 pacientes. Algunos de ellos presentaban anomalías cardíacas, mientras que los otros tenían un ritmo sinusoidal normal.

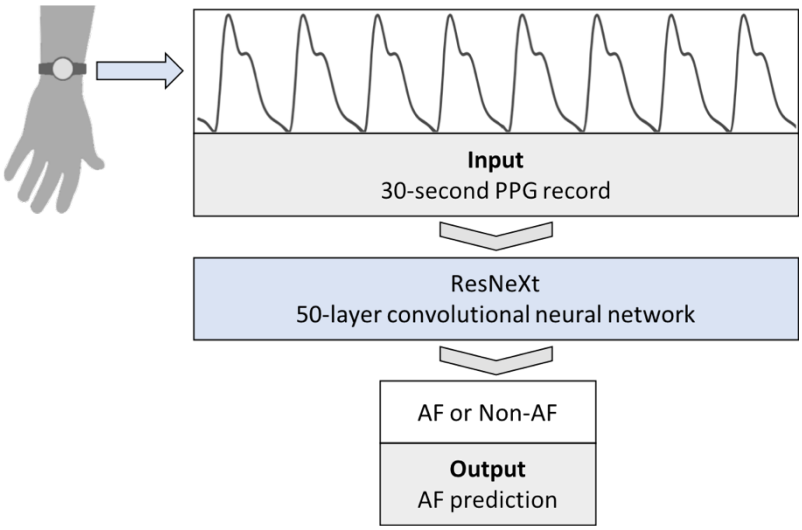


Figura 9: Flujo de datos en el estudio

Se realizó la captura de información a partir de dispositivos Samsung en la muñeca, y se utilizó una red neuronal de 50 capas para clasificar las señales PPG.

Apple (Apple 2018)

El Apple Watch Series 4, a partir de la actualización de software watchOS 5.1.2, permite detectar arritmias como la fibrilación auricular. Esto permite una monitorización a partir de un Smartphone o a través del mismo reloj.

[1] *Area Under Curve*: medida que indica como de bien distingue entre clases el modelo. Un modelo que clasifique todas las muestras mal, tendrá AUC = 0, mientras que si clasifica el 100% bien tendrá AUC = 1

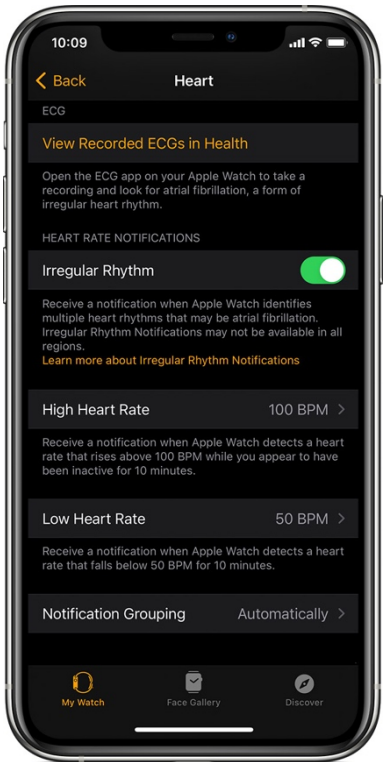


Figura 10: Configuración de notificaciones de arrítmias

Un estudio realizado en 2018, mostró que mientras que es posible utilizar únicamente PPG para la detección de fibrilación auricular, la detección es más precisa si se utiliza un electrocardiograma (ECG). El dispositivo de Apple también dispone de uno en un botón del lateral.

JAMA Cardiology (Geoffrey, y otros 2018)

Al igual que con el estudio de Apple, en este estudio se llegó a la conclusión de que es posible detectar fibrilación auricular, pero con menos precisión y sensitividad que a través de ECG. Como dispositivo para realizar las medidas, se utilizó la aplicación “Cardiogram” que recoge información sobre dispositivos Apple Watch.

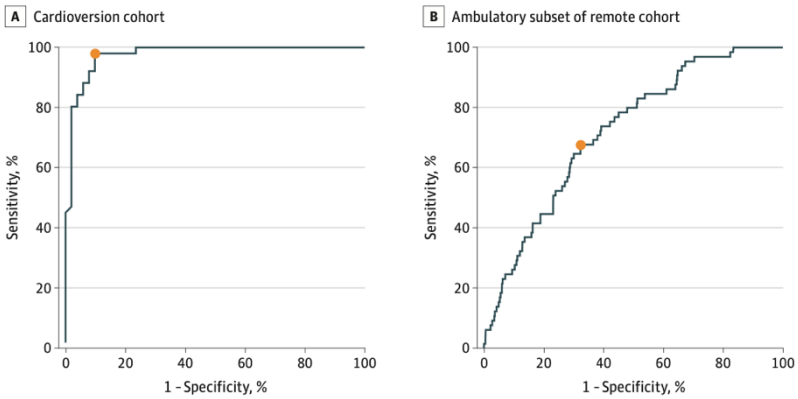


Figura 11: ROC para dos conjuntos de datos diferentes

University of California (Defne, y otros 2020)

Para este estudio se utilizó un dispositivo pulsera de Jawbone Health. El artículo se centra en definir cual es el mejor modelo para detectar FA a partir de PPG. Se prueban tres modelos diferentes:

- Modelo tradicional: se usa la variabilidad de frecuencia cardíaca como variable predictora del modelo de regresión logística.
- LSTM (Long Short-Term Memory): se utiliza esta red neuronal a la que tiene como entrada 35 latidos consecutivos.
- DNN (Deep convolutional-recurrent Neural Network): esta red utiliza la señal PPG sin procesar.

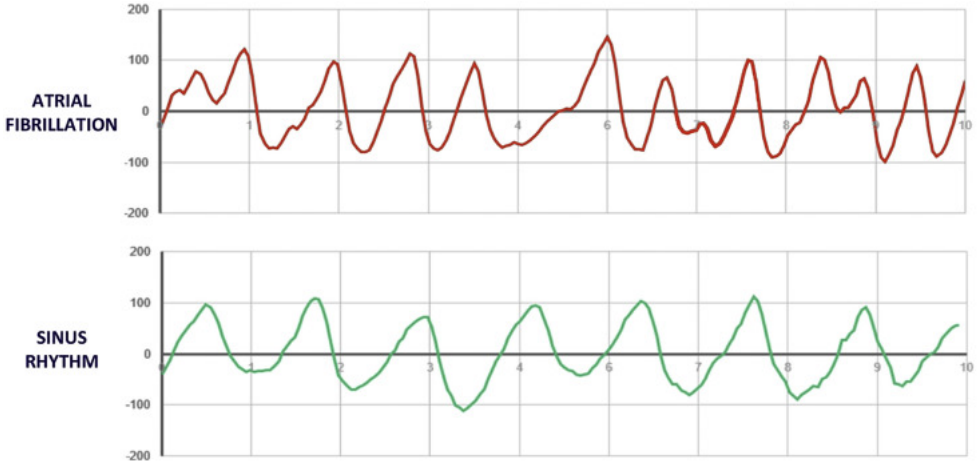


Figura 12: PPG con FA y sin FA

El resultado final de las pruebas indica que el modelo más fiable para predecir FA se obtiene a partir de la señal PPG sin procesar y la DNN.

3. Diseño e implementación

En este apartado se define la estructura que tendrá la implementación de nuestro proyecto, cómo se realizará dicha implementación, y las herramientas utilizadas para conseguirlo.

3.1. Herramientas utilizadas

Python será el lenguaje de programación utilizado. Dispone de las librerías que contienen los métodos y funcionalidades necesarios para realizar el análisis que se propone en este proyecto. Además, el objetivo final es que el código funcione en una Raspberry Pi que analizará los datos de acelerómetro que el reloj transmitirá en tiempo real, por lo que un script Python es compatible con esta.

El IDE que se usará para la implementación del código es Spyder, de la distribución de Anaconda. Se trata de un entorno con funcionalidades muy útiles como el explorador de variables, funciones de visualización avanzadas y ejecución por lotes de código.

El hecho de utilizar la plataforma Anaconda permitirá la fácil instalación de librerías, y tener siempre actualizado el entorno de desarrollo.

Para el control de versiones se utilizará Github, de modo que se podrán ir creando nuevas funcionalidades para el código con la tranquilidad de tener versiones estables en todo momento guardadas. El código puede ser encontrado en: www.github.com/arnautiendat/fall_detection.

3.2. Diseño e implementación

A continuación, se presenta un esquema sobre el diseño final de la implementación del proyecto, el cual se divide en los siguientes bloques (se verán en detalle en los siguientes apartados):

1. **Extraer, Transformar y Cargar:** A partir de las diferentes bases de datos de acelerómetros, se trata la información para unificar los formatos en uno de solo. Adicionalmente se calculan nuevas características que servirán para entrenar el modelo. Se acaba creando un único conjunto de datos que contendrá los datos de diferentes estudios.
2. **Análisis exploratorio:** En este análisis se visualizan las principales características de los datos de los que disponemos.
3. **Entrenamiento de los modelos:** Se entrenan varios modelos a partir de una división de los datos.
4. **Evaluación de los modelos:** Se utiliza la matriz de confusión y la curva ROC (Receiver Operating Characteristic) para evaluar la calidad de los modelos**.**
5. **Detección:** Se selecciona el modelo que ha dado mejores resultados para realizar una detección de caídas a partir de un conjunto de datos de test.

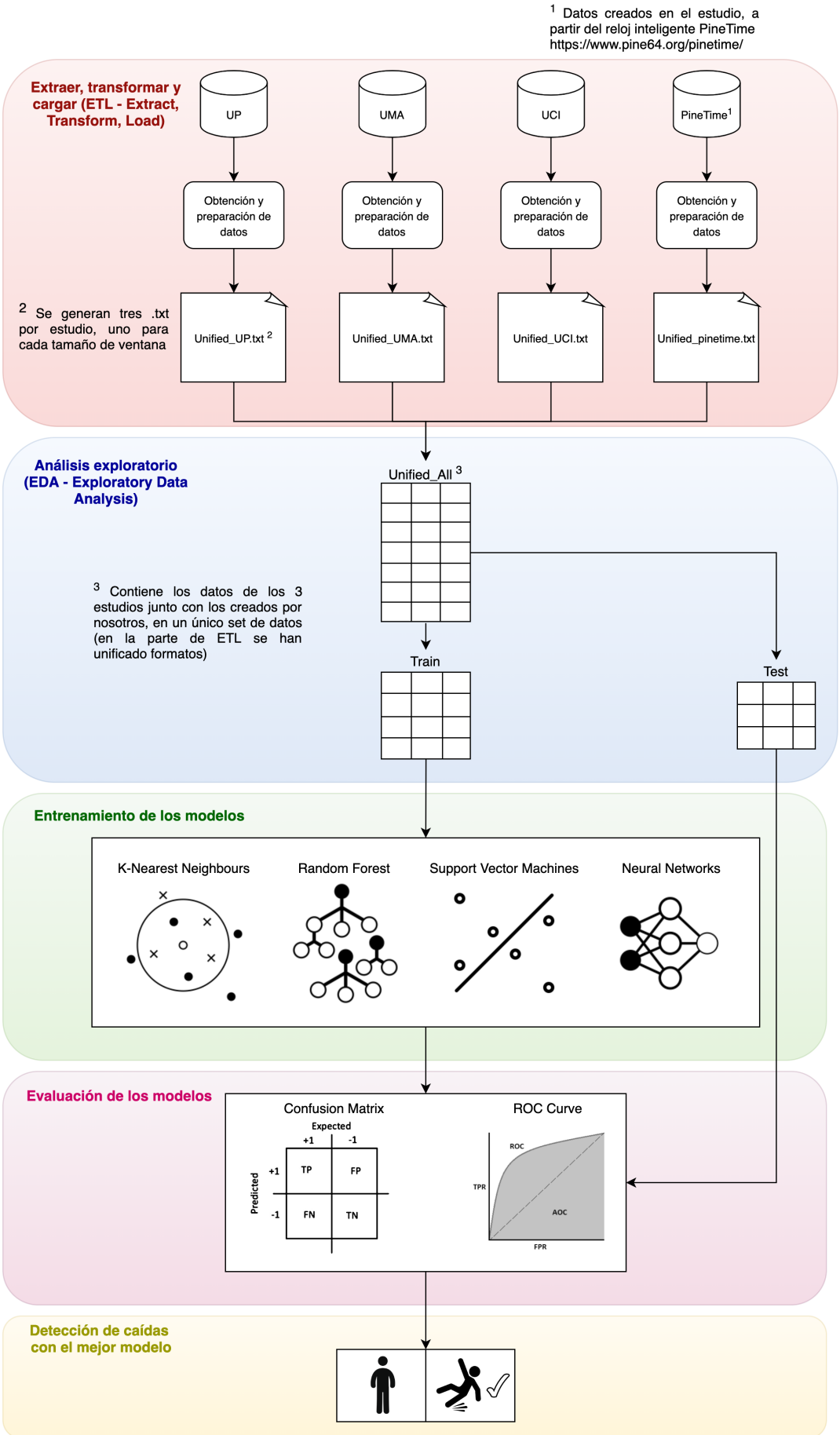


Figura 13: Diseño de la implementación del proyecto

3.2.1. Extraer, transformar, cargar (ETL)

Para empezar, se seleccionarán los datos que pueden contribuir a entrenar los modelos de aprendizaje. Estos datos saldrán básicamente de las investigaciones que se han analizado en el apartado anterior.

El proceso de ETL sigue los mismos pasos para cada uno de los conjuntos de datos de cada estudio. Se trata del siguiente:

Extracción

Se realiza la carga de los ficheros, que para cada estudio (UP, UCI y UMA, los cuales se pueden ver en la Figura 13) se hace de una forma distinta. En cada uno de ellos se proporciona un documento que indica el significado de la nomenclatura de ficheros. Por ejemplo, en el caso del estudio UP-FALL, se proporciona la siguiente leyenda:

Information About Data Formation:

Folder Name 1XX Male Volunteers

Folder Name 2XX Female Volunteers

Folder Name 8XX Activities of Daily Living

Folder Name 9XX Fall Actions

Figura 14: Leyenda UP-FALL

Junto con un detalle de lo que significan las X en el nombre de los ficheros:

20 class of FALL ACTIONS # Label and Description

901 front-lying, from vertical falling forward to the floor

902 front-protecting-lying, from vertical falling forward to the floor with arm protection

903 front-knees, from vertical falling down on the knees

904 front-knees-lying, from vertical falling down on the knees and then lying on the floor

905 front-quick-recovery, from vertical falling on the floor and quick recovery

Figura 15: Ejemplo de detalle del significado de los nombres de ficheros

A modo de resumen, el nombre de las carpetas y ficheros está indicando el tipo de acción que se realiza y se utilizará para identificar a qué clase corresponde la lectura del acelerómetro de cada fichero. Para el resto de estudios, se utiliza una forma similar para la nomenclatura de carpetas y ficheros, especificada en un documento específico. Al descomprimir la carpeta que contiene todos los ficheros se obtiene la distribución que indican:

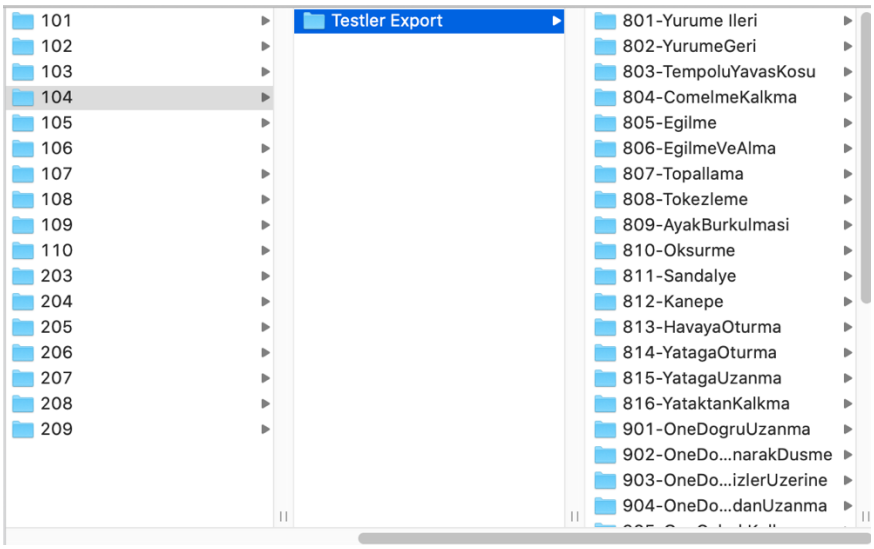


Figura 16: Distribución de carpetas y ficheros

Una vez se dispone de los datos, se filtrarán únicamente aquellas columnas que tienen las lecturas de los tres ejes X, Y, Z. En muchos de los ficheros aparecen lecturas de otros tipos de sensores. En nuestro caso, dado que el smartwatch utilizado únicamente tiene acelerómetro tri-axial, solo nos interesan estas tres componentes.

Transformación

Algunos estudios proporcionan la información del acelerómetro directamente en unidades de aceleración "g" (1g = 9.8m/s^2). Por otro lado, en algunos disponemos la lectura del conversor analógico-digital (AD), la cual debe ser transformada a unidad "g" para tener el mismo formato en todas las lecturas. Para hacerlo, se utiliza la siguiente fórmula:

$$Acceleration[g] = AD * \frac{2 * Range}{2^{Resolution}}$$

Donde:

AD (mV): Lectura del acelerómetro
Range (±g): Nivel de aceleración que soporta el sensor.
Resolution (bits): Niveles de aceleración diferentes que puede medir el sensor.

Los datos de los que se disponen son grabaciones de acciones de distintas longitudes. Pueden durar entre 6 y 60 segundos, dependiendo de como se haya realizado la captura durante el estudio. Para detectar el momento en el que se puede realizar la caída, se selecciona una ventana alrededor del punto de máxima aceleración de los tres ejes, utilizando la fórmula para calcular la norma vectorial de los tres vectores:

$$A_T = \sqrt{A_x^2 + A_y^2 + A_z^2}$$

La ventana será de medidas variables, y será uno de los parámetros que se modificarán para ver con cual se obtiene mejores resultados.

Se probarán ventanas de 1,5; 2; y 3 segundos alrededor del punto de máxima aceleración. Ejemplos:

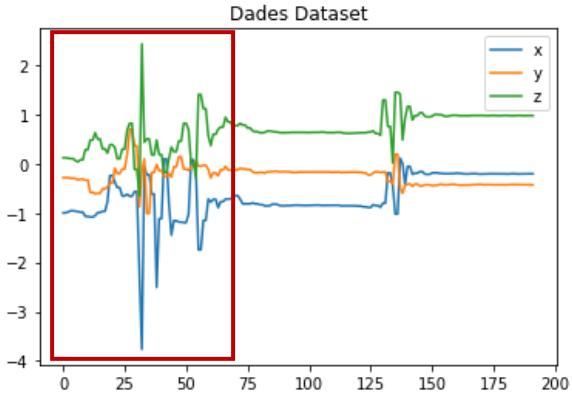


Figura 17: Ventana 2s en lectura de acelerómetro de caída frontal

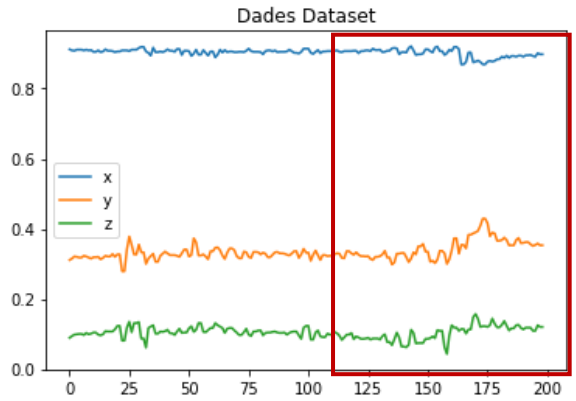


Figura 18: Ventana 3s en lectura de acelerómetro de persona andando

Para cada uno de los ficheros, se realizarán tres ventanas sobre el punto de máxima aceleración, de los tamaños definidos anteriormente.

Sobre estas ventas, se calcularán 54 características que permitirán a los modelos de entrenamiento sacar más relaciones que con únicamente los valores de los tres ejes X, Y y Z. La elección de las características se basa en una mezcla entre los diferentes estudios analizados en apartados anteriores y alguna contribución de Github (SpaceMinds 2019).

```
eatures
['kurtosis_X','max_X','mean_X','min_X','range_X','skewness_X','std_X','var_X',
 'kurtosis_Y','max_Y','mean_Y','min_Y','range_Y','skewness_Y','std_Y','var_Y',
 'kurtosis_Z','max_Z','mean_Z','min_Z','range_Z','skewness_Z','std_Z','var_Z',
 'kurtosis_N_XYZ','max_N_XYZ','mean_N_XYZ','min_N_XYZ','range_N_XYZ','skewness_N_XYZ','std_N_XYZ','var_N_XYZ',
 'kurtosis_N_HOR','max_N_HOR','mean_N_HOR','min_N_HOR','range_N_HOR','skewness_N_HOR','std_N_HOR','var_N_HOR',
 'kurtosis_N_VER','max_N_VER','mean_N_VER','min_N_VER','range_N_VER','skewness_N_VER','std_N_VER','var_N_VER',
 'corr_HV','corr_NH','corr_NV','corr_XY','corr_XZ','corr_YZ']
```

Figura 19: Características nuevas

A continuación, se define el significado de cada una de las medidas. Cabe recordar que se realizan los cálculos únicamente sobre la ventana definida, no sobre la duración de la grabación original.

Para los tres ejes:

- *Kurtosis*: indica si la distribución tiene más “cola” o menos respecto una distribución normal.
- *Skewness*: indica la falta de simetría de una distribución.
- *Mean*: media.
- *Max*: valor máximo.
- *Min*: valor mínimo.
- *Range*: valor máximo – valor mínimo.
- *Std*: Desviación estándar.
- *Var*: Varianza.

Adicionalmente a la nueva columna creada XYZ, la cual tendrá el valor obtenido de la fórmula de aceleración total vista anteriormente, también se crean dos campos computados más. Se trata de la norma vectorial de la aceleración horizontal, y la norma vectorial de la aceleración vertical. Se calculan teniendo en cuenta únicamente dos componentes en vez de tres.

$$A_V = \sqrt{A_x^2 + A_z^2}\ ;\ A_H = \sqrt{A_Y^2 + A_z^2}$$

Tanto para la aceleración total, como para la vertical y la horizontal, se calculan las características especificadas anteriormente.

Finalmente, se calculan correlaciones entre los tres ejes (2 a 2), y entre las aceleraciones totales, verticales y horizontales (2 a 2). Como se puede observar en la Figura 19, obtenemos un total de 54 características. Las cuales se escriben en un fichero al cual llamaremos Unified_*nombre_estudio*.txt.

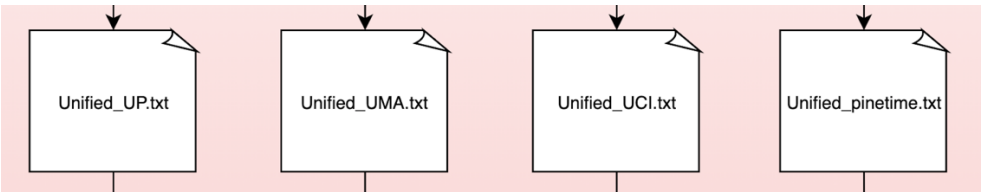


Figura 20: Parte del esquema referente a los ficheros Unified

A modo de resumen, se agrupan todos los ficheros de caídas y actividades diarias de un estudio concreto en un solo fichero, en el que cada línea de este son las 54 características descritas anteriormente de la ventana de ‘X’ segundos escogida del fichero. Con tal de hacer referencia al diseño inicial, se trataría de los ficheros Unified_XX_YY.txt (Figura 20), que contienen la información de todas las caídas y actividades diarias de su estudio correspondiente, uno en cada línea. ‘XX’ corresponde al nombre del estudio, ‘YY’ corresponde a la longitud de la ventana de observación.

Se pasa de tener un archivo con las lecturas X, Y y Z una en cada columna, a una línea en el archivo Unified que resume sus 54 características en la ventana de observación correspondiente.

Carga

Se cargan todos los ficheros .txt en tres tablas diferentes, una para cada longitud de ventana de observación.

Figura 21: Ejemplo de tabla creada

En el ejemplo se puede observar que se dispone de las siguientes columnas:

- Nombre del fichero original.
- Clase: F para caídas (*Falls*), D para actividades diarias (*Daily activities*). Esta es la clase que se usará para entrenar el modelo, y la que querremos predecir.
- Act_Type: Detalle sobre el tipo de actividad. La correspondencia del código con la actividad la tenemos en los archivos *Readme* de los diferentes estudios.
- 54 características creadas en el apartado de transformación.

Esta tabla contiene la información necesaria para generar los conjuntos de entrenamiento y de test.

3.2.2. Análisis exploratorio

Se realiza el análisis exploratorio para una de las tablas (ventana de 3 segundos), ya que las otras dos siguen el mismo formato. Se realiza únicamente sobre una tabla ya que el análisis exploratorio realizado es básicamente sobre la estructura de las tablas, la cual es la misma para todas. Respecto al análisis de outliers, sí que serán distintos para cada uno de los conjuntos, pero el resultado final es el mismo para las tres, ya que como decisión de diseño no se realizarán acciones sobre ellos (se explicará con detalle en los siguientes puntos).

Para empezar, se analizan las medidas de la tabla. Se trata de una tabla de dimensiones (4649,57). Esto significa que disponemos de un total de 4649 observaciones, que corresponden a 4649 acciones, y 57 columnas correspondientes a los datos explicados en el apartado anterior.

Se empieza viendo cuantos de los registros son caídas, y cuantos son actividades diarias (*F – Falls, y A – Activites of daily life*).

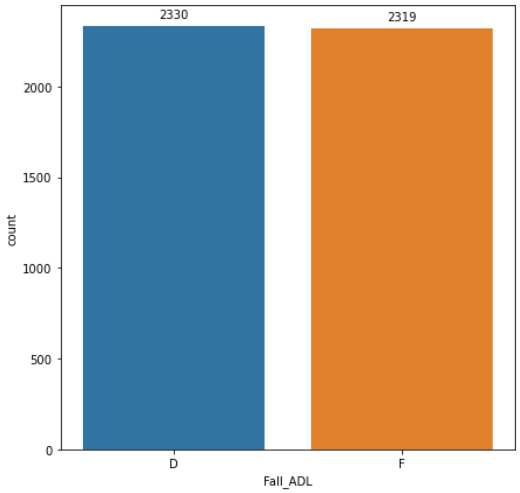


Figura 22: Número de caídas y actividades diarias para el estudio

Se observa que se dispone de un número muy balanceado de registros para las dos clases, lo que permitirá un buen entrenamiento de los modelos.

Se busca también si hay campos que no hayan sido cargados correctamente, o campos vacíos. Se realiza un mapa de calor para ver aquellas variables que tengan este tipo de campos.

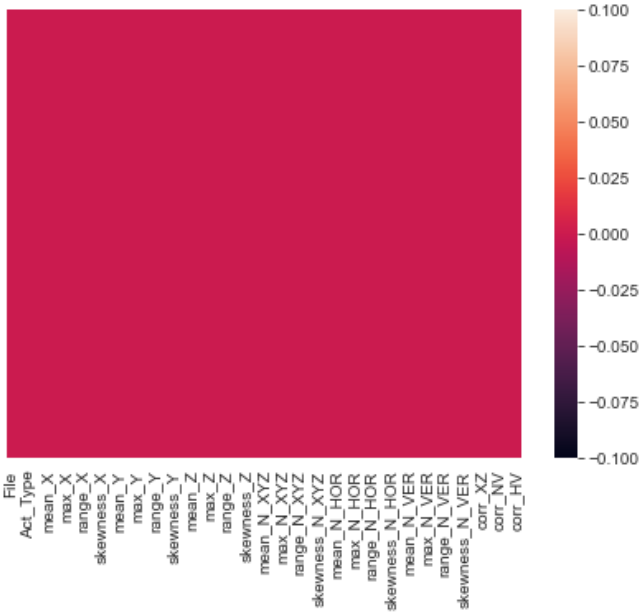


Figura 23: Mapa de calor de campos vacíos

Como se puede comprobar mediante el mapa de calor, el color indica el número de registros que están en blanco. Todos son del mismo color, el cual corresponde a 0, por lo que se confirma que no hay ningún campo vacío.

Seguidamente, se comprueba la existencia de valores atípicos, también llamados *outliers*. Se muestra un diagrama de 4 variables de las que disponemos como ejemplo.

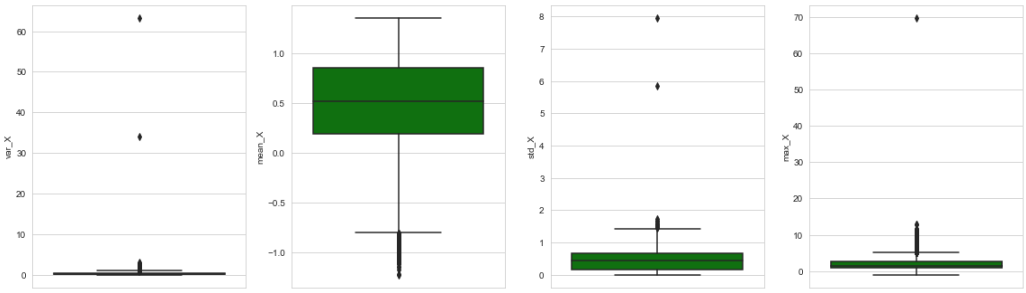


Figura 24: Outliers de las variables de estudio

En el caso de este proyecto, al tratar con variables que se desprenden de la aceleración en los tres ejes durante un proceso como puede ser una caída, se interpreta que puede haber valores atípicos al poderse producir un movimiento de gran aceleración en un tipo de caída determinada. Por ello, como decisión de diseño no se van a realizar acciones sobre estos valores atípicos.

Finalmente, se realiza una separación en dos conjuntos de datos diferentes, uno de entrenamiento, con el 70% de los datos, y uno de test, con el 30% restante. El número total de registros para cada uno de ellos es el siguiente:

```
Total ADL: 2330
Total Falls: 2319
GRAND Total: 4649
_________________________

70% train, 30% test

Train Falls: 1618
Train ADL: 1636
Train TOTAL: 3254

Test Falls: 701
Test ADL: 694
Test TOTAL: 1395

_________________________
```

Figura 25: Número de registros de cada conjunto

Estos conjuntos son los que se utilizarán en los siguientes apartados, el primero para entrenar los modelos, y el segundo para evaluarlos y obtener una puntuación sobre su capacidad para clasificar caídas y actividades diarias.

3.2.3. Entrenamiento de los modelos

Para el entrenamiento del modelo, únicamente se utilizan las 54 características creadas en apartados anteriores, de manera que se excluyen las columnas con el nombre del fichero, la clase (se utilizará como variable a predecir), y el tipo de actividad.

En la Tabla 1 se han podido ver los métodos utilizados en otros estudios. En ellos se utiliza un solo conjunto de datos (creado *ad-hoc* para el estudio), y algún modelo de aprendizaje supervisado. En este proyecto se trabaja con 4 conjuntos de datos distintos, los cuales se unifican en un solo conjunto, y se entrenan 4 modelos diferentes: K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Random Forest (RF), Neural Networks (NN).

Para los tres primeros modelos se utilizará un método de búsqueda de hiperparámetros[2] óptimos llamado *Grid Search.* Este método nace de la necesidad de ajustar los hiperparámetros y analizar recursivamente el resultado de aplicar el modelo con estos. El proceso que se seguiría si no utilizáramos el método de optimización de hiperparámetros es el siguiente:

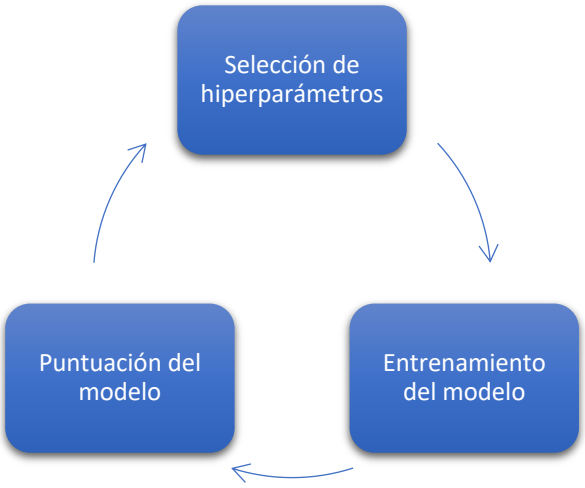


Figura 26: Selección de hiperparámetros manual

Se seleccionarían unos hiperparámetros, basándonos en nuestra experiencia o en estudios similares. Por ejemplo, en el modelo KNN, se seleccionaría K = 1 y una función de pesos uniforme. Se entrenaría el modelo y se miraría la puntuación del modelo con un conjunto de test. Se iría repitiendo el proceso hasta obtener un modelo que se considere suficientemente bueno. Como se puede intuir, es un proceso bastante repetitivo y poco óptimo.

Cuando se utiliza un *Grid Search*, se definen los hiperparámetros a probar, y la propia función realiza todas las combinaciones posibles entre ellos para el entrenamiento analizando las puntuaciones de los modelos

[2] Valores que deben configurarse manualmente en un modelo para controlar el proceso de aprendizaje

resultante para cada combinación, y guardando siempre los parámetros que han conseguido un modelo con mayor puntuación.

Además, la función *GridSearchCV* (Malik 2018) de la librería *Scikit-learn* realiza la evaluación del modelo mediante validación cruzada, garantizando de esta forma que las particiones en conjuntos de entrenamiento y prueba no influyen en el resultado.

Figura 27: Selección de hiperparámetros mediante GridSearch

Finalmente, se obtiene la mejor combinación de hiperparámetros para generar el modelo con los datos de los que se disponen. A continuación, se detallan los modelos utilizados y su funcionamiento (se usa como ejemplo el estudio sobre la ventana de 3 segundos).

K-Nearest Neighbors

Se trata de un modelo de aprendizaje supervisado, muy utilizado en problemas de clasificación. Se basa en la similitud entre las características de los datos. En el caso del estudio, se puede considerar que las diferentes caídas tendrán unas características similares (por ejemplo, un valor de aceleración total elevado), mientras que el resto de actividades diarias también serán parecidas entre sí.

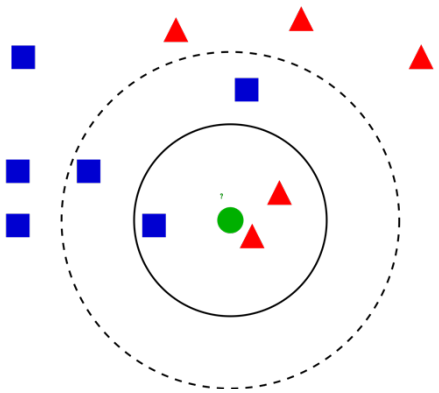


Figura 28: Esquema ejemplo del funcionamiento de KNN

De este modo, el algoritmo clasificará una nueva muestra en base a como de parecidas son sus 54 características respecto a las muestras de entrenamiento. El número de muestras en el que se fijará (número de vecinos) lo marca el hiperparámetro, *n_neighbors* y el método de cálculo de pesos en la distancia se define con *weights.*

Se utiliza la función *KNeighborsClassifier* de la librería *Scikit-learn*. A partir del análisis *Grid Search* se han seleccionado los siguientes hiperparámetros a analizar:

- *n_neighbors = range(1,11)*
- *weights = ['uniform', 'distance']*

Random Forest

Al igual que para el algoritmo KNN, se trata de un algoritmo de aprendizaje supervisado, el cual se basa en la creación de árboles de clasificación y su mezcla para obtener mejores resultados que con uno solo. Aquí veríamos un ejemplo con dos árboles:

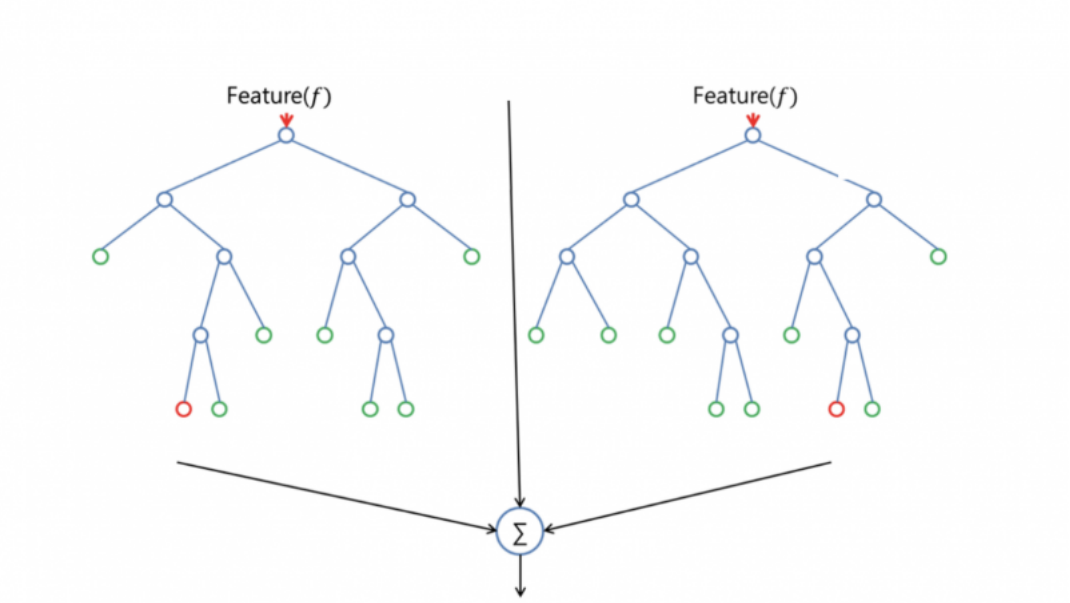


Figura 29: Random Forest con dos árboles (Donges 2019)

En los árboles generados no se usan todas las características del conjunto de datos, por lo que generalmente se obtiene una mejor precisión al combinar diferentes características para los árboles de decisión.

Los parámetros a definir son el número de arboles que se utilizarán (*n_estimators*) y la profundidad máxima del árbol (*max_depth)*. A más profundidad, tendrá más divisiones y podrá capturar más información sobre los datos.

Se utiliza la función *RandomForestClassifier* de la librería *Scikit-learn*. A partir del análisis *Grid Search* se han seleccionado los siguientes hiperparámetros a analizar:

- *n_estimators = range(1,200,20)*
- *max_depth = range(1,10)*

Support Vector Machines

Las máquinas de vectores de soporte son un método de aprendizaje supervisado que genera hiperplanos que intentan separar las muestras de distintas clases. Se busca aquel hiperplano que tenga una distancia máxima a las muestras de cualquier otra clase.

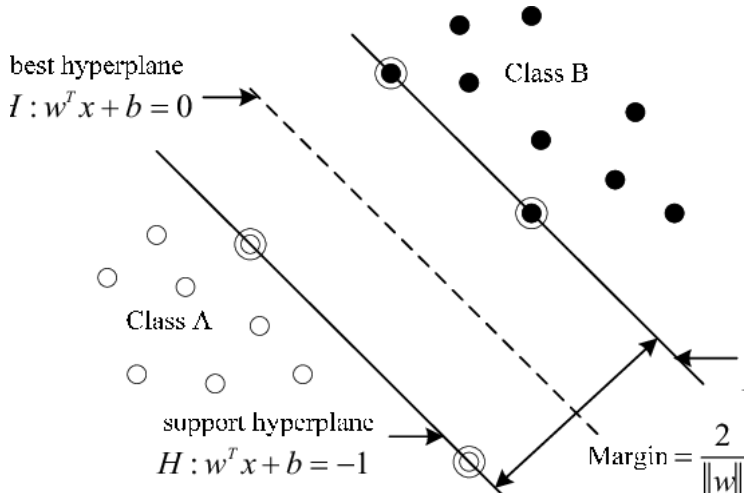


Figura 30: Ejemplo de SVM

En la Figura 30 observamos que el mejor hiperplano seleccionado corresponde a aquel que maximiza la distancia entre los dos hiperplanos de soporte.

Los hiperparámetros a definir son el margen entre vectores de soporte (*C*), la transformación que se le aplica a los datos de entrada (*kernel*) y el radio de influencia de cada muestra de entrenamiento (*gamma*).

Se utiliza la función *SVC* de la librería *Scikit-learn*. A partir del análisis *Grid Search* se han seleccionado los siguientes hiperparámetros a analizar:

- *C = range(1,10,1)*
- *kernel = ['rbf','linear','poly','sigmoid']*
- *gamma = ['scale','auto']*

Neural Network

Se usará una red con características similares a las utilizadas en los estudios de la bibliografía. Para poder usar los datos de los que

disponemos como datos de entrada de la red neuronal, se realiza una normalización de cada una de las columnas, y se transforma la variable objetivo a binaria (hasta el momento teníamos la clase “F” correspondiente a caídas y “D” correspondiente a actividades diarias). La red neuronal requiere que sean 1 i 0.

Se utiliza la siguiente red, con una capa de entrada de 54 nodos (correspondientes con las 54 características que se analizan de la lectura de acelerómetro), y tres capas ocultas de 128, 64, y 64 nodos respectivamente. La capa final consta de dos nodos, para clasificar en las dos clases del estudio: caídas o actividades diarias. La función de activación para las capas es “Relu”, la cual mantendrá las salidas si son positivas, o las fijará a 0 si son negativas. Esta función de activación permite obtener buen rendimiento en entrenamiento. Finalmente, se utiliza la función “Softmax” para la capa de salida, para obtener valores entre 0 y 1 que nos marquen la probabilidad de que la lectura seleccionada sea una caída o una actividad diaria.

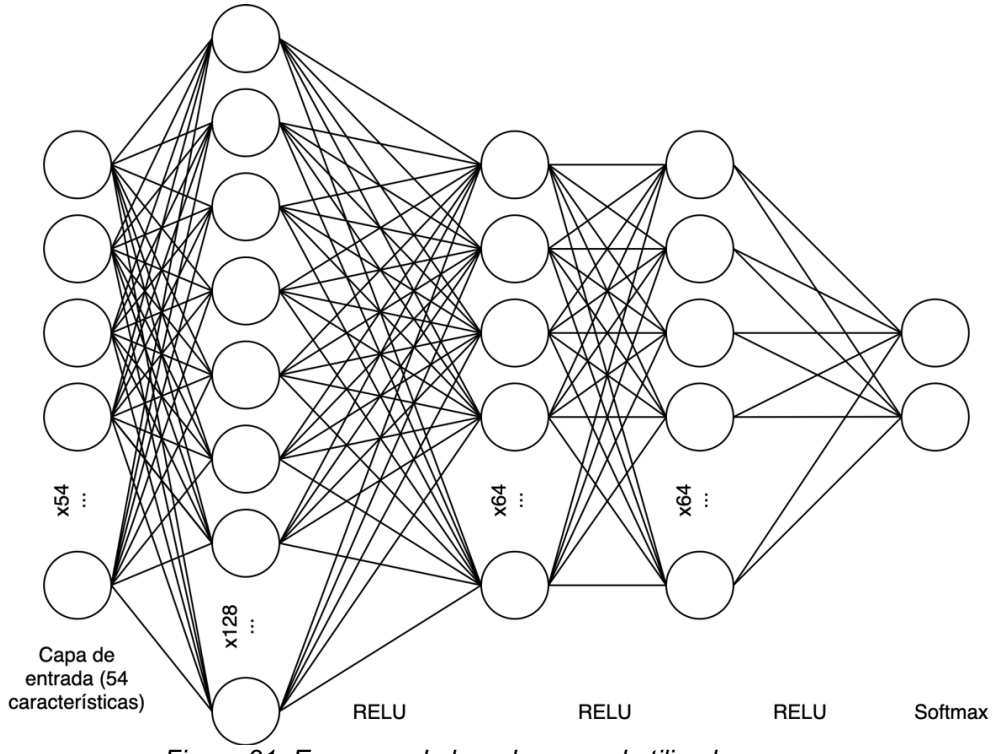


Figura 31: Esquema de la red neuronal utilizada

Los hiperparámetros utilizados son:

- *optimizer = Adam.*
- *loss = cross entropy* (mide la diferencia entre la probabilidad predicha y la etiqueta real).
- *epochs = 500*
- *batch_size = 32*

Se utiliza una clase de *tensorflow* llamada *callbacks* que permite guardar ciertos parámetros del proceso de entrenamiento. En este caso nos

servirá para guardar el mejor modelo creado en las 500 iteraciones realizadas.

3.2.4. Evaluación de los modelos

Se utilizarán dos herramientas de visualización de resultados. La primera es la matriz de confusión, y el segundo la curva AUC (*Area Under Curve*) – ROC (*Receiver Operating Characteristics*). Antes de detallar estos dos métodos de evaluación, se indican algunos términos que los definen, y se indicará como se relacionan con nuestro caso de estudio (caídas y actividades diarias) (Narkhede, Towards Data Science 2018):

- Positivo Verdadero (*TP – True Positive).* Predicho como caída y es caída.
- Positivo Falso (*FP – False Positive).* Predicho como caída y es actividad diaria.
- Negativo Verdadero (*TN – True Negative).* Predicho como actividad diaria y es actividad diaria.
- Negativo Falso (*FN – False Negative).* Predicho como actividad diaria y es caída.

A partir de estos términos, se pueden definir los siguientes valores asociados:

- Sensibilidad: capacidad de clasificar los casos positivos como positivos (determinar que una caída ha sido una caída).

$$Sensibilidad = \frac{TP}{TP + FN}$$

- Especificidad: capacidad de clasificar los casos negativos como negativos (determinar que una actividad diaria ha sido una actividad diaria).

$$Especificidad = \frac{TN}{TN + FP}$$

Este proyecto se orienta a alertar a los familiares de personas mayores cuando estas sufran una caída. **Es por ello que, a diferencia de otros estudios de la bibliografía, se va a dar más importancia a la sensibilidad frente a la especificidad**. Es menos crítico alertar que ha habido una caída cuando en realidad no la ha habido, que no alertar una caída que sí que ha sucedido.

La fórmula utilizada será una media ponderada de los dos puntos anteriores:

$$Puntuación\ del\ modelo\ = Sensibilidad * 0.7 + Especificidad * 0.3$$

De esta forma una gran parte del peso de decidir el modelo radica en clasificar perfectamente las caídas.

Matriz de confusión

Representa el número de observaciones de la clase predicha en las filas, y el número de observaciones de la clase real en las columnas, de manera que es sencillo visualizar aquellas que no han sido clasificadas correctamente.

Figura 32: Leyenda de matriz de confusión

Se mostrarán las matrices de confusión para cada uno de los cuatro modelos utilizados, y las tres ventanas de tiempo correspondientes. Tendremos un total de doce matrices de confusión. Los valores calculados por la matriz de confusión se usan para calcular la sensibilidad y especificidad utilizados en el apartado anterior para calcular la puntuación del modelo.

Curva AUC – ROC (Narkhede, Towards Data Science 2018)

La curva AUC – ROC se usa como medidor del desempeño de un método de aprendizaje supervisado. Indica como de bueno es el modelo distinguiendo clases. El AUC indica el área bajo la curva ROC, por lo que un buen modelo tendrá una AUC cercana a 1, y un mal modelo tendrá AUC cercana a 0 (clasificará todas las clases mal).

La curva ROC muestra en el eje "y" el Ratio de Verdaderos Positivos (*TPR – True Positive Rate),* también llamado Sensibilidad, y en el eje de las "x" el Ratio de Falsos Positivos (*FPR – False Positive Rate)*, calculado como:

$$FPR = 1 - Especificidad$$

La curva ROC a continuación muestra su funcionamiento:

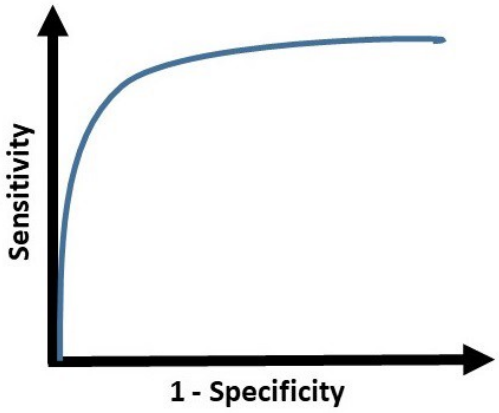


Figura 33: Ejemplo de curva ROC

Se analizará la AUC para cada ventana temporal y modelo, buscando siempre aquella más cercana a 1.

3.2.5. Evaluación de los modelos

Una vez se ha seleccionado el mejor modelo, será el utilizado para la detección de caídas. El programa solicitará como entrada la ruta a una grabación de acelerómetro (sus 3 ejes), y la frecuencia de muestreo a la que se ha realizado la grabación. Cabe destacar que los datos de aceleración tienen que estar en unidades g (1g = 9.81m/s^2). Al ejecutar el programa, obtenemos la siguiente pantalla:

```
---------------------------------
 _____     _     _      _
|  ___|   / \   | |    | |
| |_     / _ \  | |    | |
|  _|   / ___ \ | |___ | |___
|_|    /_/   \_\|_____||_____|


 ____   _____  _____  _____   ____  _____  ___   ___   _   _
|  _ \ | ____||_   _|| ____| / ___||_   _||_ _| / _ \ | \ | |
| | | ||  _|    | |  |  _|   | |      | |   | | | | ||  \| |
| |_| || |___   | |  | |___  | |___   | |   | | | |_| || |\  |
|____/ |_____|  |_|  |_____| \____|  |_|  |___| \___/ |_| \_|


---------------------------------
Instructions:
        File accepted format: 3 columns, separated by commas, one for each axis (X,Y,Z).
        Values need to be coded in g (1g = 9.81m/s^2).

Example:

\        X      Y      Z
4  -0.998 -0.270  0.151
5  -0.998 -0.270  0.151
6  -1.000 -0.332  0.192
7  -0.976 -0.373  0.133
8  -0.976 -0.373  0.133
..    ...    ...    ...
---------------------------------

Introduce the path to your csv file:
```

Figura 34: Ejemplo pantalla de inicio del programa

Una vez introducida la ruta al fichero y la frecuencia de muestreo, se indica si se ha detectado una caída o no. Probamos con un fichero de una actividad diaria:

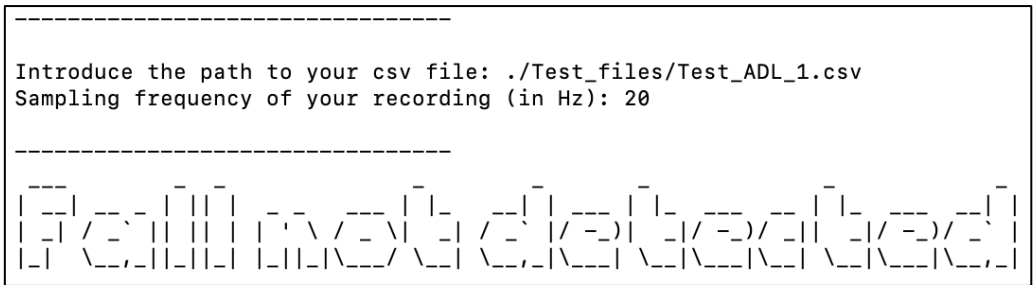


Figura 35: Ejemplo de no caída no detectada

Y con un fichero de una caída:

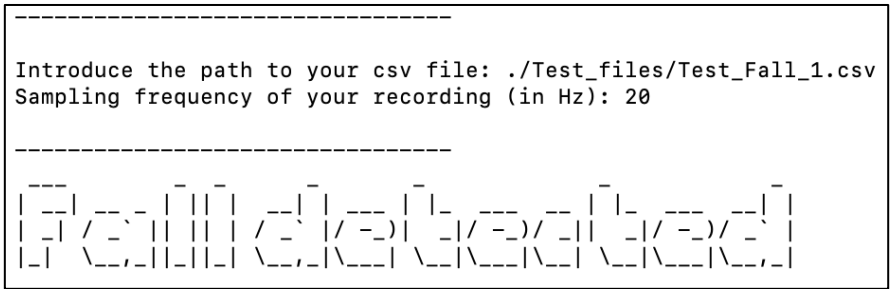


Figura 36: Ejemplo de caída detectada

Esta detección de caídas iniciaría un proceso en el sistema de alertas para que los familiares de las personas mayores pudieran realizar las acciones necesarias.

4. Resultados

4.1. Selección de hiperparámetros y precisión

A continuación, se indicará, para cada modelo, el valor de los hiperparámetros seleccionados por la función *Grid Search*, y la precisión media en entrenamiento de las cuatro validaciones cruzadas.

Modelo	**Hiperparámetros**	**Precisión media**
KNN	*n_neighbors* = 6 *weights = distance*	91.89%
RF	*n_estimators* = 140 *max_depth* = 9	94.93%
SVC	*C* = 9 *kernel = scale* *gamma = linear*	93.33%

Tabla 2: Resultados de la función Grid Search

Para la red neuronal, los hiperparámetros se han seleccionando teniendo en cuenta redes usados en investigaciones similares de la bibliografía. Se pueden ver estos parámetros en el apartado anterior.

4.2. Puntuación de los modelos

Se obtiene la siguiente tabla de puntuaciones de modelos según la ventana de estudio.

	Puntuación = Sensibilidad * 0.7 + Especificidad * 0.3		
Modelo	**Ventana 1.5s**	**Ventana 2s**	**Ventana 3s**
KNN	0.92	0.92	0.92
RF	0.96	0.96	0.96
SVC	0.93	0.93	0.95
NN	0.93	0.94	0.95

Tabla 3: Puntuación de modelos según ventanas temporales

A partir de este los resultados obtenidos se puede concluir que el tamaño de la ventana analizada no influye cuando se utilizan modelos KNN o RF, mientras que para SVC y NN las diferencias no son muy notables (obteniendo, no obstante, mejores resultados con ventanas más amplias). Los tamaños de 1.5, 2 y 3 segundos han sido seleccionados al ser el tipo de prueba realizada por múltiples estudios analizados en la bibliografía.

A nivel de modelo, los mejores resultados se obtienen utilizando bosques aleatorios (puntuación de 0.96 sobre 1), aunque muy seguido de máquinas de vectores de soporte y redes neuronales con 0.95 usando ventanas de tres segundos. El peor modelo es KNN obteniendo una puntuación de 0.92 para cualquiera de los tamaños de ventana de observación.

4.3. Matriz de confusión

Se indican los resultados obtenidos para cada modelo y tamaño de ventana de observación. Los valores se utilizan para calcular la sensibilidad y la especificidad usadas para definir la puntuación del modelo.

Ventana de 1.5 segundos

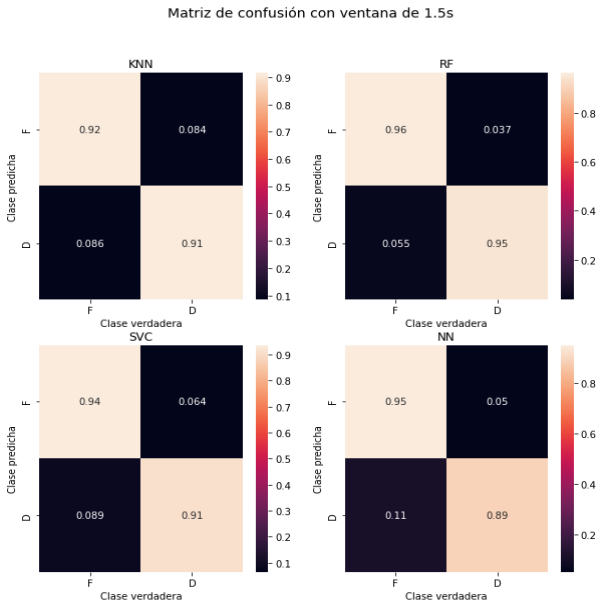


Figura 37: Matriz de confusión para ventana de 1.5 segundos

Ventana de 2 segundos

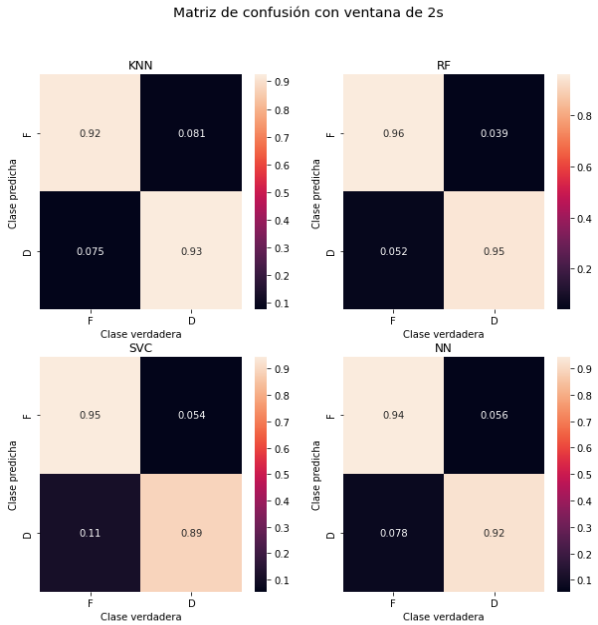


Figura 38: Matriz de confusión para ventana de 2 segundos

Ventana de 3 segundos

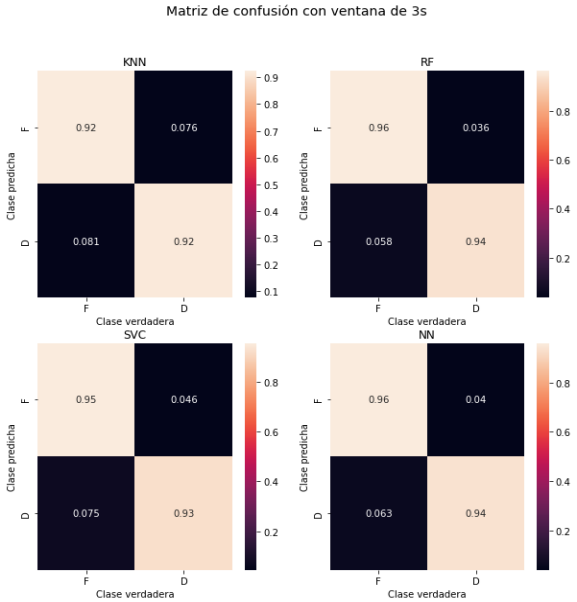


Figura 39: Matriz de confusión para ventana de 3 segundos

Como se ha indicado en apartados anteriores, en este estudio se da más importancia a la sensibilidad que a la especificidad, ya que es crítico clasificar de forma correcta las caídas para realizar las acciones que los familiares crean necesarias. La fórmula para calcularla es:

$$Sensibilidad = \frac{TP}{TP + FN}$$

Se deberá prestar mayor atención a los positivos verdaderos y a los falsos negativos de las matrices de confusión (también se ha tenido en cuenta a la hora de calcular la puntuación en el apartado anterior).

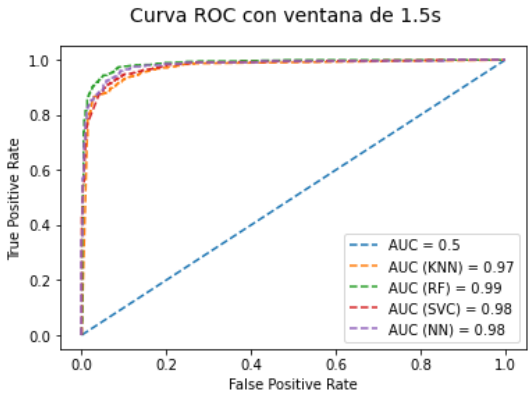
4.4. ROC – AUC

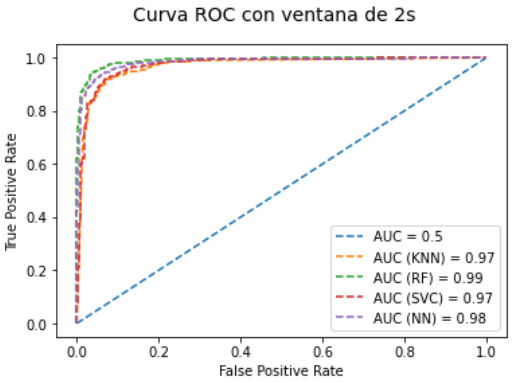
Se resumen las AUC de cada modelo y ventana temporal en la siguiente tabla (también pueden verse en las anotaciones de las curvas de la Tabla 5).

	AUC		
Modelo	**Ventana 1.5s**	**Ventana 2s**	**Ventana 3s**
KNN	0.97	0.97	0.97
RF	0.99	0.99	0.99
SVC	0.98	0.97	0.98
NN	0.98	0.98	0.99

Tabla 4: AUC de modelos según ventanas temporales

Se obtienen los siguientes valores de Area por Debajo de la Curva y curvas ROC para cada modelo y ventana temporal.





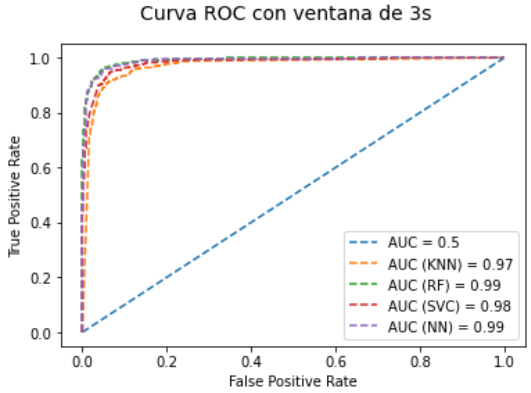


Tabla 5: ROC-AUC para cada modelo y ventana temporal

Al igual que usando la puntuación definida por la matriz de confusión, si nos fijamos en la AUC el mejor modelo es también bosques aleatorios, para el mayor tamaño de ventana de observación.

El modelo escogido como el que da mejores resultados es **bosques aleatorios (RF – Random Forest) con ventana de observación de 3 segundos.**

5. Conclusiones

El objetivo principal de este proyecto, definido en los primeros apartados, era crear un sistema de monitorización que permitiera detectar caídas con gran precisión. Basándonos en los resultados, y la alta puntuación obtenida de los modelos en base a el análisis de la sensibilidad y la especificidad de estos, se puede confirmar que el objetivo ha sido alcanzado. Se ha podido obtener un modelo basado en bosques aleatorios que clasifica las caídas como tales en más del 96% de los casos, valor que, comparado con otros estudios de la bibliografía, se considera elevado.

Como objetivo secundario, se había propuesto estudiar la viabilidad de crear un algoritmo para detección de fibrilación auricular, el tipo más común de arritmia. Se ha llegado a la conclusión que con los medios de los que se dispone no es posible desarrollar esta funcionalidad. Se necesitan datos médicos de usuarios sanos y de usuarios con este tipo de arritmia. No se ha podido encontrar ningún conjunto de datos público que contenga esta información. Los estudios de la bibliografía son mayoritariamente de grandes empresas (Apple) o universidades (Stanford), que disponen de recursos suficientes y bases de usuarios amplias para realizar sus estudios.

No obstante, como líneas de trabajo futuras, se propone contactar con alguna entidad, ya sea del ámbito tecnológico (como puede ser un gran fabricante de smartwatch, con una base de usuarios amplia), o médico (un hospital, que permita usar nuestro smartwatch para monitorización de pacientes). El objetivo sería crear una base de datos de pacientes sanos y pacientes con fibrilación auricular, en la que tengamos varias observaciones de PPG (Fotopleistograma) para cada uno de ellos, de manera que se pueda llegar a entrenar un modelo y crear una funcionalidad que alerte en tal caso. Tal como se ha introducido al inicio, el hecho de usar PPG no permite confirmar que haya una arritmia, ya que para eso se necesita hacer un electrocardiograma, pero sí que permite detectar una anomalía que puede ser suficiente para que una persona decida someterse a un electrocardiograma en un centro médico.

A nivel de realización del proyecto, se han podido alcanzar todas las entregas y objetivos planificados al inicio del trabajo. Este proyecto empezó en el contexto de una *start-up* en la que se quería sacar un producto al público, por lo que se marcaron unos tiempos lo más cortos posibles para poderlo comercializar. Finalmente, por motivos ajenos a la empresa el producto no llegó a salir, pero la fecha límite marcada se fijó para navidades, de manera que coincidía con las fechas propuestas para la finalización del trabajo de final de máster.

A modo de resumen, se considera que se ha alcanzado el objetivo principal, y que el producto tiene margen de mejora con la posibilidad de añadir otras funcionalidades, como la detección de fibrilación auricular.

6. Glosario

AUC: Area Under Curve
ECG: ElectroCardioGrama
ETL: Extract, Transform, Load
FA: Fibrilación Auricular
FN: False Negative
FP: False Positive
FPR: False Positive Rate
IDE: Integrated Development Environment
KNN: K-Nearest Neighbors
NN: Neural Network
PPG: PhotoPlethysmoGram
RF: Random Forest
ROC: Receiver Operating Characteristic
SVM: Support Vector Machine
TN: True Negative
TP: True Positive
TPR: True Positive Rate
UCI: UC Irvine

7. Bibliografía

Ã–zdemir, Ahmet Turan, y Billur Barshan. 2014. «Detecting Falls with Wearable Sensors Using Machine Learning Techniques.» *Sensors* 10691-10708.

Allen, John. 2007. «Photoplethysmography and its application in clinical physiological measurement .» *Pubmed* 28-39.

Angela Sucerquia, Jose David Lopez, y Jesus Francisco Vargas Bonilla. 2017. «SisFall: A Fall and Movement Dataset .» *Wearable Biomedical Sensors.*

Apple. 2018. *Apple.* Diciembre. Último acceso: Octubre de 2020. https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwipycLvk7HsAhUJLBoKHZO7AJcQFjAIegQIAhAC&url=https%3A%2F%2Fwww.apple.com%2Fhealthcare%2Fdocs%2Fsite%2FApple_Watch_Arrhythmia_Detection.pdf&usg=AOvVaw3qtY4tYU0ZbA8xxTAc-M99.

—. 2020. *Heart health notifications on your Apple Watch.* 17 de Septiembre. Último acceso: 13 de Octubre de 2020. https://support.apple.com/en-us/HT208931.

CSIC. 2018. *Envejecimiento en red.* Febrero. Último acceso: 26 de Septiembre de 2020. http://envejecimiento.csic.es/documentos/documentos/enred-estadisticasresidencias2017.pdf.

Defne Yilmaz, Yaniv Kerem, Stuart Crawford, David Benaron, Kristin Aschbacher, Jiaqi Liu, Megan Eaton, y otros. 2020. «Atrial fibrillation detection from raw photoplethysmography waveforms: A deep learning application.» *Heart Rhythm O2* 3-9.

Donges, Niklas. 2019. *A complete guide to the random forest algorithm.* 16 de Junio. Último acceso: 29 de Octubre de 2020. https://builtin.com/data-science/random-forest-algorithm.

Eduardo Casilari, Jose A. Santoyo Ramon, y Jose M. Cano Garcia. 2017. «UMAFall: A Multisensor Dataset for the Research on Automatic Fall Detection.» *Procedia Computer Science* 32-39.

Geoffrey Tison, Jose Sanchez, Brandon Ballinger, Avesh Singh, y Jeffrey Olgin. 2018. «Passive Detection of Atrial Fibrillation Using a Commercially Available Smartwatch.» *JAMA Cardiol* 409-416.

Gobierno de España. 2015. *Imserso.* Diciembre. Último acceso: 26 de Septiembre de 2020. https://www.imserso.es/imserso_01/documentacion/estadisticas/ssppmm_esp/2015/re/index.htm.

IBM. s.f. *Conceptos básicos de ayuda de CRISP-DM .* Último acceso: Septiembre de 2020. https://www.ibm.com/support/knowledgecenter/es/SS3RA7_sub/modeler_crispdm_ddita/clementine/crisp_help/crisp_overview.html.

Kirstin Aschbacher, Defne Yilmaz, Yaniv Kerem, Stuart Crawford, David Benaron, y Jiaqi Liu. 2020. «Atrial fibrillation detection from raw photoplethysmography waveforms: A deep learning application.» *Heart Rhythm O2* 3-9.

Malik, Farhad. 2018. *Medium.* Febrero. Último acceso: 17 de Octubre de 2020. https://medium.com/fintechexplained/what-is-grid-search-c01fe886ef0a.

Narkhede, Sarang. 2018. *Towards Data Science.* 26 de Junio. Último acceso: 30 de Octubre de 2020. https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5.

—. 2018. *Towards Data Science.* 9 de Mayo. Último acceso: 30 de Octubre de 2020. https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62.

SpaceMinds. 2019. *Github.* 09 de Enero. Último acceso: 15 de Octubre de 2020. https://github.com/SpaceMinds/FDS.

Valentina Cotechini, Alberto Belli, Lorenzo Palma, Micaela Morettini, Laura Burattini, y Paola Pierleoni. 2019. «A dataset for the development and optimization of fall detection algorithms based on wearable sensors.» *Data in Brief* 1-4.

Villaseñor, Lourdes Martinez, Hiram Ponce, Jorge Brieva, Ernesto Moya Albor, Jose Nuñez Martinez, y Carlos Peñafort Asturiano. 2019. «UP-Fall Detection Dataset: A Multimodal Approach.» *Sensors* 19-28.

Yichen Shen, Alireza Aliamari, Maxime Voisin, Anand Avati, Awni Hannun, y Andrew Ng. 2018. *Ambulatori atrial fibrillation monitoring using wearable photoplethysmography with Deep Learning.* Noviembre. Último acceso: 11 de Octubre de 2020. https://arxiv.org/abs/1811.07774#.

8. Anexo

La parte práctica del proyecto se ha realizado con Python. Puede ser encontrado en el siguiente enlace: https://github.com/arnautiendat/fall_detection.

En el repositorio puede encontrarse lo siguiente:

- **Test_files:** carpeta en la que se pueden encontrar 5 grabaciones de acelerómetro de caídas, y 5 de actividades diarias.
- **Load_scripts:** carpeta con los scripts de carga de los datos de los diferentes estudios.
- **Unified_datasets:** carpeta que contiene los datasets con las características calculadas.
- **Fall_detection.py:** script de detección de caídas.
- **Model_creation.py**: creación del modelo de detección de caídas a partir de los unified_datasets.