Reconocimiento de actividad mediante pulsera con sensores

Iván Aguilera Calle
Daniel García Moreno
Verónica Morante Pindado
Alejandro Pidal Gallego

Dirigido por:
Javier Arroyo Gallardo
Marlon Félix Cárdenas Bonett

Trabajo de fin de grado del Grado en Ingeniería Informática

Facultad de Informática
Universidad Complutense de Madrid
Curso 2016-2017

16 de junio de 2017

UNIVERSIDAD COMPLUTENSE MADRID

AUTORIZACIÓN PARA LA DIFUSIÓN DEL TRABAJO DE FIN DE GRADO Y SU DEPÓSITO EN EL REPOSITORIO INSTITUCIONAL E-PRINTS COMPLUTENSE

Los abajo firmantes, alumnos y tutor del Trabajo Fin de Grado (TFG) en el Grado en Ingeniería Informática y Grado en Ingeniería del Software de la Facultad de Informática, autorizan a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el Trabajo Fin de Grado (TFG) cuyos datos se detallan a continuación. Así mismo autorizan a la Universidad Complutense de Madrid a que sea depositado en acceso abierto en el repositorio institucional con el objeto de incrementar la difusión, uso e impacto del TFG en Internet y garantizar su preservación y acceso a largo plazo.

Título del TFG:
Reconocimiento de actividad mediante pulsera con sensores

Curso académico: 2016/2017

Nombres de alumnos:
Iván Aguilera Calle
Daniel García Moreno
Verónica Morante Pindado
Alejandro Pidal Gallego

Tutor/es del TFG y departamento al que pertenece:
Javier Arroyo Gallardo
Marlon Félix Cárdenas Bonett
Ingeniería del Software e Inteligencia Artificial

Firma de alumnos Firma del tutor/es

Agradecimientos

Queremos dar las gracias, en primer lugar, a nuestros tutores Javier Arroyo Gallardo y Marlon Cárdenas Félix, por haber confiado en nosotros y ayudarnos siempre que hemos tenido algún problema. Sin ellos este trabajo no hubiera sido posible debido a su dedicación y paciencia.

También agradecer especialmente a nuestro profesor Guillermo Jiménez por animarnos a hacer en su asignatura el primer diseño de nuestra aplicación, así como a nuestro compañero Jesús Menéndez por su colaboración en él.

Queremos mencionar y dar las gracias también a todas las personas que nos han ayudado a hacer las pruebas necesarias de este proyecto, tanto en la parte de entrevistas a usuarios como en la recogida de datos de las actividades.

Agradecemos también a todo el personal docente de la facultad que nos ha ayudado en nuestra formación y adquisición de los conocimientos necesarios a lo largo de los años para poder realizar este trabajo. También queremos hacer mención especial al personal de la biblioteca, ya que nos han ayudado mucho a lo largo de este proyecto.

Por último, agradecer a familiares, amigos y compañeros de la universidad que nos han acompañado a lo largo de estos cuatro años dándonos su ánimo y su apoyo.

A todos ellos, muchas gracias☺.

Índice

Índice de figuras

Índice de tablas

Índice de abreviaturas

API	Application Programming Interface
AR	Activity Recognition
ASD	Autism Spectrum Disorder
AST	Abstract Syntax Tree
BLE	Bluetooth Low Energy
CPU	Central Processing Unit
DFT	Discrete Fourier Transform
DGO	Diseño Guiado por Objetivos
DSI	Desarrollo de Sistemas Interactivos
FFT	Fast Fourier Transform
GATT	Generic Attribute Profile
GPL	General Public License
GPS	Global Position System
GSM	Global System for Mobile
HMM	Hidden MarkovModel
KNN	K-Nearest Neighbors
MAC	Media Access Control
MB	Mega Bytes
QDA	Quadratic Discriminant Analysis
RFFT	Real-valued Fast Fourier Transform
RMS	Root Mean Square
SI	Sistema Internacional
SVM	Support Vector Machines
TFG	Trabajo de Fin de Grado
UUID	Universal Unique IDentifier
WSGI	Web Server Gateway Interface

Resumen

Desde hace unos años, con el desarrollo de los smartphones y actualmente con el auge del internet de las cosas, la tecnología ha invadido gran parte de nuestra vida como en campos de la salud, el deporte o el bienestar. Este trabajo se centra en los dos últimos aspectos con el fin de hacer una app para smartphone de reconocimiento de actividad mediante sensores, con el que se busca conseguir que las personas lleven un control de las actividades que hacen a lo largo del día, pudiendo tener un reconocedor personalizado, a diferencia de otras aplicaciones comerciales.

Se trata de un trabajo que está compuesto por una parte de aprendizaje automático, que se ha llevado a cabo gracias a los árboles de decisión que sirven para la clasificación de las actividades y una app Android que puede tomar los datos de los propios sensores del smartphone o de una pulsera Texas Instruments que se conecta mediante Bluetooth al smartphone. En la app en la que se implanta el árbol de decisión que hará la clasificación, también puedes ver tu historial de actividades. También puedes registrar nuevas actividades mediante un servidor externo a la aplicación que se encarga de recibir datos de las nuevas actividades y de generar un nuevo clasificador que incorpora esa actividad, el cual es implantado automáticamente en la app para que sea capaz de renococer esta nueva actividad durante el uso de la app.

En este documento se describe el trabajo realizado, los sensores utilizados, las tecnologías empleadas, el proceso para la implementación de un reconocedor, el desarrollo de la app, los experimentos realizados, las conclusiones finales y por último las posibles mejoras que se pueden llevar a cabo en un futuro.

Palabras clave

Árboles de decisión, app Android, sensores, reconocimiento de actividad, aprendizaje automático, Internet de las cosas (IoT)

Abstract

For some years, with the development of smartphones and now with the rise of the internet of things, technology has invaded much of our lives as in fields of health, sports or well-being. This work focuses on the last two aspects to make a smartphone app for recognition of activity through sensors, which seeks to get people to take control of the activities they do throughout the day, being able to have a custom recognizer, unlike other commercial applications.

It is a work that is composed of a part of automatic learning, which has been carried out thanks to the decision trees that serve for the classification of activities and an Android app that can take data from the smartphone's own sensors or a Texas Instruments wristband that connects via Bluetooth to the smartphone. In the app that implements the decision tree that will do the classification, you can also see your activity history. You can also register new activities using a server external to the application that is responsible for receiving data from new activities and generating a new classifier that incorporates that activity, which is automatically implemented in the app to be able to renococer this new activity During the use of the app.

This document describes the work done, the sensors used, the technologies used, the process for the implementation of a recognizer, the development of the app, the experiments carried out, the final conclusions and finally the possible improvements that can lead to in the future.

Keywords

Decision Trees, Android app, sensors, activity recognition, Machine Learning, Internet of Things (IoT)

1 Introducción

1.1 Introducción al reconocimiento de actividades

El objetivo del reconocimiento de actividad (AR, activity recognition en inglés) es reconocer y diferenciar las distintas acciones o actividades que realiza una persona, basándose en una serie de observaciones previas que se consiguen realizando las distintas actividades que queremos que el sistema reconozca y almacenando en él los distintos datos de las actividades para, posteriormente, analizarlos a través de métricas y así entrenar o enseñar al sistema a distinguir unas actividades de otras.

En la actualidad, es un campo de investigación que ha atraído mucho la atención de distintos agentes, ya que sirve para prestar servicios de apoyo personalizado y está conectado con distintos ámbitos (entre ellos, el más destacado es el ámbito de la salud, pero también tiene una gran utilidad en los campos del bienestar, del deporte, de la domótica, control de rehabilitación de pacientes…).

Para comprender más fácilmente el campo de reconocimiento de actividad, describiremos un escenario en el que sería aplicable esta técnica:

Un hombre de edad avanzada, que actualmente vive solo en su casa, realiza distintas tareas a lo largo de un día: enciende el microondas para preparar su desayuno, pone el lavavajillas, enciende el gas para prepararse las comidas…. El hombre a veces olvida las tareas que tenía que realizar, pero tiene una aplicación que le recuerda, por ejemplo, que tiene que apagar el gas.

Su casa tiene implementada un amplio sistema de reconocimiento de actividad que registra todas las acciones que realiza el hombre a lo largo de sus días, de esta forma su hija puede acceder remotamente al historial de las acciones que ha realizado su padre a lo largo de los días y puede comprobar que está comiendo correctamente y que se toma su medicación acorde a las indicaciones y que su padre está en buenas condiciones, lo que le permite realizar un seguimiento y control.

En el escenario anterior vemos la gran importancia del reconocimiento de actividad en el ámbito de la salud. Nuestro trabajo va a tratar otros temas más orientados al deporte y al bienestar, gracias a ello personas que realizan deporte pueden mejorar su entrenamiento, fisioterapeutas pueden ser notificados de distintas lesiones o entrenadores pueden elegir su equipo en función del estado de cada persona. Gracias el reconocimiento se puede mejorar la técnica en cada deporte, por ejemplo, en el estudio [1] observando el acelerómetro en los distintos movimientos de braza al nadar, se pudo mejorar la técnica de natación. Nuestro foco va a estar centrado en distinguir los distintos ejercicios que realiza una persona.

Los smartphones han experimentado un gran avance, lo que ha propiciado, en gran parte, el aumento de estudios y proyectos relacionados con el reconocimiento de actividades. Actualmente, un smartphone comercial que puede estar a disposición de cualquier persona, proporciona al usuario un número elevado de sensores de bajo coste (giroscopio, acelerómetro, GPS, sensores de luz…), con los que podemos obtener distinta información del usuario con el fin de reconocer las distintas actividades que realiza, además de facilitarle sus tareas diarias. Hay que considerar, además, que actualmente, el smartphone es un objeto esencial en la vida de una persona, como lo puede ser las llaves de casa o la cartera con toda nuestra documentación, por lo que también es considerado como un dispositivo poco intrusivo para el reconocimiento de actividad. Es, por lo tanto, comprensible, que el desarrollo de aplicaciones relacionadas con la salud, el bienestar y el Internet de las Cosas haya aumentado considerablemente en los últimos años.

1.2 Motivación

A lo largo de los años, han florecido numerosos estudios relacionados con el reconocimiento de actividades. Aproximándonos al día de hoy, se puede observar un incremento por el interés de las tecnologías que se encargan de tal fin.

Por lo tanto, con la realización de este trabajo nuestra principal motivación es enfrentarnos al desarrollo de un sistema de reconocimiento de actividades y entender su funcionamiento, así como también adquirir distintos conocimientos en las tecnologías

punteras que se utilizan en el reconocimiento de actividad y adentrarnos en temas tan actuales como el análisis de datos y el aprendizaje automático.

El fin último es desarrollar una aplicación para móvil que destaque por su sencillez y que cualquier persona pueda ser capaz de utilizar de una manera satisfactoria, y que, además, sea capaz de reconocer y registrar una serie de actividades físicas que una persona realiza durante su vida diaria. El sistema también tendrá que permitir al usuario incorporar sus propias actividades para que posteriormente sean monitorizadas por el teléfono, lo cual resulta un tanto novedoso en este campo ya que no hemos encontrado estudios ni apps que implementen esta opción de permitir al usuario añadir sus propias actividades al sistema para su posterior reconocimiento y monitorización.

Creemos que este es un factor muy importante, ya que las personas que por ejemplo sufren algún tipo de discapacidad no podrían utilizar las aplicaciones existentes, ya que la mayoría son genéricas (es decir, para la mayoría de la sociedad) y no servirían para este otro sector de la población, ya que estas aplicaciones no serían capaces de reconocer, por ejemplo, cuando una persona con un tipo de discapacidad que hace que tenga que andar de manera muy distinta de como lo hace el resto de la población, quiera utilizar alguna de estas aplicaciones actuales, no conseguirían el resultado esperado, ya que no serían capaz de detectar que esta persona está andando. Nos gustaría dar la posibilidad a todas las personas de llevar un control de sus actividades y añadir cualquier actividad que no fuera convencional (por ejemplo: dar patadas en Capoeira, barrer, boxear etc…).

1.3 Objetivos

Una vez explicado de manera breve en los apartados anteriores en qué se basa el reconocimiento de datos y la motivación que nos ha llevado a realizar este trabajo, explicaremos los distintos objetivos que nos hemos marcado.

Como comentábamos en el apartado anterior, nuestro objetivo prioritario consiste en desarrollar e implementar una aplicación para Android que sea capaz de distinguir las distintas actividades que el usuario de la aplicación realiza (ya sea a través de los sensores del móvil o de una pulsera que se comunique con el móvil).

El objetivo secundario consiste en dar la posibilidad al usuario de añadir nuevas actividades a la aplicación, para que estas sean reconocibles cuando el usuario esté realizando la actividad que en un principio el sistema no era capaz de reconocer, proporcionando así una mayor personalización de la aplicación e incrementando así el rendimiento y disminuyendo la tasa de fallos del sistema, al incluir los datos personalizados realizando tus propias actividades.

De esta manera, nos adentraremos en el mundo de la clasificación de actividades, analizando de manera detallada los datos que obtenemos de distintos sensores para poder construir un clasificador generalista que clasifique de la manera más precisa posible. Realizaremos distintas pruebas donde la elección de distintos parámetros nos influirá en los resultados que posteriormente podamos obtener. Implementaremos, pues, una aplicación para dispositivos móviles Android con la que podamos clasificar y reconocer las actividades que esté realizando el usuario.

1.4 Implementación de los objetivos

Como explicamos en apartados anteriores, hemos elegido desarrollar la aplicación para el Sistema Operativo Android ya que es el que más extendido se encuentra actualmente entre los dispositivos móviles, está bastante documentado, es libre y toda la información se encuentra fácilmente disponible en la web, lo cual pensamos que nos facilitará el desarrollo de la aplicación, además el lenguaje utilizado es bastante similar a Java, por lo que tenemos experiencia en este tipo de lenguajes.

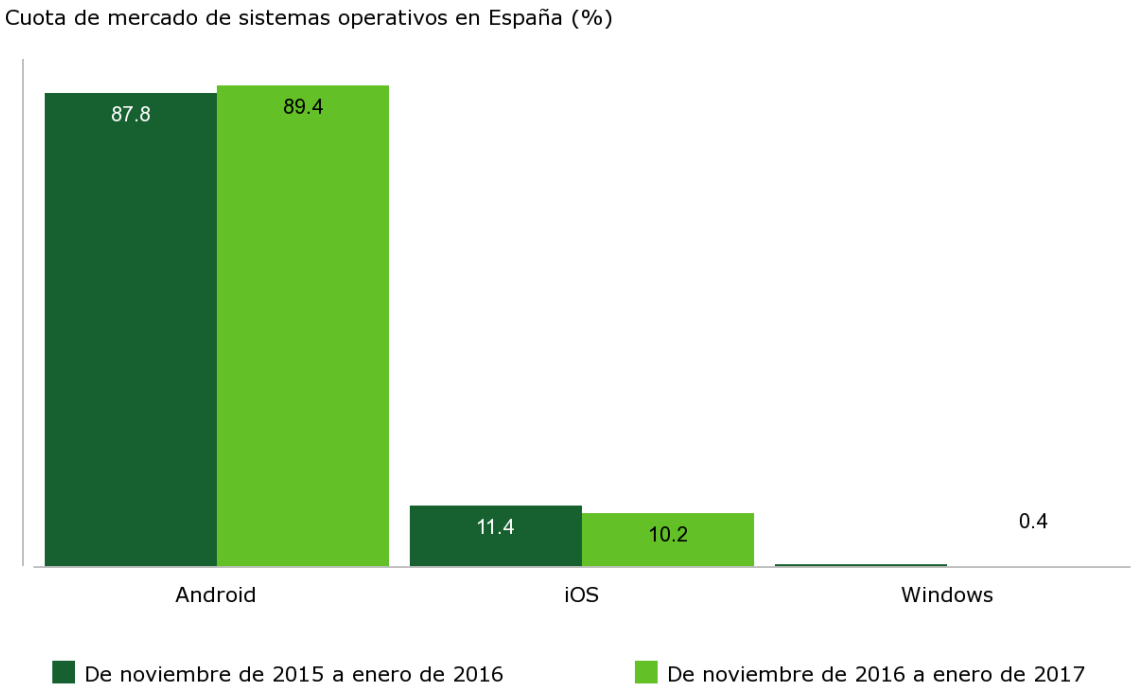


Figura 1.1: Diagrama de barras SO Móviles. Fuente: Kantar.es.

También hemos optado por permitir incorporar una pulsera al sistema de reconocimiento de actividades ya que, llevar el móvil colocado en alguna parte de nuestro cuerpo suele ser en general bastante molesto por su tamaño y peso, y como la pulsera es más cómoda, puede pasar por un objeto "Wearable", lo que al usuario no le entorpece en absoluto en la realización sus actividades del día a día. La pulsera también es muy útil para aquellos teléfonos que no posean giroscopios, así pueden obtener los datos y utilizar la aplicación sin depender de las características del teléfono.

Figura 1.2: SensorTagCC2650 TexasInstruments. Fuente: Rs-online.com.

Para poder conseguir los distintos objetivos necesitaremos aplicar distintas técnicas de análisis de datos y Machine Learning (como pueden ser los árboles de decisión o los Random Forest), los cuales iremos explicando en apartados posteriores.

En cuanto a la personalización de actividades, hemos optado por utilizar un servidor externo el cual genere el clasificador personalizado, con el fin de hacer más ligera la aplicación al ahorrar tiempo de cómputo.

1.5 Estructura del documento

El resto del documento se encuentra estructurado de la siguiente manera:

Para empezar, en el capítulo de **Introducción** iremos definiendo la motivación que nos ha llevado a realizar este proyecto y explicando los objetivos del mismo.

Después, continuaremos con el capítulo de **Estado del arte**. En él analizaremos la importancia que tienen hoy en día los detectores de actividades en la sociedad y comentaremos los tipos de dispositivos que hay para la detección de actividades. También explicaremos, para poner en contexto, el proceso genérico que se suele seguir para el reconocimiento de las actividades. Y para finalizar este capítulo, hablaremos de algunos estudios realizados anteriormente, los cuales analizamos para ver cómo se han realizado los sistemas de reconocimiento de actividades y qué técnicas, sus limitaciones, etc… se han utilizado en estos estudios.

A continuación, el capítulo **Implementación de reconocimiento de actividad**, hablaremos más en detalle de lo que es el proceso integral para el entrenamiento, una vez ya teniendo definida la aplicación de reconocimiento. Contaremos qué dispositivos hemos utilizado y cómo se ha realizado la obtención de datos. Además, explicaremos más detalladamente cómo es el procesado de esos datos y que técnicas hemos utilizado para la clasificación de los mismos.

Después, el capítulo **Aplicación Android** incluirá todo lo relacionado con la aplicación que hemos realizado para el trabajo, explicando cómo se ha realizado el diseño

e implementación con gran nivel de detalles. Detallaremos el esquema que tiene la aplicación, cómo realiza la recogida de datos y su posterior procesado. También definiremos cómo se realiza el entrenamiento de los mismos, así como todos los desafíos a los que nos hemos enfrentado al realizar la aplicación.

En el capítulo siete, **Experimentación** desarrollaremos las distintas pruebas que hemos realizado (a través de varios ejemplos) todo lo explicado en los capítulos anteriores, tanto en la parte realizada en pc como la parte realizada en real, es decir las pruebas reales del sistema final.

En el capítulo **Conclusiones y trabajo futuro**, expondremos los resultados totales obtenidos y pondremos de manifiesto posibles trabajos y/o mejoras a realizar en un futuro, tanto en el proceso de reconocimiento de actividad como en la aplicación Android.

Finalmente incluimos la sección de **Anexos** (que cuenta con una guía de uso de la aplicación Android), el **Glosario (**dónde explicamos distintos términos que ayuden a facilitar la comprensión del documento), la **Bibliografía** y, por último, las **Aportaciones** realizadas al trabajo por cada uno de los miembros.

2 Introduction

2.1 Introduction to activity recognition

The purpose of activity recognition (AR) is to recognize and differentiate the activities that a person performs, based on previous observations. Metrics and statistics are computed from sensor observations of different activities and used to train a system to classify future observations into the observed activities.

At present, it is a field of research that has attracted a lot of attention from different agents, as it serves to provide support services and is connected to different areas (among them, the most prominent in the field of health, well-being, sports, domotics...).

To understand the field of activity recognition, we will describe a scenario in which this technique can be applied:

An elderly man, currently living alone in his house, do multiple tasks in the course of a day: he turns on the microwaves to prepare his breakfast, he puts the dishwasher, he turns on the gas to prepare the meals.... The man, sometimes, forgets his tasks, but he has an application that remembers him, for example, to turn off the gas.

His house has implemented an activity recognition system that records all the actions that man do in the length of his days, so in his daughter can remotely access the history of the actions that his father has done to him to long days, and she can check that he is eating properly and he takes his medication according to the indications and that his father is in good condition, allowing her to follow up and control.

In the previous scenario we see the great importance of the recognition of the activity in the field of health. Our work will address other issues more oriented to sports and wellness, more precisely, sports practitioneers to keep record of their training. Thanks to the recognition can improve the technique in each sport, for example, in the study [1], watching the accelerometer in the various movements of breaststroke while swimming,

did improve the technique of swimming Our focus will be centered on distinguishing the Different exercises performed by a person.

The smartphones have a great advance, which has led, in large part, the increase of studies and projects related to the recognition of the activities. Currently, a commercial smartphone that can be available to anyone, provides the user with a high degree of low cost sensors (gyroscope, accelerometer, GPS, light sensors...), which can obtain information from the user with the objetive to recognize the different activities that he performs, besides facilitating his daily tasks. We must also consider that, currently, the smartphone is an essential object in a person's life, as can be the house keys or the portfolio with all our documentation. It is therefore understandable that the development of applications related to the health, well-being and Internet of Things has increased considerably in recent years.

2.2 Motivation

Throughout the years, numerous studies related to the recognition of activities have flourished. Approaching today, we can see an increase in the interest of the technologies that are responsible for such an end.

Therefore, with the accomplishment of this work our main motivation is to face us to develop a system of recognition of activities and to understand their operation, as well as to acquire different knowledge in the leading technologies that are used in the recognition of activity and to enter in subjects as current as data analysis and automatic learning.

The ultimate goal is to develop a mobile application that stands out for its simplicity and that anyone can be able to use in a satisfactory way, and that is also capable of recognizing and recording a series of physical activities that a person performs during his daily life. The system will also have to allow the user to incorporate their own activities so that they are later monitored by the phone, which is a bit new in this field since we have not found recent studies that implement this option to allow the user to add their own activities to the system for later recognition and monitoring.

We believe that this is a very important factor, since people who, for example, suffer some form of disability, could not use existing applications, since most of them are generic (ie for the majority of society) and would not serve this purpose another sector of the population, since these applications would not be able to recognize, for example, when a person with a type of disability that causes them to walk very differently than the rest of the population wants to use some of these current applications, would not get the expected result because they would not be able to detect that this person is walking. We would like to give all people the opportunity to keep track of their activities and add any activity that is not conventional (for example: kicking Capoeira, sweeping, boxing etc.…).

2.3 Targets

After explaining briefly in the previous sections on what is based on data recognition and the motivation that has led us to perform this work, we will explain the different goals we have set.

As we mentioned in the previous section, our priority objective is to develop and implement an Android application that is able to distinguish the different activities that the user of the application performs (either through the sensors of the mobile or a bracelet that is communicate with the mobile).

The secondary objective is to give the user the possibility of adding new activities to the application, so that they are recognizable when the user is performing the activity that the system was initially unable to recognize, thus providing a greater customization of the application. This increasing performance and decreasing the system failure rate by including custom data by performing your own activities.

In this way, we will enter on world of the classification of activities, analyzing in detail the data we obtain from different sensors in order to build a generalist classifier that classifies as precisely as possible. We will perform different tests where the choice of different parameters will influence the results that we can later obtain. We will implement

an application for Android mobile devices with which we can classify and recognize the activities that the user is doing.

2.4 Development of targets

As we explained in previous sections, we have chosen to develop the application for the Android Operating System since it is the most widespread is currently among mobile devices, is well documented, is free and all information is readily available on the web. We think will facilitate the development of the application, in addition the language used is quite similar to Java, so we have experience in this type of languages.

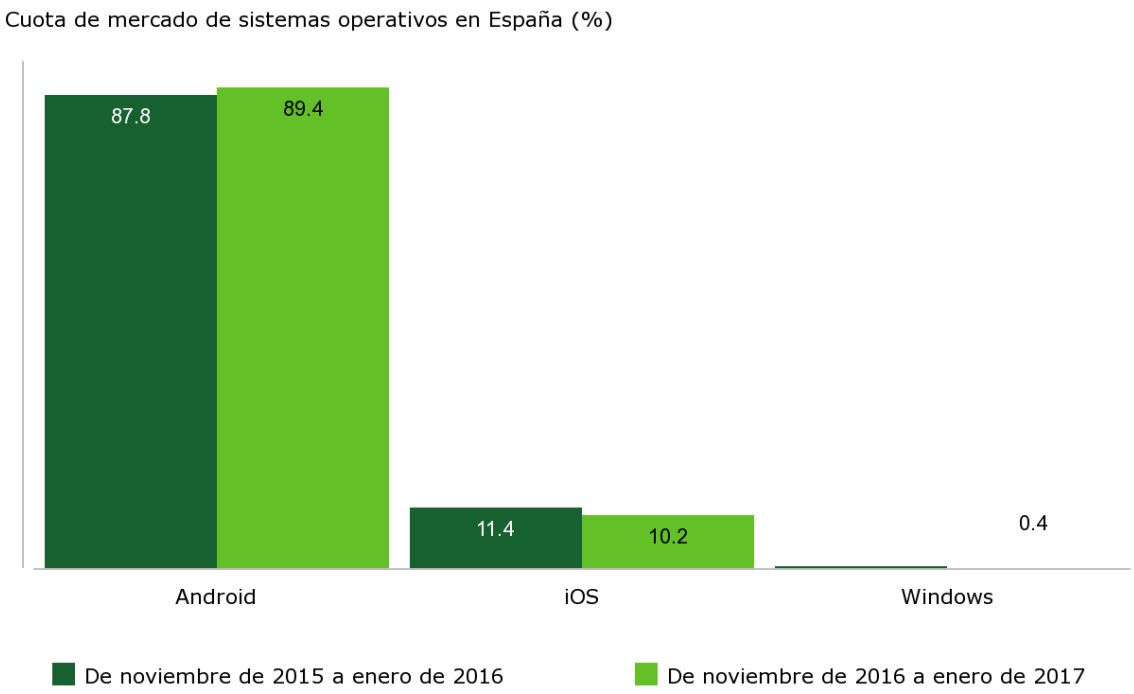


Figura 2.1: Bar Chart of Smartphones OS. Src: Kantar.es.

We have also opted to allow a bracelet to be incorporated into the activity recognition system since, carrying the mobile placed somewhere in our body is generally quite annoying because of its size and weight, and because the bracelet is more comfortable, it can pass through a "wearable" object, which the user does not obstruct at all in carrying out their day-to-day activities. The bracelet is also very useful for those phones that do not have a gyroscope, so you can get the data and allows to use the application without depending on the characteristics of the phone.

Figura 2.2: SensorTagCC2650 TexasInstruments. Src: Rs-online.com.

In order to achieve the different objectives we will need to apply different techniques of data analysis and Machine Learning (such as decision trees or Random Forest), which we will explain in later sections.

2.5 Document structure

The rest of the document is structured as follows:

To begin, in the chapter of **Introduction** we will define the motivation that has led us to carry out this project and explaining the objectives of it.

Then we will continue with the **State of Art** chapter. In it we will analyze the importance that today have the detectors of activities in the society and we will discuss the types of devices that exist for the detection of activities. We will also explain, to put into context, the generic process that is usually followed for the recognition of activities. And to conclude this chapter, we will talk about some previous studies, which we analyze

to see how the systems of recognition of activities have been realized and what techniques have been used in these studies.

Then, in the chapter **Implementation of recognition of activity**, we will speak more in detail of what is the integral process for training, once already having defined the application of recognition. We will tell you which devices we have used and how the data collection has been performed. In addition, we will explain in more detail how is the processing of these data and which techniques we have used to classify them.

Afterwards, in the **Android Application** chapter will include everything related to the application we have done for the job, explaining how the design and implementation have been done with detail. We will detail the scheme that the application has, how it performs data collection and its subsequent processing. We will also define how the training is performed, as well as all the challenges that we have faced in implementing the application.

In chapter seven**, Experimentation**, we will develop the various tests that we have done (through several examples) all explained in the previous chapters, both in the part realized in PC as the part realized in real, that is to say the actual tests of the final system.

In the chapter Conclusions and future work, we will present the total results obtained and show possible work and / or improvements to be made in the future, both in the activity recognition process and in the Android application.

Finally, we include the Attachments section (which has a guide to use the Android application), the Glossary (where we explain different terms that help to facilitate the understanding of the document), Bibliography and, finally, Contributions made to work by each of the members.

3 Estado del arte

En este capítulo se enuncia la importancia del reconocimiento de actividades y su uso en diferentes ámbitos. Se mencionan los diferentes dispositivos que actualmente existen y la diferencia entre ellos. También se relata de manera muy breve el proceso que se sigue para llegar al resultado final (se detalla de manera más exhaustiva en el capítulo cuatro). Por último, se mencionan los artículos que nos han ayudado a entender mejor este proceso.

3.1 Importancia

En los últimos tiempos, el gran progreso que han vivido los smartphones y el avance en la cantidad del número de sensores de los que disponen ha supuesto un incremento en la detección de actividades cotidianas, ha contribuido a mejorar la vida diaria de las personas y a poder obtener más información sobre su salud o su vida deportiva. También ha entrado de lleno en áreas como la de la seguridad o el entretenimiento.

Estos nuevos avances presentan novedades para la atención de la salud de las personas. Pueden detectar la caída de una persona mayor (se puede estudiar sus trayectorias para poder obtener diferentes conclusiones) o detectar el autismo a jóvenes. También puede ayudar al usuario de forma proactiva para tener un buen hábito de actividad física, ya que su uso en dispositivos móviles permite monitorizar los niveles de actividad, gasto de energía…

Estos sistemas de detección comenzaron centrando el foco de investigación inicial en la detección de movimiento mediante cámaras [2]. La tendencia actual ha seguido la línea de la detección inercial de movimientos con dispositivos que puede llevar el usuario en el cuerpo. A día de hoy, los teléfonos móviles son la principal fuente para el reconocimiento de actividad.

Como se comentaba en el primer párrafo, los smartphones han vivido un gran desarrollo desde su aparición. La mayoría de ellos están equipados con varios sensores, como puede ser el acelerómetro, el GPS, el giroscopio, sensores de luz, entre otros.

Dichos sensores tienen un bajo coste que no encarece el precio del producto final, lo que convierte a los smartphones (portadores de muchos sensores) en la principal plataforma para el reconocimiento de actividad.

En un futuro, la cantidad de sensores que contengan los teléfonos móviles aumentará, permitiendo añadir sensores de humedad o sensores de gas (detector de concentración de C02, por ejemplo), entre otros. La combinación de estos sensores permite la obtención de datos de manera muy precisa. Por ejemplo, combinando el acelerómetro con el micrófono podemos saber si una persona está sentada hablando con otra persona. La señal GSM nos puede decir sobre un 80% de exactitud si una persona está en un vehículo moviéndose o parado. La combinación de sensores o "fusión de sensores" es un aspecto que comentaremos a continuación en el apartado de "Elementos para la captación de datos".

3.2 El proceso del reconocimiento de actividad

El reconocimiento de la actividad, tal y como hemos comentado en la introducción, es la tarea de diferenciar distintas acciones o actividades que realiza una persona. Este proceso, de forma generalista, tiene una serie de pasos [3], los cuales los describimos de manera resumida a continuación:

1. **Sensing**: en este primer paso se recoge la información de los sensores con una determinada frecuencia de muestreo.
2. **Preprocesado**: los datos recogidos en el paso anterior son procesados (eliminación de ruido, recalibración, ect). Luego se aplica una ventana deslizante o se segmentan.

Figura 3.1: Ventana deslizante. Fuente: 2.bp.blogspot.com

3. **Extracción de características**: se extraen determinadas características de los datos segmentados en el paso anterior. Las que más interesen según que actividades queremos clasificar. Estas características pueden ser de tipo estadísticas sobre el valor del sensor, como puede ser la desviación típica, por ejemplo, o sobre el dominio de la frencuencia, como puede ser calcular la transformada de Fourier discreta.
4. **Entrenamiento**: antes de que un sistema de reconocimiento de actividades pueda empezar a ser utilizado, previamente hay que entrenar a sus clasificadores. Este entrenamiento puede ser offline (en una máquina de escritorio) u online (en el propio dispositivo móvil). En el entrenamiento, los datos en bruto de los sensores son recolectados, etiquetados con la actividad correspondiente, y almacenados para obtener, posteriormente, los parámetros del modelo. Esto es, tras haber elegido una técnica de clasificación, realizamos un entrenamiento con el objetivo de encontrar un conjunto de parámetros que nos permita obtener una buena tasa de aciertos de clasificación, intentando que sea lo más generalista posible (que generaliza más allá del conjunto de datos de entrenamiento) y evitando a toda costa el sobreajuste. Para realizar esta tarea podemos apoyarnos en algoritmos como KNN, Random Forest, árboles de decisión… El paso de entrenamiento de explicará con más detenimiento en el capítulo 4.
5. **Clasificación**: en este último paso, los clasificadores ya están entrenados, y por tanto, listos para usarse en la clasificación de las distintas actividades.

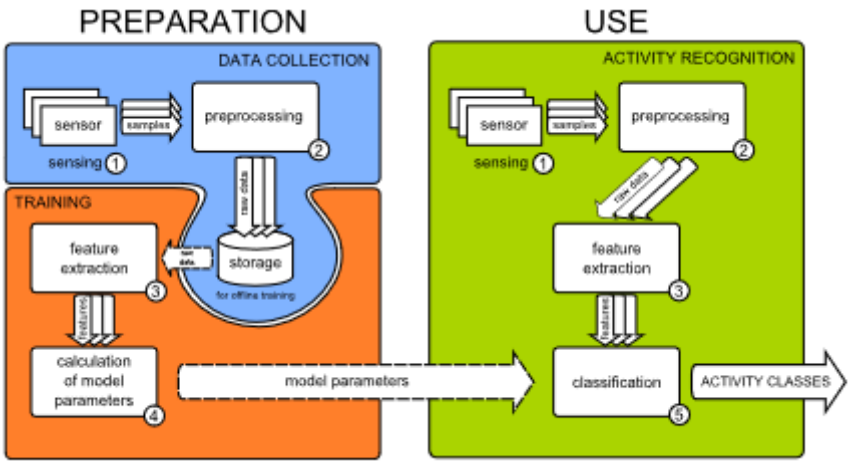


Figura 3.2: Proceso del reconocimiento de actividad [3]

3.3 Elementos para la captación de datos

Para poder obtener datos en brutos, disponemos de pequeños elementos llamados sensores. Un sensor [4] es un dispositivo que capta magnitudes físicas y las transforma en señales eléctricas. Existen multitud de sensores, cada uno con su magnitud de medida, algunos tienen mayor precisión que otros. Coexisten diversas clasificaciones de sensores [4], como por ejemplo, por su magnitud. En esta línea tenemos sensores que miden la aceleración, la fuerza, la presión, la temperatura, de visión artificial (cámaras de vídeo), de proximidad, de luz…

Dejando de lado la magnitud, en general un sensor es de uno de los tipos que se describen a continuación [5]:

- **Sensores físicos**: pieza de hardware que se encuentra alojado en algún dispositivo.
- **Sensores sintéticos**: combinación de varios sensores que conforman un sensor único.

Un sensor nos proporciona datos que pueden tener tres tipos de naturaleza:

- **Datos en bruto:** son los datos que obtenemos directamente de los sensores.
- **Datos calibrados:** datos en bruto a los que, a partir de diferentes algoritmos de corrección, se mejora la calidad de los mismos, como puede ser la reducción de ruido o la compensación de la deriva.
- **Datos combinados:** datos que vienen de diferentes sensores y se combinan para formar un nuevo tipo de dato. Por ejemplo, para obtener los datos de la aceleración

de la gravedad necesitamos combinar los datos que nos proporcionan sensores como el del acelerómetro y del giroscopio.

Haciendo más énfasis en el mundo móvil, de la gran cantidad de sensores de los que disponemos, nos centraremos en los siguientes tipos de sensores:

- **Sensores de movimiento:** captan cualquier tipo de fuerza que sea potencialmente responsable de crear movimientos. Estos tipos de sensores trabajan sobre tres ejes imaginarios (eje X, eje Y y eje Z). Dentro de este segmento, se encuentran sensores tan conocidos como el acelerómetro, el giroscopio, detector de pasos…
- **Sensores de posición:** sensores que se encargan de medir la posición de un objeto. Dentro de esta categoría se encuentran sensores como el magnetómetro, los sensores de proximidad...
- **Sensores de ambiente:** sensores que son responsables de medir las propiedades del medio ambiente donde se encuentran. Propiedades tales como la temperatura, la humedad, la cantidad de luz, la presión...

Cobran bastante importancia dos sensores en concreto, el giroscopio y el acelerómetro (sensores que se encuentran, precisamente, dentro del grupo de sensores de movimiento). El acelerómetro nos permite medir las aceleraciones que sufre un objeto. Como pertenece al grupo de los sensores de movimiento, este trabaja sobre tres ejes ortogonales entre sí (eje X, eje Y y eje Z).

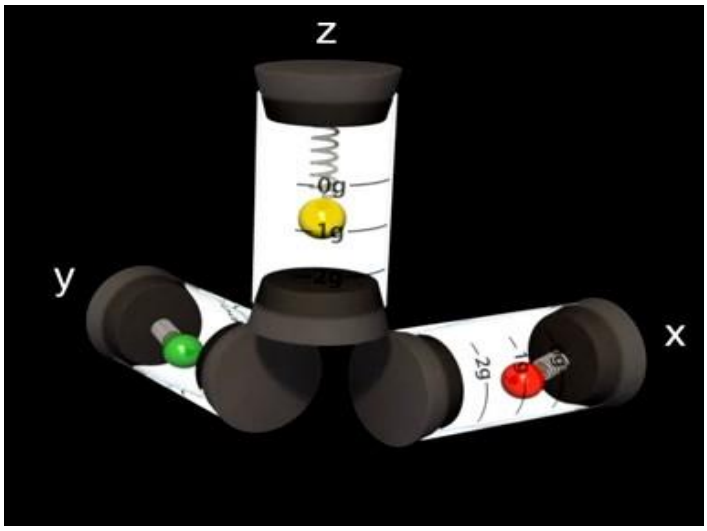


Figura 3.3: Sensor acelerómetro. Fuente: Oscarliang.com

Por otro lado, el giroscopio nos permite conocer la orientación del objeto en el espacio. La combinación de ambos sensores nos permite conocer la posición en el plano de un objeto que posea dichos sensores [6]. Podremos saber la aceleración que toma cuando lo movemos, y hacia qué lado lo estamos girando.

Figura 3.4: Ejes giroscopio en un teléfono. Fuente: Josueggh.com

Figura 3.5: Ejes acelerómetro en un teléfono. Fuente: Oscarliang.com

El acelerómetro ha sido un sensor que tiene a sus espaldas gran cantidad de estudios de investigación. No menos importante, otros sensores, como el giroscopio o el magnetómetro están empezando a ser utilizados, en combinación con el acelerómetro, para obtener mejores resultados. Esta "fusión de sensores" requiere de un estudio amplio, pues tiene implicaciones directas en aspectos como el consumo de batería [7]. Este problema hace preguntarse a los investigadores qué sensores utilizar (combinados o no)

y cuándo es el momento indicado para que su utilización sea la mejor opción. Diversos estudios contenidos en el artículo anterior obtienen conclusiones diversas, como que la combinación de acelerómetro y giroscopio en conjunto mejora, en un cierto porcentaje, la precisión a la hora de reconocer actividades. En contra, en el artículo anterior también reflejan que existen personas que afirman todo lo contrario, que dicha combinación no mejora en nada los resultados. Este mar de distintos resultados refleja que la manera de organizar los experimentos para realizar los estudios da lugar a posibles conclusiones diferentes. Hay que añadir, que la fusión de sensores es un frente de investigación el cual no ha sido muy explorado. El artículo [7] investiga sobre el tema de combinación de sensores, concluyendo que la combinación del acelerómetro y del giroscopio proporciona buenos resultados, pues se complementan bien entre sí. Se aclara que, con la combinación de ambos sensores se consigue una alta precisión, pero que añadir nuevos sensores no provocará mejora alguna.

Otra alternativa a la fusión de sensores es el uso dinámico y adaptativo de estos [3]. Esta funcionalidad viene motivada por el consumo energético de varios sensores funcionando a la vez. La idea consiste en conectar y desconectar, de manera adaptativa, los sensores a la hora de reconocer la actividad para realizar un reconocimiento eficientemente energético. Por poner un ejemplo, en [3] se citan estudios donde se utilizaron de manera conjunta el acelerómetro y el GPS. El sensor GPS solamente se activaba cuando el usuario estuviera realizando actividades al aire libre. Por lo tanto, el GPS se activa y desactiva de manera dinámica dependiendo de la ubicación del usuario, utilizando el contexto del buen funcionamiento del acelerómetro en ciertas situaciones, en este caso, en interiores.

Otro aspecto importante a tener en cuenta relacionado con los sensores es la frecuencia de muestreo, pues también tiene un efecto directo sobre el consumo de batería [3]. La elección de una tasa de muestreo correcta es una tarea importante a la hora de realizar reconocimiento de actividad. Esta elección debe adaptarse al tipo de actividades que vayamos a reconocer. Por ejemplo, para reconocer movimientos humanos complejos se debe de utilizar tasas de muestreo altas para evitar perder demasiada información sobre los movimientos que realiza la persona. Relacionando este aspecto con nuestro proyecto, al reconocer nosotros actividades humanas, utilizamos una frecuencia de 10 Hz, ya que

diversos estudios coinciden en que las frecuencias idóneas para reconocer actividades humanas se encuentran entre 0 y 20 Hz [8]. Utilizar tasas de muestreo que luego no van a resultar útiles tiene una implicación negativa en el gasto de batería, aspecto bastante importante en dispositivos de batería limitada, como es el caso de los smartphones y para la implantanción del reconocimiento durante periodos cercanos al día.

3.4 Características de los datos usadas para el proceso de clasificación

Las características de las que se va a estar hablando son las medidas estadísticas que se utilizan durante todo el proceso.

El artículo [3] nos muestra una tabla donde se muestran las características utilizadas para el reconocimiento de las actividades:

Tabla 1: Características utilizadas en distintos estudios

Study	Activities	Data Features	Phone
[38]	A1, A2, A3, A5	mean, SD, number of peaks	Nokia N95
[41]	A1, A5, A8, A9, A10	A's mean, VAR, FFT coefficients and GPS speed	Android Phone, Nokia N95
[42]	A1, A5, A8, A11	SD (based on accelerometer magnitude)	Nokia N95
[60]	A1, A2, A3, A6, A9, A16, A17, phone in hand, typing text messages; talking on the phone	mean, VAR	OpenMoko Neo Freerunner
[51]	A1, A5, A6, A7, A12, A17, Idle	Fundamental frequency, average acceleration, max and min amplitude (based on accelerometer magnitude)	Motorola Droid
[36]	A1, A5, A8, A9, A11	mean, VAR, mean crossing rate, spectrum peak, sub-band energy, sub-band energy ratio, spectral entropy	Nokia N95, iPhone
[32]	A1, A5, A8, A9, A11	A's VAR, DFFTcomponents and GPS speed	Nokia N95
[45]	A1, A4, A5, A12	similarity score using geometric template matching algorithm	Android phones
[31]	A1, A5, A8, A10	mean, VAR	Android Nexus One
[54]	A1, A2, A3, A5, A6	mean, root mean square, difference between max and min values	Android phones
[55]	Different physical activities	maximum and minimum euclidean norm	ZTE Blade
[59]	A1, A2, A3, A4, A5, A9	signal magnitude, coefficient of variance, counts per minute	Samsung Galaxy S
[50]	A1, A2, A3, A5	mean, min, max, SD	Samsung Galaxy Gio
[34]	A1, A3, A5	mean, VAR, SD, correlation between axes, inter-quartile range, mean absolute deviation, root mean square and energy	HTC Evo 4G
[35]	A1, A2, A3, A5, A6, A7, A9, A10, A12, A16 (prone, supine)	mean (axis, magnitude), SD (axis, magnitude), tilt, linear regression coefficients, wavelet coefficients	HTC G11, Samsung i909
[48]	A1, A5, A15, A16, WD, IR, BT, HD, FTT, BRD, unknown	A's VAR, MFCC (Mel-frequency cepstral coefficient), RMS (root mean square), ZCR (zero-crossing rate) as acoustic features.	Android phone
[40]	A1, A2, A4, A6, A7	peak, SD/mean, FFT energy	HTC Hero
[52]	A1, A2, A3, A5, A9, A10	21 features, including mean, SD, min, max, 5 different percentiles and observations below/above these percentiles	Samsung Galaxy Mini, Nokia N8
[49]	A1, A1 (power), A4, A8	For details, refer to [49]	iPhone
[43]	A1 (slow), A2, A3 (relax, normal), A7, A13, A14	Mean, VAR, magnitude, covariance, FFT energy and entropy	Nokia N95, Samsung Galaxy S2
[44]	A1, A2, A3, A6, A7, A16	For details, refer to [44]	Samsung Galaxy S2
[33]	A1, A5, A6, A7, A8, A9, A10	9 features based on the auto-correlation function of accelerometer signals	Samsung Galaxy Y
[58]	A1, A2, A5, A6, A7, A12	Auto-regressive coefficients	LG Nexus 4
[47]	A1, A5, A6, A7, A8	SD and auto-regressive fitting of y-axis, correlation of x, y, z, signal magnitude area, mean, SD and skewness of the pitch	Android smartphone
[37]	A1(slow, normal, rush), A2, A3, A5	Mean, VAR, zero crossing rate, 75th percentile, correlation, inter-quartile, signal energy, power spectrum centroid, FFT energy, frequency-domain entropy	Google Nexus S
[39]	A1, A2, A3, A5, A9, A10, A17	SD, min, max, the remainder between percentiles (10, 25, 75, 90), median, the sum, square sum and number of crossings of values above or below the percentile (10, 25, 75 and 90)	Nokia N8, Samsung Galaxy Mini
[53]	A1, A2, A3, A5, A12, A16	mean, SD	iPhone 4S
[56]	A1, A4, A6, A7, A9	time gap peaks, mean, SD, A's energy, Hjorth mobility and complexity	HTC Nexus
[57]	A1, A4, A6, A7, A8, A9, A13, A14	Average period, VAR, average energy, binned distribution for each axis and correlation between y and z	Samsung Nexus S
[46]	A1, A5, A1/A5 on treadmill, A6, A7, A9, A10, A11, A12, A13, A14, A15, idle (A2/A3), watching TV	mean, SD, correlation, signal magnitude area, auto-regressive and moving average coefficients for A; altitude difference for pressure sensor; mean, VAR, min and max for audio sensor	LG NEXUS 4

Activities: walking, A1; standing, A2; sitting, A3; jogging, A4; running, A5; walking upstairs, A6; walking downstairs, A7; still, A8; biking, A9; driving a car, A10; in vehicle, A11; jumping, A12; using elevator up, A13; using elevator down, A14; vacuuming, A15; laying, A16; phone on table/detached, A17; washing dishes, WD; ironing, IR; brushing teeth, BT; hair drying, HD; flushing the toilet, FTT; boarding, BD; unknown

Durante la fase de preprocesado del reconocimiento de actividades se extraen varias características de los datos aportados por los sensores. Estas características son usadas durante el entrenamiento y fase de prueba de los métodos de clasificación implementados.

Hay dos tipos de características de los datos, ambos usados en el reconocimiento de actividades:

- Tiempo.
    - Media, mediana, varianza, desviación típica, máximo, mínimo, etc.

- Dominio de la frecuencia.
    - FFT, RMS de Fourier, RFFT, DFT, etc….

El coste computacional del dominio del tiempo es más ligero que el del dominio de la frecuencia por el cálculo extra de la transformada de Fourier. Esto provoca que las características de tipo Dominio de Frecuencia no puedan ser factibles en sistemas en tiempo real (pese a que este tipo de características suelen comportarse bastante bien obteniendo buenos porcentajes de acierto en actividades específicas, como puede ser subir o bajar escaleras).

3.5 Entrenamiento Online vs entrenamiento Offline

Una vez se obtienen datos en crudo utilizando los sensores necesarios y con una tasa de muestreo adecuada, se realiza el proceso de entrenamiento. Se debe de realizar un correcto entrenamiento para que posteriormente los clasificadores funcionen de manera correcta. Como se indicaba en el apartado del Proceso de reconocimiento de la actividad de este mismo capítulo, existen dos formas de entrenamiento: online u offline. En un entrenamiento Online el proceso se realiza en tiempo real en el propio dispositivo móvil. Por el contrario, en el entrenamiento Offline, el proceso se realiza en un dispositivo ajeno al clasificador, normalmente en un ordenador de escritorio al cual se le pasan los datos en bruto, donde se manejan para poder realizar el entrenamiento.

La mayoría de estudios que se han realizado sobre métodos de entrenamiento han coincidido en la forma offline para su realización. Esto se debe a que la parte de

entrenamiento requiere de una computación bastante elevada, sobre todo si se manejan miles de datos o técnicas de clasificación cuyo ajuste de parámetros requiera de cálculo intensivo o del uso de algoritmos de optimización metaheurísticos (como los algoritmos genéticos).

Por otro lado, el proceso de la clasificación, sí que se realiza en la mayoría de los casos de manera online en tiempo real en el propio smartphone.

3.6 Independencia de la orientación y posición en el reconocimiento de actividades

Los resultados generados por el reconocimiento de actividad son susceptibles a algunos cambios en la orientación de los sensores. El uso del acelerómetro obliga a los usuarios a colocar el dispositivo que contenga los sensores en una orientación concreta, lo que limita la libertad de los mismos a la hora de utilizar el dispositivo. Del mismo modo ocurre con la posición del dispositivo, que en la mayoría de los casos debe ser fija.

Existen una serie de métodos con el que poder ser independientes en la orientación y posición de los sensores [3]:

- En el caso de la orientación, disponemos de dos métodos para la independencia:
  - Utilización de características independientes de la orientación. Por ejemplo, se calculan las características basándonos en la magnitud del acelerómetro (valores absolutos) en vez de en los valores de sus tres ejes individuales, ya que la magnitud del acelerómetro es menos sensitiva a los cambios de orientación (valores a los que no les interfiere que el teléfono móvil se encuentre en el bolsillo mirando hacia la pierna de la persona o hacia el pantalón).
  - Realizar la transformación de la señal obtenida de los sensores. Por ejemplo, el sistema de coordenadas de los dispositivos móviles se transforma un sistema de coordenadas global de la Tierra para contrarrestar los cambios de orientación.

- En caso de la posición, disponemos de cuatro métodos para obtener una posible independencia:

- **Método 1:** entrenar el clasificador del sistema de reconocimiento usando datos de los sensores colocados en todas las posiciones posibles.
- **Método 2:** similar al método 1. Entrenar al clasificador con datos de todas las posiciones posibles, añadiendo, además, un análisis discriminante tras la extracción de datos.
- **Método 3:** utilizar diferentes subclases para las distintas posiciones que pueden tener los sensores en el cuerpo. Por ejemplo, dentro de la actividad "caminar", creamos distintas subclases de caminar para las diferentes posiciones, como puede ser llevar los sensores en la mano, en el pantalón, en el brazo… El clasificador, a bajo nivel considera todas estas posiciones como actividades diferentes, pero a alto nivel, todas corresponden a la actividad "caminar".
- **Método 4:** utilizar un clasificador distinto para cada posición. Esto requiere, que, en tiempo real, primero haya que detectar la posición para luego utilizar el clasificador adecuado.

3.7 Resultados obtenidos utilizando diversos clasificadores

En la actualidad, existen multitud de clasificadores con el que poder reconocer actividades. Por nombrar algunos conocidos, disponemos de árboles de decisión, SVM (Support Vector Machines), KNN (K-nearest neighbors), Naive Bayes, QDA (Quadratic Discriminant Analysis), tablas de decisión… Cada clasificador tiene sus ventajas y desventajas, por lo que dependerá del tipo de actividad que vayas a analizar, el coste o las métricas que utilices. Para realizar una primera aproximación a la elección de un clasificador realizamos la lectura de diferentes estudios, donde se hacen pruebas con distintas personas y con diferentes actividades.

En este apartado, comentaremos el estudio [9], donde se utilizaron dos de los clasificadores anteriormente nombrados: KNN y QDA.

La prueba realizada consistió en la evaluación de la exactitud con el cual estos dos clasificadores clasificaban. Para ello, se utilizó dos teléfonos móviles (Nokia N8 y un teléfono Samsung Galaxy Mini). La ubicación del dispositivo sería igual para todas las

personas que participaran en la prueba. Para el teléfono Nokia N8, ocho sujetos llevaron el dispositivo en el bolsillo delantero del pantalón. En el caso del teléfono de la marca Samsung, fueron seis los sujetos que llevaron el dispositivo en la misma ubicación. El sensor de los teléfonos utilizados fue el acelerómetro (en sus tres ejes), el cual trabajó a una frecuencia de 40Hz. Respecto al tema del procesamiento de los datos, se trabajó con ventanas deslizantes cuyo tamaño de ventana eran 300 observaciones. De los datos obtenidos, en total se extrajeron veintiuna características. Entre ellas, algunas fueron la desviación típica, la media, el mínimo y el máximo, el uso de percentiles (cinco percentiles: 10, 25, 50, 75 y 90), la suma de cuadrados…

Para poder clasificar las actividades se utilizó la siguiente taxonomía, la cual presenta la siguiente forma:

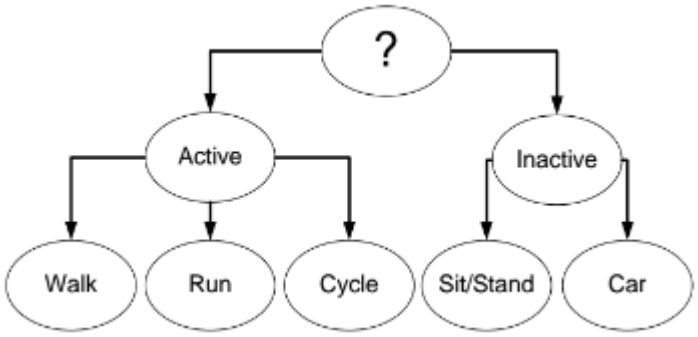


Figura 3.6: taxonomía de actividades [9]

Se agrupan las actividades en dos grandes grupos: en las que el usuario está activo y en las que el usuario está inactivo.

Los resultados que se obtuvieron con el Nokia N8 fue de una exactitud del 95,4% por parte del clasificador QDA y de un 94,5% por parte del clasificador KNN. A primera vista, se puede observar que los resultados obtenidos por ambos clasificadores son muy parejos. El KNN utiliza menos porcentaje de CPU que su adversario. Este aspecto es importante a la hora de elegir un clasificador. El estudio refleja que el uso de estos clasificadores no supone un uso excesivo de batería por lo que se puede implementar aplicaciones móviles que reconozcan actividades y funcionen en segundo plano

permitiendo al usuario seguir usando su smartphone con otras aplicaciones, como puede ser juegos.

Otro aspecto común que han sufrido ambos clasificadores es la confusión de actividades, por ejemplo, caminar y montar en bicicleta. Aquí se nos presenta un problema muy común en el reconocimiento de actividad (la confusión). Este problema viene motivado a que, por ejemplo, la manera de caminar que tiene cada persona hace de esta actividad difícil de reconocer, por lo que disponer de una gran cantidad datos antes de realizar el entrenamiento es un punto a favor a la hora de obtener buenos resultados y poder evitar el problema de la confusión.

En la prueba realizada con el dispositivo Samsung, el ciclismo también se confunde, de manera débil, con caminar. En general, el rendimiento ofrecido por el dispositivo de Samsung es superior al Nokia, debido a que el Galaxy Mini dispone de una potencia mayor. Un problema a tener en cuenta es el uso de la batería. Aunque en este estudio se consigue una duración de la batería de 24 horas ejecutándose en segundo plano, es un aspecto que no hay descuidar, adaptando el sistema de reconocimiento, por ejemplo, utilizando una tasa de muestreo lo más baja posible.

3.8 Consumo de recursos en reconocimiento de actividad

Los sistemas de reconocimiento de actividad, como cualquier otro sistema informático, requieren del uso de CPU, utilización de memoria y consumo de energía para poder realizar su función. A la hora de implementar un sistema de reconocimiento hay que realizar un análisis del consumo de estos recursos. Este análisis se realiza para conocer si un sistema de reconocimiento de actividades online puede ser utilizado con configuraciones del mundo real. Para la realización de este proceso, se utilizan las siguientes medidas según el recurso [3]:

- **Consumo de batería:**
  - Cantidad de tiempo (horas) que dura la batería mientras está corriendo un sistema de reconocimiento de actividades.
  - Cantidad de potencia consumida por el sistema de reconocimiento expresada en Watios/Hora. Esta medida es más fiable que la medida anterior (cantidad de tiempo en horas), ya que no depende de la capacidad de la batería del dispositivo donde se encuentra el reconocedor.

- **Consumo de CPU:**
  - Porcentaje de CPU que ocupa el proceso del reconocedor de actividad.

- **Consumo de memoria:**
  - Cantidad de memoria ocupada, en Megabytes (MB).

3.9 Utilidades del reconocimiento de actividad

Las utilidades finales del reconocimiento de actividad se reducen a tres aplicaciones principales [2]:

- Para usuarios finales, centrándose sobre todo en temas de salud, bienestar y estilo de vida (fitness, monitores de salud, detección de caídas…).
- Para terceros, como pueden ser grupos de investigación para la recopilación de datos.
- Para grupos o colectivos, tales como redes sociales basadas en la realización de actividades, o detección de lugares y eventos.

De estas aplicaciones, nos centraremos más en aquellas relacionadas con la salud, el bienestar y el estilo de vida.

La utilización de un sistema de reconocimiento de actividad tiene una relación bastante fuerte con el nivel de actividad física y bienestar, concretamente acompañada de la utilización en telefonía móvil. Una práctica común entraría dentro del marco de enfermedades como la obesidad, hipertensión, las cuales están muy relacionadas con la inactividad física [2]. La utilización, en estas situaciones, de un reconocimiento de actividad, permite, de manera sencilla, rápida y fiable, llevar una especie de historial que

permita saber en todo momento información relacionada con las actividades que realiza, con el objetivo de poder tratar las enfermedades del usuario [2]. La realización de este seguimiento por parte del usuario de manera manual dependería en gran medida de la voluntad del mismo. En contraposición, la utilización de dispositivos móviles facilita la tarea al usuario, obteniéndose un éxito superior. Los resultados a obtener son múltiples, desde poder mejorar el tratamiento de una enfermedad hasta poder ayudar a diferentes tipos de usuarios a establecer hábitos diarios, así como rutinas de actividades. Además, se pueden obtener evidencias de desviaciones de las anteriores rutinas de actividades, permitiendo a los médicos observar diagnósticos que durante los exámenes médicos ordinarios no podrían observar.

Además del tratamiento de enfermedades, otra posible práctica sería la de la rehabilitación de enfermedades [2]. En este ámbito los sistemas de reconocimiento encajan muy bien, ya que permiten a los médicos observar la correcta aplicación de los ejercicios que manda al usuario.

Lo comentado en los dos párrafos anteriores resulta muy útil de cara al cuidado de personas mayores. Se puede así, establecer hábitos diarios para este grupo de personas y llevar un seguimiento eficaz gracias a los sistemas de reconocimiento. Por ejemplo, en este sentido existen otras prácticas de uso, como puede ser el poder conocer el estado mental de ancianos que padecen de Alzheimer, ya que se puede establecer una relación entre el nivel de actividad física y su condición mental, ya que los enfermos afectados por este tipo de enfermedad, así como otras como puede ser la demencia, presentan inconsistencias en sus rutinas diarias [2].

Podemos establecer el tema de las caídas como una práctica aparte. No solo supone un obstáculo para los ancianos, sino también para personas con enfermedades neurodegenerativas [2]. Cuando se produce una caída al aire libre, entra en juego otros sensores, como puede ser el GPS o sensores de altitud, para poder localizar a la persona.

No todo está relacionado con personas mayores o enfermedades. Otra práctica de uso está relacionada con el bienestar y la monitorización de la actividad física [2]. La cantidad de sensores que disponen los smartphones permite el uso del dispositivo como otro tipo

aparato. Por ejemplo, puede actuar como un podómetro, posibilitando la supervisión de distancias recorridas, calorías quemadas… Cada vez es más común que los deportistas se monitoricen a la hora de realizar las actividades para así poder mejorar la técnica, consiguiendo así mejores resultados y evitando, dentro de lo posible, futuras lesiones por malas costumbres que puedan tener a la hora de la realización de las actividades físicas. En resumen, con el uso de estas se puede conseguir que el usuario tenga un estilo de vida más activo [2]. Tanto como para actividades físicas específicas como actividades cotidianas, como puede ser subir/bajar escaleras, correr… existen multitud de aplicaciones disponibles para descargar Play Store. El reconocimiento de actividad se puede utilizar con otros dispositivos conocidos como Nike Feulband, Fitbit o Xiaomi fit, por ejemplo, que están equipados de sensores como el acelerómetro, giroscopio, GPS. Son más sofisticados que los smartphones, pero más caros.

Otras prácticas más concretas podrían ser, por ejemplo, el uso del reconocimiento de actividad como recordatorio para la toma de medicamentos. También es aplicable para personas que tienen bebés, existiendo aplicaciones que consiguen predecir cuándo tienen hambre y quieren comer. Incluso para personas que quieren monitorizar sus ciclos de sueño mientras duermen. Existen otras técnicas utilizadas en niños con trastornos, como ASD (Autism Spectrum Disorder).

3.10 Desafíos

3.10.1. Sensibilidad al sujeto

Como se ha comentado anteriormente, cada persona tiene una forma de moverse, no todos andamos igual, hay gente que mueve más los brazos, o da los pasos más largos o más cortos y lo mismo pasa con todas las actividades, todos tenemos una forma personal de realizarlas. Esta personalización del desarrollo de las actividades representa una gran dificultad a la hora de hacer un reconocedor genérico de actividades. Para subsanar este problema se proponen diferentes soluciones. En [10] proponen recoger información durante periodos más largos y de personas de diferentes edades y tipo de cuerpo.

3.10.2. Independencia de la localización del dispositivo

Como ya hemos explicado, los sensores son los encargados detectan los cambios, ya sea de movimiento o de temperatura, por ejemplo. Los sensores de movimiento cuentan con unos ejes que utilizan para medir y cuantificar estos cambios. Según la posición de los sensores los ejes estarán en una posición u otra y eso influirá a la hora de reconocer las actividades.

3.10.3. Complejidad de las actividades

Hay actividades que por sí solas ya son difíciles de reconocer, ya sea por la cantidad de movimientos que conlleva, por la similitud con otras actividades, etc. Si a eso le sumamos que estamos realizando actividades sin parar y que entre esas actividades hay un momento de transición en el que seguimos mandando señales, el problema crece considerablemente. En este periodo de tiempo se envían señales, que para el reconocedor de actividades serán confusas, porque ni está realizando exactamente una actividad, ni exactamente la otra, sino una mezcla de dos actividades. Una solución que propone [10], para el reconocimiento de las transiciones entre actividades, es usar el algoritmo HMM.

3.10.4. Recursos limitados

Las aplicaciones de reconocimiento de actividades necesitan continuamente la detección de movimiento y actualizarse para realizar la clasificación. Esta detección del movimiento y la recogida de datos continúa requiere de un gasto elevado de los recursos, que dependiendo del dispositivo que se utilice para el reconocimiento será más o menos asumible.

3.10.5. Batería

Como ya veníamos comentando, la batería es un aspecto muy importante a tener en cuenta. Cuanto más optimizada esté la recogida de datos, para su clasificación, mayor duración tendrá la batería de nuestro dispositivo. En [3] ofrecen varias soluciones para que la vida de nuestra batería sea mayor. Una solución es la elección dinámica de los sensores, que no estén continuamente encendidos, sino que se enciendan o se apaguen según sean necesarios.

Otra causa que podrá afectar a la duración de nuestra batería es la frecuencia de la recogida de datos. A mayor tasa de muestreo conseguimos una mayor precisión en los resultados, pero también obtenemos un incremento en el consumo de la batería. El artículo [3] propone como solución tener una tasa de muestreo adaptable.

3.10.6. Utilización de sistemas de reconocimiento por parte de personas mayores

En el apartado anterior se comentaba una de las posibles prácticas de utilización de reconocimiento de actividad. Una de ellas sería para poder establecer hábitos diarios a ancianos, conocer su estado mental a partir de sus rutinas diarias, detección de caídas… El desafío que se nos presenta es que las personas mayores pueden presentar dificultades a la hora de utilizar, por ejemplo, smartphones, ya que, por lo general, esta generación de personas presenta dificultades a la hora de utilizar estas tecnologías [2].

4 Implementación del sistema de reconocimiento de actividad

En este capítulo explicaremos los dispositivos utilizados para implementar nuestro sistema de reconocimiento de actividad, detallaremos cada parte del proceso y además el método seguido para la obtención de datos y su posterior procesado.

Por último, analizaremos las técnicas empleadas para clasificar los datos obtenidos. En el capítulo seis, realizaremos un experimento con todo lo que explicamos en este capítulo.

4.1 Captura de datos

Para comenzar a implementar nuestro sistema de reconocimiento de actividades lo primero que necesitamos es recoger datos de distintos usuarios realizando las distintas actividades que queremos que nuestro sistema reconozca, para posteriormente, entrenar con estos datos al clasificador elegido.

Para ello, tomamos datos de dos sensores distintos: el acelerómetro y el giroscopio. Con el primero obtenemos los datos de aceleración de los tres ejes, X, Y y Z, que se le aplica al dispositivo (generalmente proporciona las unidades del SI en m/s^2igual que la gravedad) y con el segundo obtenemos los datos de giro o de rotación que se produce en el dispositivo (las principales medidas proporcionadas por sensores de este tipo son en rad/s o bien en $grados/\mathrm{s}$).

Junto a los datos del acelerómetro y del giroscopio, adjuntamos el timestamp del momento en el que los datos han sido tomados, para poder realizar su posterior procesado en ventanas temporales, como se explicará en el siguiente punto.

Comenzamos obteniendo los datos a través de una aplicación web que desarrollamos, la cual recibía los datos de los sensores y los graficaba en la pantalla para obtener una primera idea de cómo era el funcionamiento y disposición de los ejes de los sensores (anteriormente explicado en el capítulo tres).

Figura 4.1: Aplicación web (vista móvil)

En la pestaña principal hay un botón que te lleva a otra ventana en la que puedes indicar tu nombre, la actividad que vas a realizar y el tiempo de duración (30 o 60 segundos). Tras dar al botón “play” te indicará el progreso del tiempo y cuando termine, se generará un fichero en el ordenador con los datos de todos los ejes del acelerómetro y del giroscopio.

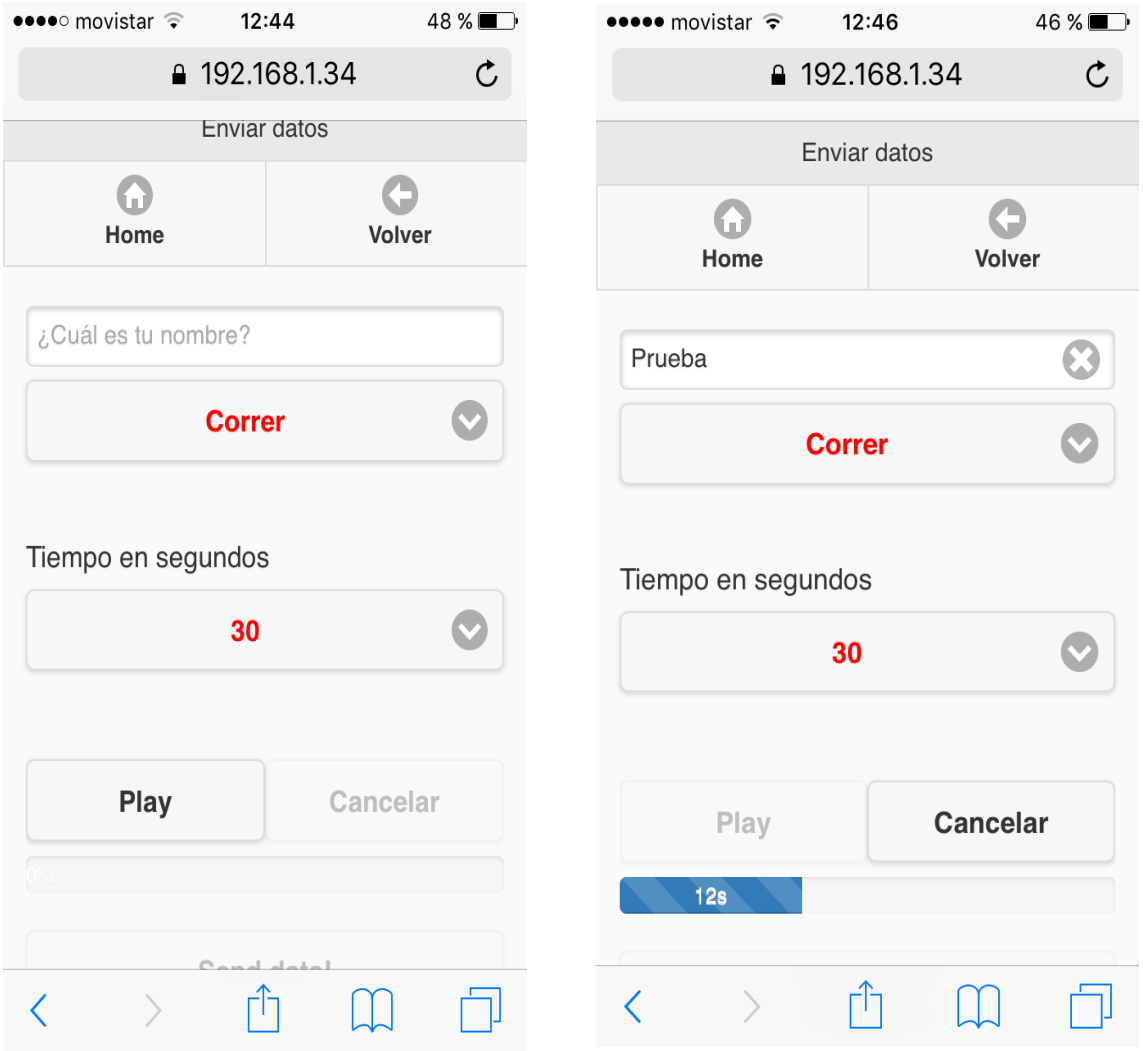


Figura 4.2: Generación de fichero y recogida datos

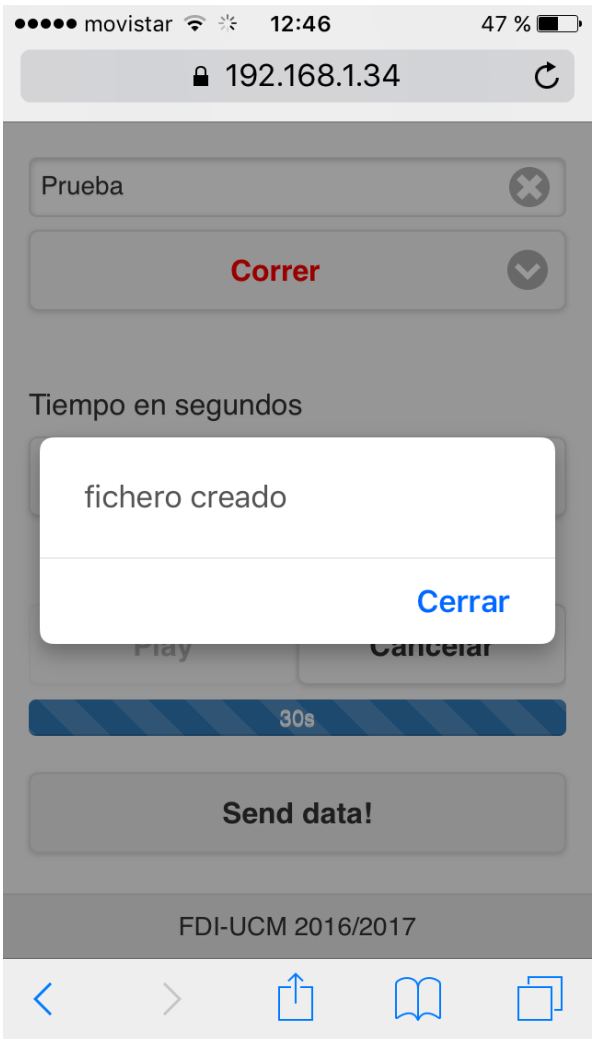


Figura 4.3: Finalización fichero

Tras esta primera toma de contacto con los sensores, adquirimos una pulsera de la marca Huawei, sobre la cual íbamos a realizar nuestro trabajo en un principio, pero tuvimos diferentes problemas debido a que la implementación de esta pulsera no permitía acceder a los datos en bruto ("RAW") de los sensores. En cambio, sí que te proporcionaba una interfaz para acceder a los datos ya procesados (esto es, los pasos que llevas, las calorías que has quemado…). Así pues, por este motivo, decidimos adquirir otra pulsera, ya que los datos procesados no nos interesan para la realización de nuestro proyecto. En esta ocasión, la pulsera era de la marca Texas Instruments, la cual sí que nos permite acceder a los datos en bruto de los distintos sensores.

Una vez analizada la nueva adquisición, obtuvimos datos de la pulsera (CC2650 Texas Instruments) a través de un script en Python que tuvimos que modificar para adecuarlo a nuestras necesidades, como por ejemplo la creación de ficheros de treinta segundos de recogida de datos, adjuntando el nombre del usuario y de la actividad que se estuviera realizando, para, proceder al posterior procesamiento. Este script permite la conexión con el dispositivo a través de la tecnología BLE ("Bluetooth Low Energy"), indicando la dirección MAC de la pulsera.

Figura 4.4: SensorTag

En el proceso de recogida de datos con la pulsera, realizamos distintas actividades (por ejemplo, andar, barrer, subir escaleras…) con distintos individuos durante aproximadamente un minuto y medio (tres ficheros de treinta segundos) por cada persona, colocando la pulsera siempre en la muñeca izquierda y siempre en la misma posición, con el objetivo de evitar errores de clasificación, ya que siempre debe ir en la misma posición o se complicaría el entrenamiento. Esto se debe a que los datos son significativamente distintos al recogerlos en la muñeca izquierda o en la derecha, o que llevarlo en otra parte del cuerpo, por lo que acordamos que para que el sistema funcione correctamente, el dispositivo (ya sea pulsera o teléfono) ha de estar colocado siempre en la misma posición. Esto último hace referencia al problema de la independencia de la orientación, que como explican en el artículo [3], en la mayoría de estudios analizados no hay independencia de la orientación, esto quiere decir que para poder realizar el reconocimiento de actividades, el dispositivo de reconocimiento ha de estar siempre en el mismo lugar y posición (es decir, hay dependencia de la orientación) para que no haya confusión a la hora de reconocer la actividad. Así mismo [2] también proporciona algunas soluciones que se han llevado a cabo para solucionar el problema de la dependencia de la orientación, los cuales

los explicamos en el capítulo 3. En nuestro caso, como hemos comentado anteriormente, nuestro sistema es dependiente de la orientación, ya que pensamos que consume menos recursos y no está entre uno de nuestros objetivos dotar independencia de orientación al sistema y por ello decidimos que la pulsera se debía poner mirando hacia arriba, sobre la muñeca de la mano izquierda.

Por último, recogimos datos del acelerómetro y del giroscopio del móvil, lo cual nos supuso nuevos retos. En los móviles, generalmente, el sensor del acelerómetro y del giroscopio se encuentran separados, por lo que al obtener los datos de ambos sensores puede obtenerse diferencias notables en los tiempos, lo cual perjudica gravemente el proceso de reconocimiento de actividades, ya que, por ejemplo, estaríamos obteniendo los datos del acelerómetro en el instante t y otro dato del giroscopio en el instante t + 10, por lo que no estarían coordinados o sincronizados los datos entre sí.

Se realizó un proceso de comprobación de estos tiempos para ver si era viables juntar los datos de ambos sensores (cada uno con su timestamp) en un único timestamp. Para ello se hicieron tres datasets distintos, uno con el timestamp y los datos del acelerómetro, otro con el timestamp y los datos del giroscopio y otro final con el timestamp tras haber obtenido ambos datos y los datos de ambos sensores. Tras haber hecho estos tres datasets, se desarrolló un script en Python para comprobar que la diferencia de los tres timestamps anteriores no fuese muy distintos unos de otros.

Tras ejecutar el script con los tres datasets anteriores, se analizaron las gráficas obtenidas y se concluyó finalmente que era viable juntar los datos del acelerómetro y del giroscopio en un único dataset con un único timestamp, ya que las diferencias en los tiempos eran mínimos. En los siguientes gráficos podemos observar como los timestamps del acelerómetro y del giroscopio no encajan a la perfección, si no que tienen un desfase, por lo que realizamos distintas pruebas haciendo gráficas, y finalmente coinciden sin mucha diferencia, por lo que ha sido posible juntar ambos datos de acelerómetro y giroscopio en una única lista con un único timestamp:

	timestamp	x	y	z	Fecha
0	1492622020576	0.156128	6.443527	7.479523	19-04-2017 19:13:40.576000
1	1492622020670	0.156128	6.443527	7.479523	19-04-2017 19:13:40.670000
2	1492622020770	0.156128	6.443527	7.479523	19-04-2017 19:13:40.770000
3	1492622020869	0.156128	6.443527	7.479523	19-04-2017 19:13:40.869000
4	1492622020974	0.156128	6.443527	7.479523	19-04-2017 19:13:40.974000

Tabla 2: Timestamps acelerómetro

	timestamp	ga	gb	gc	Fecha
0	1492622020574	0.005325	-0.003189	-0.011719	19-04-2017 19:13:40.574000
1	1492622020668	0.005325	-0.003189	-0.011719	19-04-2017 19:13:40.668000
2	1492622020768	0.005325	-0.003189	-0.011719	19-04-2017 19:13:40.768000
3	1492622020867	0.005325	-0.003189	-0.011719	19-04-2017 19:13:40.867000
4	1492622020973	0.005325	-0.003189	-0.011719	19-04-2017 19:13:40.973000

Tabla 3: Timestamps giroscopio

	timestamp	x	y	z	ga	gb	gc	Fecha
0	1492622020587	0.405121	6.941528	6.481140	-0.301468	0.518784	0.111847	19-04-2017 19:13:40.587000
1	1492622020682	0.187256	6.527329	8.283966	-0.365387	-0.278030	-0.231155	19-04-2017 19:13:40.682000
2	1492622020783	0.297379	6.819412	6.598450	0.457001	-0.174698	-0.073502	19-04-2017 19:13:40.783000
3	1492622020883	0.342865	6.912796	6.684647	0.009583	0.204529	0.061783	19-04-2017 19:13:40.883000
4	1492622020983	-0.241318	7.662170	6.778015	0.490021	0.140610	-0.006393	19-04-2017 19:13:40.983000

Tabla 4: Timestamps acelerómetro y giroscopio

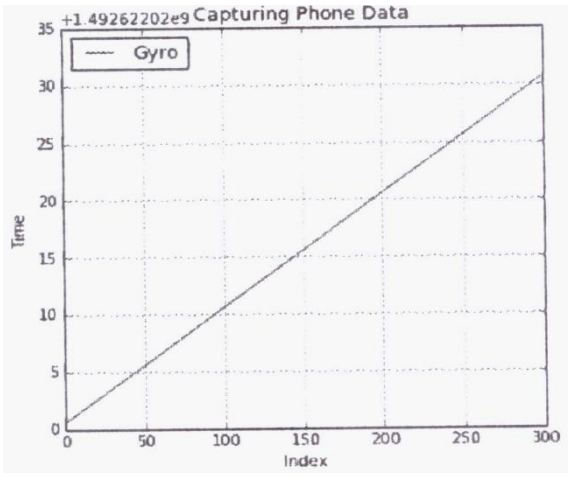


Figura 4.5: Timestamp giroscopio

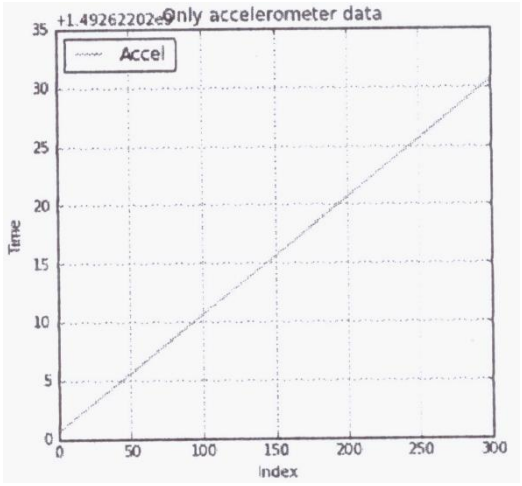


Figura 4.6: Timestamp acelerómetro

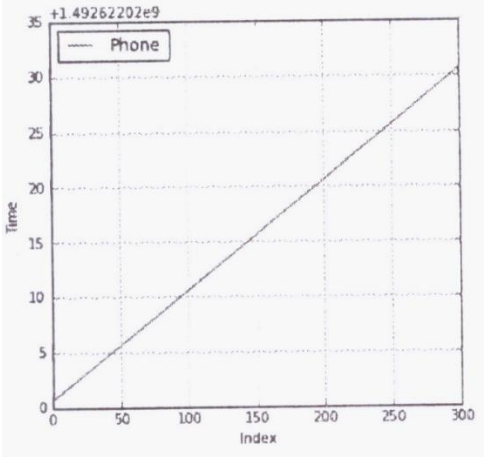


Figura 4.7: Timestamp acelerómetro y giroscopio

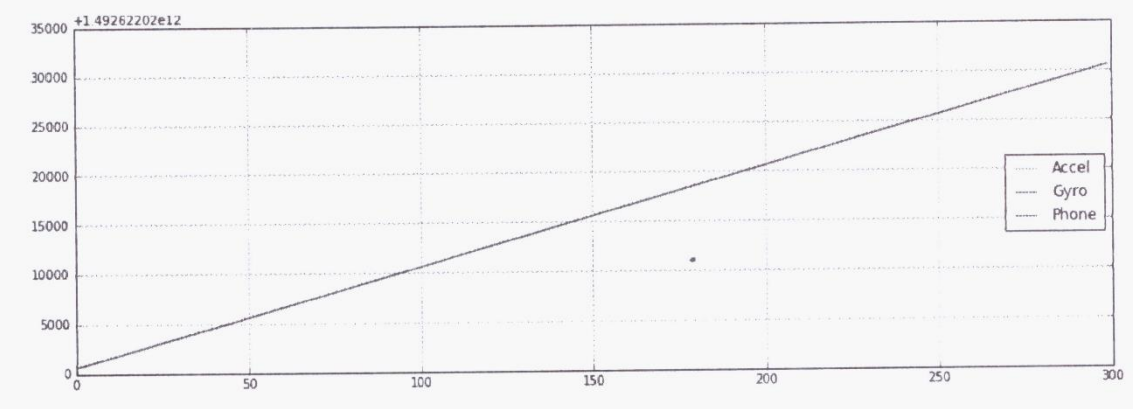


Figura 4.8: Timestamp acelerómetro, giroscopio y timestamp unificado

En el proceso de recogida de datos con la aplicación móvil, con el acelerómetro y el giroscopio del propio dispositivo, se ha desarrollado una parte bastante intuitiva de cara al usuario, en el sentido que es mucho más gráfico y sencillo (a diferencia del proceso de recogida de datos de la pulsera con el script de Python). El usuario, primero tiene que colocarse el móvil en la muñeca izquierda (al igual que la pulsera) y después simplemente tiene que introducir su nombre y la actividad que va a realizar. Siguiendo unos sencillos pasos, comienza a grabarse durante treinta segundos realizando la actividad (será explicado más en detalle en el capítulo 5 y en la guía de instalación y de uso).

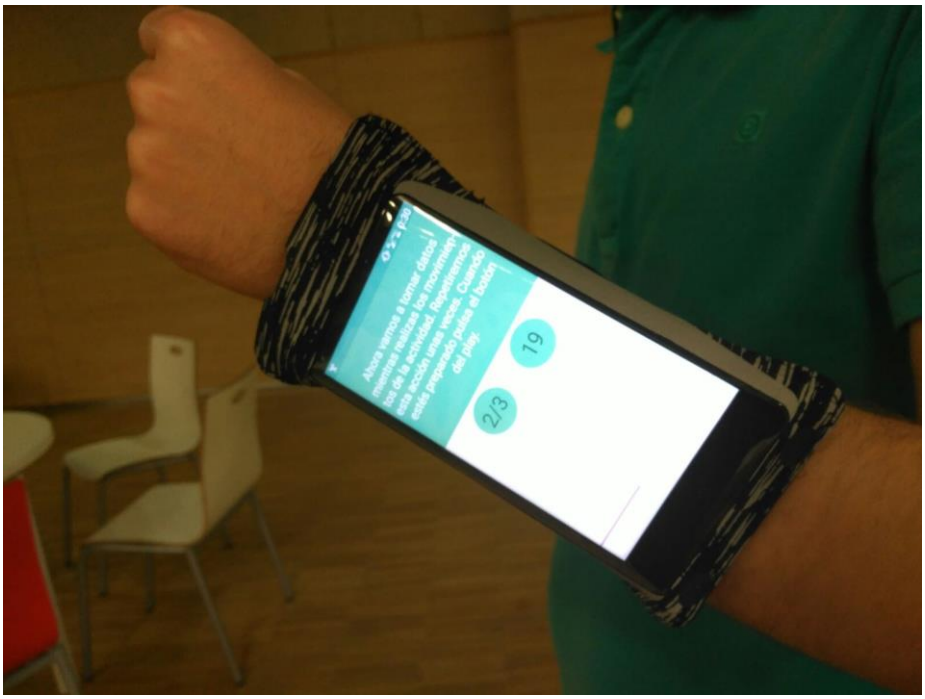


Figura 4.9: Asistente de recogida de datos en funcionamiento.

4.2 Procesamiento de datos

En este apartado describiremos los métodos que hemos utilizado para realizar la extracción de las características (esto es, las funciones matemáticas que hemos utilizado como la media, el mínimo, el máximo, la desviación estándar, la transformada rápida de Fourier…) de los datos anteriormente recogidos.

Para comenzar con la extracción de las características, lo primero que tenemos que hacer es una segmentación de los datos recogidos, esto ayuda a reducir la complejidad y es una buena práctica cuando los datos se recuperan continuamnete. Nosotros decidimos segmentar la señal en ventanas de un segundo, como se explica en el estudio[10], en que condujeron un experimento donde probaron diferentes tamaños de ventana y el resultado fue que a medida que aumentaban el tamaño de ventana, el porcentaje de acierto disminuia. Además, también decidimos que el porcentaje de solapamiento (overlap) entre ventana y ventana fuera de un 50% (en este caso, medio segundo de solapamiento entre ambas ventanas), elegimos ese valor ya que en varios estudios [3] lo utilizan. El valor del tamaño de ventana y de solapamiento lo decimos tras analizar los anteriores estudios y tras ver cual se ajustaba mejor a nuestro reconocedor comprbando la tasa de acierto. Entonces, de cada ventana hemos extraído distintas características, las cuales serán usadas posteriormente para el proceso de entrenamiento offline (es decir, fuera del dispositivo, en otra máquina) del clasificador y posterior reconocimiento de las distintas actividades en tiempo real que el usuario esté realizando.

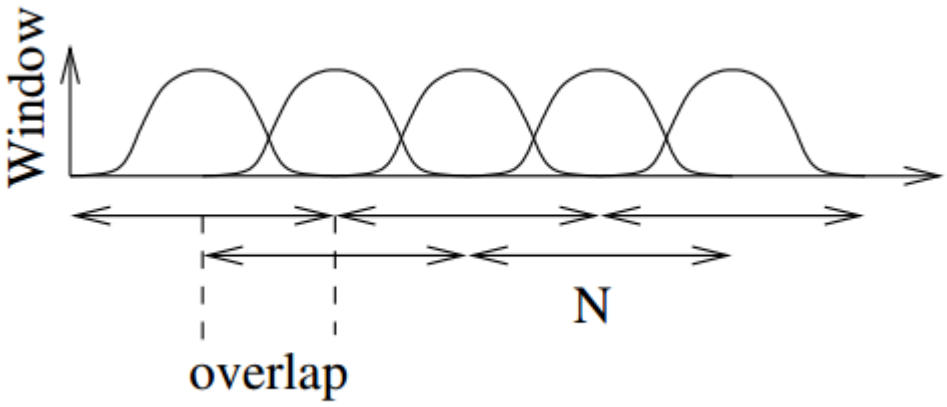


Figura 4.10: Segmentación de la señal. Fuente: stack.imgur.com

En la imagen anterior se puede observar gráficamente la segmentación de los datos en ventanas de un segundo y con un solapamiento del 50% (como hemos comentado anteriormente, de medio segundo). En la imagen anterior, N se corresponde al tamaño de la ventana ("Window Size"), que en nuestro caso sería de un segundo, y overlap se corresponde al solapamiento que hay entre ventana y ventana. En nuestro caso, como hemos comentado y anteriormente sería de medio segundo.

Para la elección de algunas de las características nos hemos basado en distintos artículos como en [3], en el cual aportan una tabla descriptiva en la que se describe las actividades que se reconocían en los distintos estudios analizados y las características que estaban utilizando para reconocer las distintas actividades.

Entre las distintas características que hemos extraído de cada, se encuentran las siguientes:

- Mínimo.
- Máximo.
- Media.
- Mediana.
- Desviación estándar.
- Correlación de los tres ejes del acelerómetro, tomados dos a dos.
- Transformada rápida de Fourier (FFT) de los tres ejes del acelerómetro. Utilizando la energía de Fourier.

Las métricas anteriormente citadas son la base de nuestro sistema, pero también hemos utilizado otras como Kurtosis, Skew, suma, desviación estándar y media de los valores absolutos de fourier, la RMS de Fourier (raíz cuadrada de media de cuadrados) o la RFFT (cálculo de la Transformada de Fourier discreta (DFT)). Todo ello gracias a bibliotecas como Scipy, Numpy o Pandas. Las métricas no se escogen aleatoriamente, depende del tipo de actividad que se esté realizando. Para actividades de **movimientos más bruscos**, como por ejemplo, barrer, boxeo... se necesitan el máximo o el mínimo para detectar esos movimientos. Para diferenciar actividades más similares como pueden ser andar o subir escaleras, una buena métrica es utilizar Fourier. La elección de métricas depende de lo que queramos observar, por ejemplo, si queremos que nuestros datos sean

sensibles a valores extremos utilizaremos la media. En cambio, si queremos que no se vean afectados elegimos la mediana. En total, hay tres tipos de métricas, las que son de tendencia central (media, mediana, etc...), dispersión (desviación, varianza, etc...) y de picos (máximo y mínimo). Algunas métricas son más sensibles a valores extremos, por tanto dependiendo de la actividad y los valores que queramos obtener, utilizaremos unas u otras.

4.3 Técnicas de clasificación empleadas

En este apartado explicaremos las técnicas que hemos utilizado para la clasificación de nuestros datos después de haber realizado su procesamiento, tal y como hemos explicado en el apartado anterior.

El fundamento de los árboles de decisión consiste en que, a partir de las métricas o características calculadas anteriormente, vaya tomando decisiones, acotando actividades hasta que se tome una decisión final, decidiendo la clase a la que pertenece. El objetivo es crear un modelo que predice el valor de una variable destino en función de diversas variables de entrada. Un ejemplo[11] se muestra en el diagrama de abajo, cada nodo interior corresponde a una de las variables de entrada. Cada hoja representa un valor de la variable destino dados los valores de la variable de entrada representados por la ruta desde la raíz a la hoja.

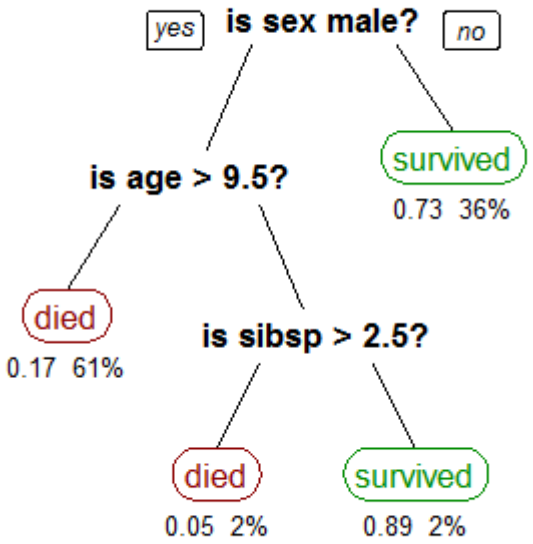


Figura 4.11: Ejemplo de árbol de decisión

Nuestra técnica principal de clasificación son los árboles de decisión, debido a que en las pruebas realizadas en [3] lo utilizaban y los resultados obtenidos eran muy buenos. Esta técnica es sencilla y permite parsear o transformar el árbol de decisión fácilmente a un sistema Android o cualquier otro lenguaje de programación (ya que los árboles de decisión, una vez entrenados, se convierten fácilmente en unos simples "if-else").

Según [12], entre algunos de los beneficios de los árboles de decisión se encuentran que no son excesivamente sensibles al número de variables de entrada que se usan, por tanto no empeora si se añaden variables que no aportan información. Además, al usar un árbol de decisión te puedes despreocupar de investigar previamente que variables son relevantes para clasificar cada actividad, ya que el propio entrenamiento del árbol, lo hará por nosotros si se controla el sobreajuste.

En cambio, algunas de las desventajas que tienen es que son propensos al sobreajuste, es decir, que ajusta mucho el sistema a los datos con los que ha sido entrenado y cuando al sistema le llega un nuevo dato que no ha sido incluido entre los datos de entrenamiento (datos para construir el árbol) del árbol, este tendrá más posibilidades de

ser clasificado erróneamente, aunque este problema lo mitigamos con una estrategia adecuada de validación cruzada, como hicimos separando los datos de validación y de entrenamiento, y haciéndolo varias veces utilizando diferentes datos para validar.

Para crear nuestro árbol de decisión utilizamos la biblioteca Scikit-learn [13] de Python.

Para finalizar este capítulo, queremos comentar, que además de los árboles de decisión, hemos utilizado otras técnicas, como son los Random Forest, los cuales proporcionan una mayor precisión, pero tienen un mayor coste y que tras haber realizado distintas pruebas, decidimos descartarlos ya que no nos daban una gran mejora respecto al aumento de coste que suponían, en comparación de los árboles de decisión, ya que estos ya nos daban unos resultados muy buenos. También fue descartada por la mayor dificultad que conlleva su implementación en Android. Esta técnica, combina una gran cantidad de árboles de decisión independientes probados sobre conjuntos de datos de datos aleatorios con igual distribución y donde cada árbol solamente usa un subcojunto de las variables. La clasificación final se obtiene como una votación de todos los árboles que integran el clasificador (de ahí el nombre de bosque). En las siguientes imágenes se puede ver la diferencia entre ambos.

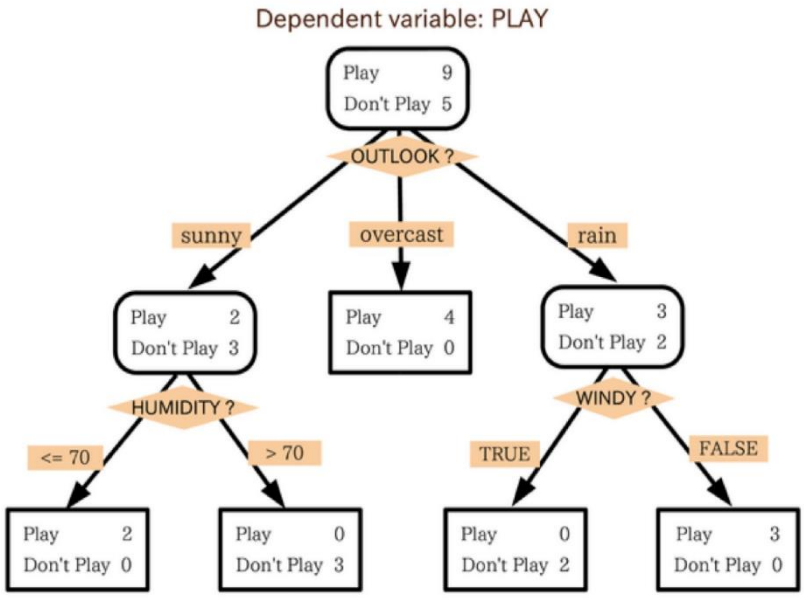


Figura 4.12: Árboles de decisión [14]

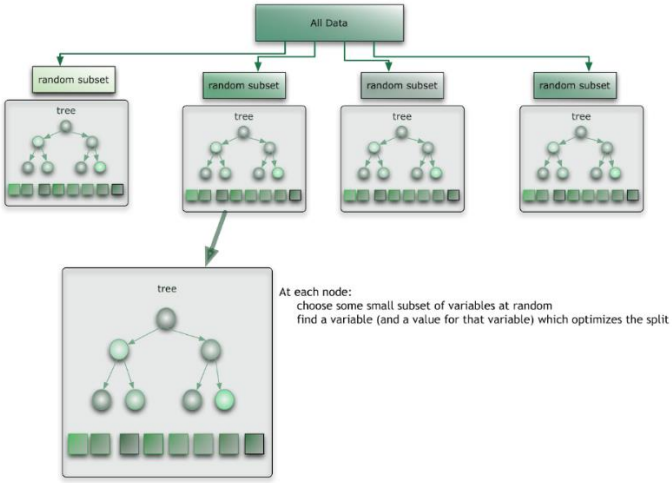


Figura 4.13: Random Forest [14]

4.4 Entrenamiento y validación

Para las primeras pruebas que realizamos, en lo referido a la validación de datos utilizamos K Fold validation. Esta técnica consiste en dividir en k subconjuntos, donde uno de ellos se utiliza como validación mientras los otros se utilizan como entrenamiento. Esta técnica sirve para comprobar el porcentaje de acierto de todo el sistema, y evita el sobreajuste, porque podemos tener un conjunto de validación, que se ajusta muy bien al sistema, pero en cambio con otros valores la precion disminuye. Con este método calculamos la media de todos los conjuntos de validación y permite saber de manera mas precisa su efectividad. Se puede ver en la siguiente figura:

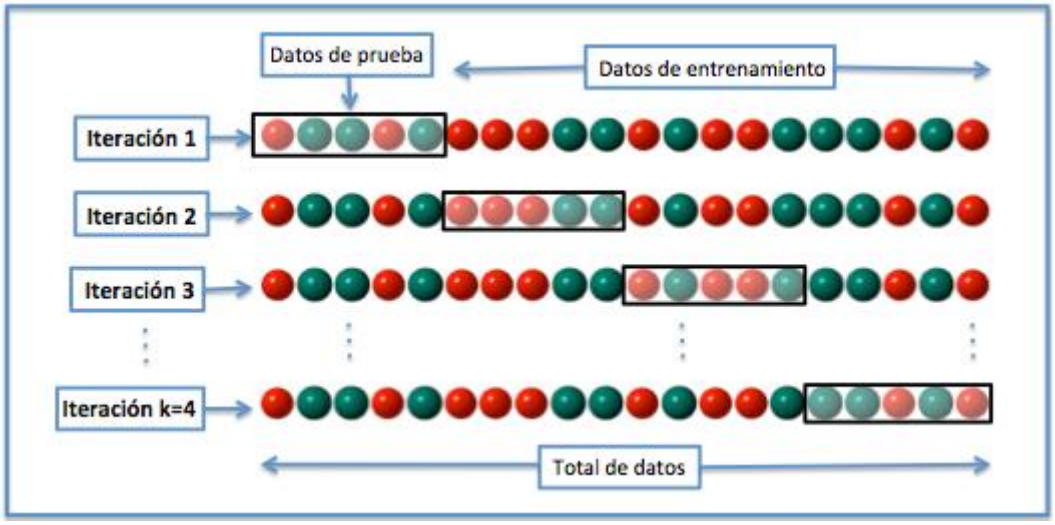
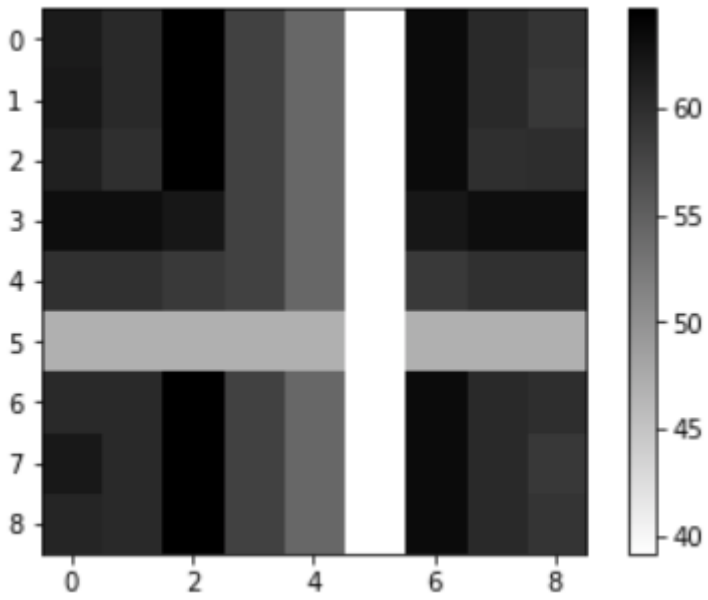


Figura 4.14: Validación cruzada. Fuente: Wikipedia

Para garantizar una cross-validation realista, que pueda informar de forma fidedigna del funcionamiento del clasificador en un entorno real hemos optado por tomar datos de N usuarios y usar N-1 usuarios para entrenar y 1 para validar. De esta forma podemos estimar cual es la tasa de acierto que tendrá el clasificador ante un usuario nuevo.

Posteriormente, utilizamos dos características esenciales para evitar sobreajuste y permitir obtener una mayor tasa de acierto que generalice bien. Estas dos métricas son el mínimo nodos para hacer la **división** entre dos actividades (min_samples_split) y el mínimo número de nodos para **convertirse en un nodo hoja** (min_samples_leaf). Esto ayuda mucho, ya que, si hay muy pocos ejemplos de entrenamiento, tras realizar las divisiones llega un momento en el que no hace más, evitando el sobreajuste y cálculos innecesarios.

Hicimos una combinación con las dos características, realizamos un mapa de calor (heat map), el cual permite ver de forma gráfica, marcando de diferente color, cuál de las combinaciones tiene una mayor exactitud. A mayor nivel de oscuridad, mayor es el porcentaje de exactitud. El eje X pertenece a la característica de min_samples_split y el eje Y a min_samples_leaf.



0 - 0.01
1 - 0.25
2 - 0.1
3 - 0.2
4 - 0.5
5 - 50
6 - 25
7 - 10
8 - 2

Figura 4.15: Mapa de calor (Heat map).

Para crear el mapa de calor primero se creaba el árbol con los datos de entrenamiento y para calcular la exactitud se probaba con los datos de validación. Los valores utilizados para las características son los que se muestran en la tabla de la derecha del mapa de calor. De la posición 0 a la 4, ambos inclusive, son el porcentaje sobre el numero de muestras que debe haber como mínimo, y de la posición 5 a la 8 indican el número mínimo de muestras que debe haber

4.5 Medidas y matrices del sistema

Construimos matrices de confusión. Este punto del proceso constituye una parte bastante importante, ya que podemos descubrir si nuestro sistema funciona correctamente. Una matriz de confusión está formado por un conjunto de filas y columnas, donde cada columna representa el número de predicciones de cada clase, mientras que cada fila representa el número de ejemplos reales.

		Clase real	
		Clase referencia	***Clase no referencia***
Clase estimada	Clase referencia	TP	FP
	Clase no referencia	FN	TN

Figura 4.16: Matriz de confusión

Para asegurarnos que nuestro sistema funcione correctamente, calculamos una serie de medidas, como son, precision, sensibilidad (recall) y exactitud (accuracy), las cuales las explicamos a continuación más detalladamente:

- **Precisión:** es el número de resultados correctamente devueltos dividido por el número total de resultados devueltos.

$$Precision = \frac{tp}{tp + fp}$$

- **Sensibilidad:** es el número de resultados correctos devueltos dividido por el número de resultados que deberían haber sido devueltos.

$$Sensiblidad = \frac{tp}{tp + fn}$$

- **Exactitud:** proporción de resultados correctos que obtiene un clasificador.

$$Exactitud = \frac{tp + tn}{tp + tn + fp + fn}$$

tp = true positive (verdaderos positivos)

fp = false positive (falsos positivos)

tn = true negative (verdaderos negativos)

fn = false negative (falsos negativos)

5 Aplicación Android

En este quinto capítulo explicaremos en detalle la aplicación para Android que hemos desarrollado para darle una forma más accesible, sencilla y fluida al trabajo desarrollado e implementado en la parte de Machine Learning.

Explicaremos desde cómo empezó el diseño, cómo se desarrolló e implementó la aplicación, problemas a los que nos tuvimos que enfrentar y la solución que se les dio y por último, mostraremos el funcionamiento de la aplicación y el resultado final obtenido.

Todo el material utilizado e implementado para la aplicación (código fuente, imágenes…) puede encontrarse actualizado en el siguiente enlace al repositorio de GitHub utilizado para llevar un control de versiones de la aplicación y permitir un desarrollo colaborativo y en paralelo de la aplicación (actualmente se encuentra bajo licencia GPL):

https://github.com/TfgReconocimientoPulseras/TrainAppTFG

5.1 Proceso de diseño

Para el desarrollo de la interfaz de la aplicación (lo que viene a ser la parte gráfica), hemos seguido una metodología que implementa el proceso de ingeniería de la usabilidad. Esta metodología se denomina Diseño Guiado por Objetivos (DGO o *Goal-Directed Design*, metodología propuesta por Alan Cooper [15]), y resumidamente, esta metodología se centra principalmente en diseñar a partir de conocer los distintos objetivos que el usuario de la aplicación busca a la hora de utilizar el sistema.

Utilizamos esta metodología ya que teníamos que realizar un proyecto para desarrollar una interfaz para la asignatura de Desarrollo De Sistemas Interactivos (DSI), siguiendo los principios de la metodología anteriormente descrita, por lo que decidimos que este proyecto podría ser útil para elaborar parte de la interfaz de nuestra aplicación de este TFG.

Aunque esta primera interfaz desarrollada para la asignatura de DSI se desvíe un poco de los objetivos finalmente implementados para nuestro trabajo, sí que nos ha sido de utilidad para basarnos en el diseño elaborado finalmente para la aplicación.

Todo el material correspondiente generado en esta primera fase de diseño puede encontrarse en el siguiente enlace (sólo visible para usuarios con cuenta ucm):

https://drive.google.com/open?id=0B1_UTuHEcUIscHhWejlhdnM0aXc

Basándonos en el Diseño Guiado por Objetivos [16], en este apartado definiremos y explicaremos brevemente algunas de las principales fases que hemos realizado, aunque para conseguir una mayor información del trabajo realizado para la primera interfaz puede consultarse el enlace anterior, aunque algunos apartados se han visto sometidos a distintos cambios posteriores para conseguir la adaptación de la primera interfaz a la interfaz realizada para este trabajo.

5.1.1 Fase de investigación

En esta primera fase elaboramos un plan de investigación, en el cual principalmente redactamos las preguntas que íbamos a generar a los distintos tipos de personas que pensamos que podrían ser propensas a usar nuestra aplicación (por ejemplo, monitores de gimnasios o colegios, deportistas o personas que tienen preocupación por su estado física).

Después de realizar estas entrevistas a distintas personas que encajaran con los tipos de personas que pensamos que podrían estar interesadas en usar nuestra aplicación, procesamos las distintas anotaciones y distintos comentarios obtenidos durante la realización de las entrevistas, habiendo obtenido los siguientes principales resultados:

- Ana, que es monitora de gimnasio, considera que llevar el móvil en el brazo es más engorroso a la hora de realizar los distintos ejercicios físicos, por eso considera que es mejor que el dispositivo de detección sea una pulsera.

- José Luis, qué es deportista, considera que el dispositivo de medición tiene que ser cómodo y no tiene que interferir de ninguna manera con el deporte o actividad que se esté realizando.
- Marlon, qué es deportista, utiliza distintas aplicaciones para móvil para registrar los resultados de sus actividades. Le gusta en especial que le propongan retos o juegos para superar sus marcas anteriores.
- Marlon busca dispositivos fáciles de llevar y que sean fáciles y rápidos de usar para no tener que parar a configurar la aplicación en medio de la actividad.
- Lorena, que tiene una preocupación por su condición física, indica que en las aplicaciones de la competencia, muchas de las funcionalidades son de pago. Este aspecto puede evitar que muchos deportistas casuales no utilicen la aplicación.
- Lorena comparte su intención por utilizar dispositivos para facilitar el entrenamiento, pero la forma del dispositivo puede ser causa de distracción al a hora de realizar las actividades.

Como podemos comprobar, en la mayoría de los comentarios obtenidos se hace una alusión a que el dispositivo utilizado para la realización del reconocimiento de actividades tiene que ser algo que evite la realización incorrecta de los ejercicios o que perturbe la realización de estos, siendo preferible usar una pulsera a un smartphone.

En esta fase también realizamos un análisis de la competencia para ver las características que estaban disponibles en las aplicaciones ya implementadas en el mercado que fueran similares a la aplicación que queremos desarrollar:

	Runtastic	Train&Run	Moves	Zombies	Podómetro	PolarFlow
Gráficas	No	No	No	No	Si	Si
Historial	Si	Si	Si	No	Si	Si
Configurable	Si	Si	No	No	Si	Si
Mapa	Si	Si	No	Si	No	No
Perfil	Si	Si	No	No	Si	Si
Andar	No	Si	Si	No	Si	Si
Correr	Si	Si	Si	Si	Si	Si
Bicicleta	No	Si	Si	No	No	No
Entrenamiento	No	Si	Si	No	No	No
Descanso	No	No	No	No	Si	Si
Entrenador	No	No	No	No	No	No
Entrenamientos	Si*	Si	No	Si	Si*	Si
Objetivos	Si*	Si	No	Si	Si*	Si
Calendario	Si*	Si	Si	No	Si	Si
Retos	No	No	No	Si	Si*	No
Clasificación	No	No	No	Si	Si	No
Jugabilidad	No	Si	No	Si	No	No
Dispositivos	Si	Si	No	No	No	Si

Tabla 5: Tabla de análisis de la competencia

5.1.2 Requisitos

Los requisitos de nuestra aplicación vienen dados en parte por la propia definición de este TFG (Reconocimiento de actividades mediante pulsera con sensores) y otros requisitos han ido surgiendo según íbamos progresando en la realización de nuestro trabajo, ya que el tema estaba abierto a desarrollar otras funcionalidades adicionales.

A continuación, listamos los requisitos principales que definimos para el desarrollo de la aplicación:

- La aplicación debe soportar el reconocimiento de actividades tanto con los sensores del propio smartphone, como con un dispositivo que sea más ergonómico como una pulsera o similares, aunque en nuestro caso solamente nos conectaremos a la pulsera SensorTag CC2650.
- En relación con el requisito anterior, también tendrá que proporcionar una manera sencilla e intuitiva de conectarse, en caso de así desearlo, al dispositivo Bluetooth que se desee usar en vez del propio smartphone.
- Además, la aplicación también tiene que proporcionar un proceso sencillo que permita al usuario añadir al sistema de una manera fácil sus propias actividades.
- También tiene que disponer de un historial que permita al usuario listar las distintas actividades que ha estado realizando y durante cuánto tiempo.
- Debe permitir al usuario realizar de manera sencilla la gestión (eliminación, modificación…) de los distintos objetos del sistema, para dar una sensación de libertad a la hora de usar la aplicación.
- La aplicación tiene que ser sencilla y fluida a la hora de utilizarla.

Ahora definiremos una serie de requisitos que consideramos adicionales y que no son de obligado cumplimiento para cumplir con las metas principales de este trabajo:

- La aplicación debe ser energéticamente eficiente, esto es, que el consumo de batería por parte de la aplicación ha de ser lo más bajo posible, ya que normalmente, este consumo se puede regular para que disminuya el consumo de batería, por ejemplo, como explicaban en el artículo [3] este consumo puede disminuirse ajustando por ejemplo la frecuencia de muestreo de los sensores, o no olvidando de apagarlos cuando ya no son necesarios utilizarlos.
- Debido a la diversidad que hay en el mundo de los smartphones (distintos tamaños de pantalla, versiones de sistemas operativos, disponibilidad de los sensores…), la aplicación debe poder utilizarse en una amplia mayoría de los dispositivos actuales y, en caso de que no sea posible utilizarla, la aplicación informará al usuario de los distintos motivos. En el caso por ejemplo de que el móvil no tenga giroscopio.
- La aplicación debe implementar un sistema de objetivos que permita al usuario marcarse un objetivo y que la aplicación lleve un seguimiento del nivel de progreso de dicho objetivo (por ejemplo, correr 30 km en una semana).
- Por último, la aplicación debe implementar una parte social, que permita intercambiar información, realizar competiciones entre los distintos usuarios y que además permita crear grupos de usuarios para cumplir con un objetivo de "grupo" (por ejemplo, correr entre todos los usuarios del grupo 100 km en una semana).

5.1.3 Escenario

En la introducción ya definimos anteriormente un escenario aplicado al campo de la salud, pero como ya comentamos nuestra aplicación se enfoca más específicamente al ámbito del bienestar o del deporte, por lo que a continuación definiremos un posible escenario de uso de nuestra aplicación enfocado al campo del bienestar a partir de un posible usuario imaginario:

Víctor, que es un deportista al que le encanta realizar diversos ejercicios, se encuentra frustrado ya que con la aplicación actual que utiliza para llevar un control de los distintos ejercicios que realiza en el gimnasio, resulta que no reconoce adecuadamente cuando realiza boxeo, no registrando que ha estado boxeando durante aproximadamente dos horas, hecho que a Víctor no le gusta para nada. Víctor le comenta este problema a su amigo Juan, el cual le recomienda que se instale nuestra aplicación, ya que podrá grabar su propia actividad "Boxear" y el sistema le reconocerá con un porcentaje bastante alto cuando esté boxeando o no. Víctor decide seguir el consejo de su amigo Juan, se instala nuestra aplicación, realiza durante un minuto y medio boxeo para que la aplicación sea capaz de reconocer posteriormente que Víctor esté realizando boxeo. Tras este "entrenamiento" de la aplicación, Víctor se propone probarla a ver si es cierto que la aplicación ha reconocido que ha estado durante una hora realizando boxeo. Tras realizar una hora de boxeo, Víctor mira la aplicación y observa, que efectivamente como su amigo Juan le dijo la aplicación ha reconocido perfectamente que ha estado durante una hora realizando boxeo, por lo que Víctor se encuentra bastante satisfecho con la aplicación, que también la usará para reconocer otras actividades que realiza en el gimnasio (pesas, flexiones…).

Con el escenario anteriormente descrito queremos ejemplificar la utilidad de nuestra aplicación en el ámbito del deporte y también en lo que difiere principalmente de otras aplicaciones existentes y que tienen objetivos similares a los de nuestra aplicación. Escenificando el reconocimiento de cualquier actividad y adaptándose a cada persona en concreto.

5.1.4 Prototipo inicial

Como comentábamos al principio de este apartado, realizamos una primera aproximación de la aplicación que queríamos realizar en este trabajo para la asignatura de DSI. En esta primera aproximación a nuestra aplicación realizamos varias iteraciones realizando primeramente prototipos en papel de la aplicación. Posteriormente realizamos cambios sobre estos prototipos en papel para adecuarnos a nuestras necesidades y, por último, digitalizamos estos prototipos en papel con la aplicación web de MyBalsamiq. Puede encontrarse información más detallada acerca del proceso seguido en el enlace que se encuentra al principio de este apartado.

A continuación, mostramos algunos de los prototipos en papel que realizamos, así como también algunas imágenes de este primer prototipo digitalizado, el cual fue entregado para la evaluación de la asignatura de DSI.

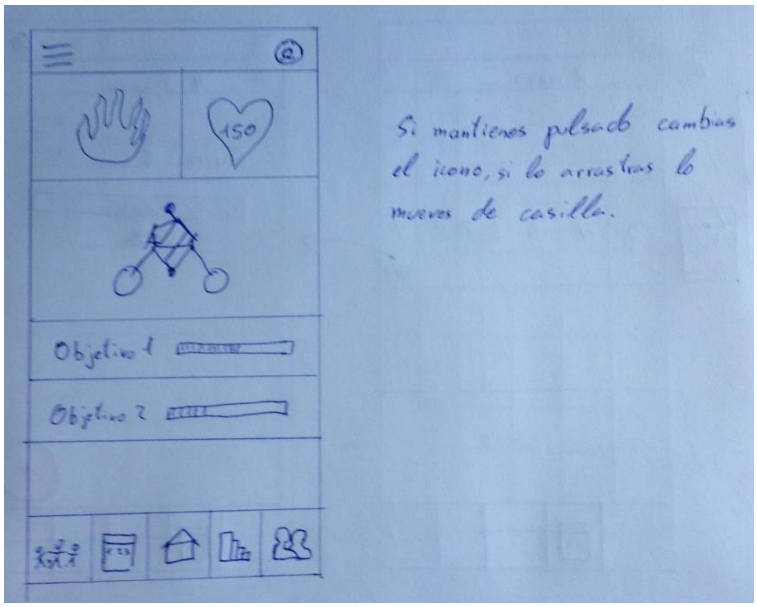


Figura 5.1: Prototipo inicial. Ventana principal

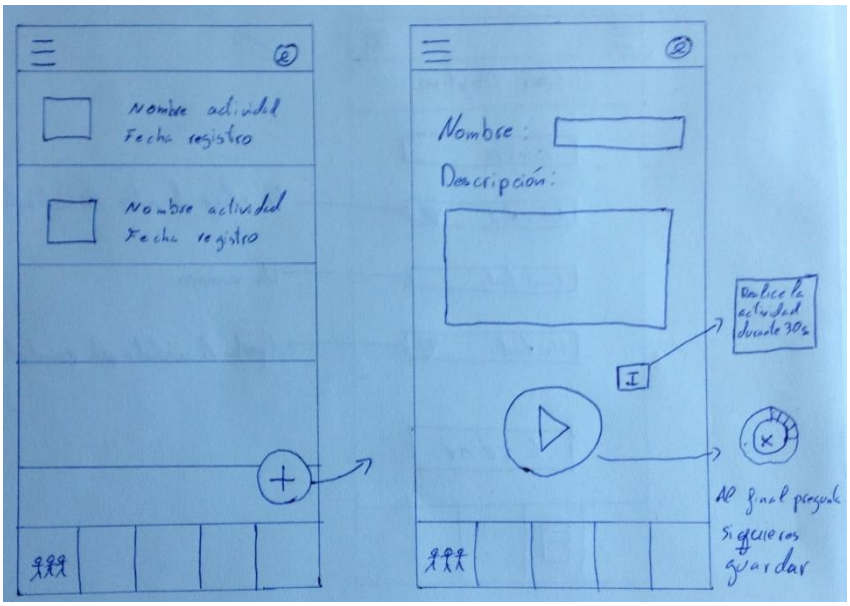


Figura 5.2: Ventana añadir nueva actividad

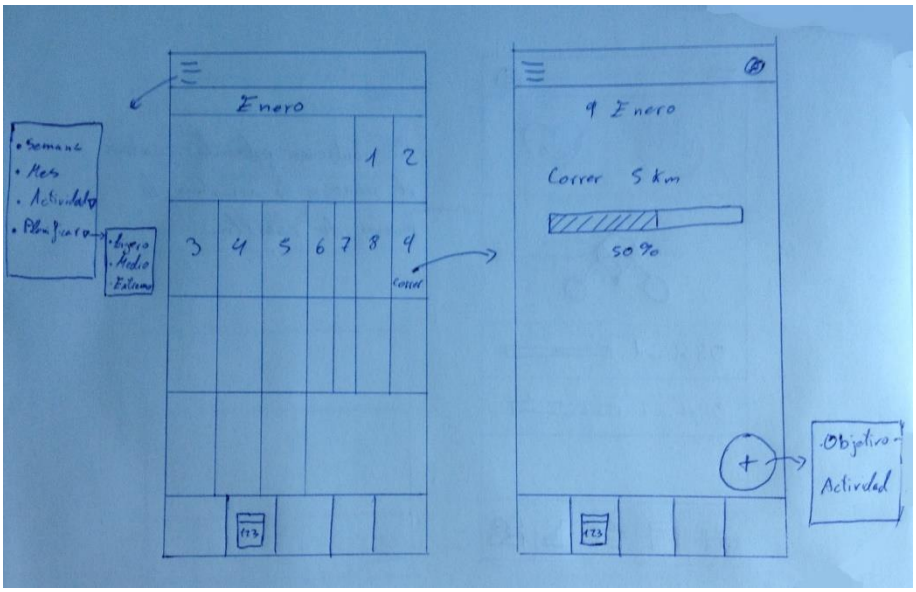


Figura 5.3: Prototipo inicial. Ventana calendario

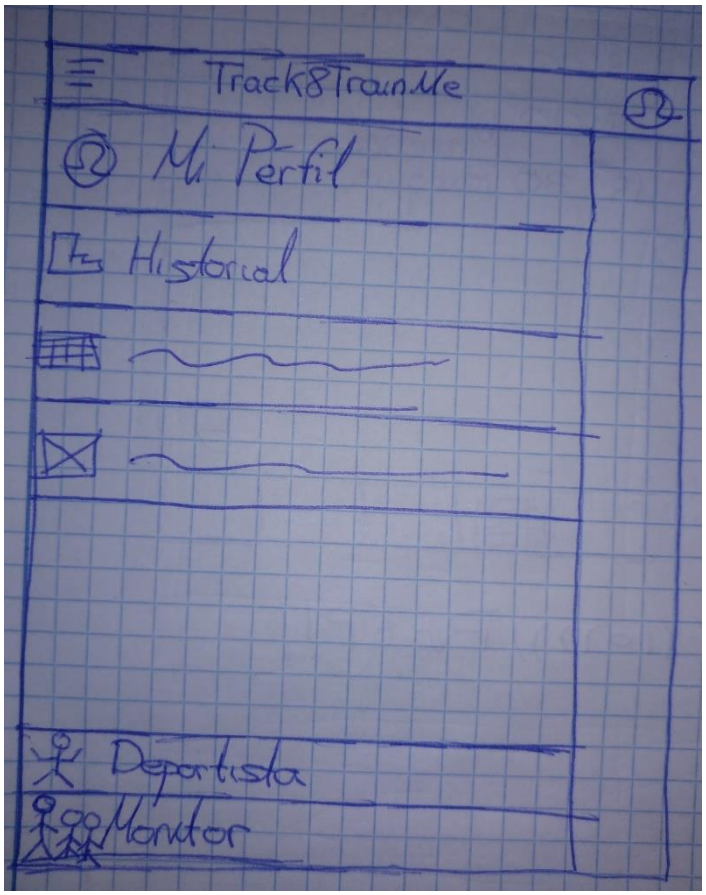


Figura 5.4: Prototipo inicial. Menú lateral

Figura 5.5: Prototipos realizados en Balsamiq

5.1.5 Prototipo final

Partiendo del prototipo inicial elaborado para la asignatura de DSI, decidimos realizar otra iteración sobre el mismo, para adecuarlo a las necesidades finales de este trabajo, ya que aunque ambos parten de la misma idea, el trabajo entregado para la asignatura de DSI está sobredimensionado, ya que teníamos total libertad de creación de funcionalidades debido a que no era necesario su posterior desarrollo e implementación.

Por lo tanto, realizamos una adaptación del prototipo inicial a los requisitos principales que tenemos que cumplir para la realización de este trabajo, obteniendo un nuevo prototipo en el cual nos basaremos para la realización de la interfaz de la aplicación. En esta primera adaptación, queremos comentar que nos hemos basado en realizar un diseño meramente funcional para poder probar las distintas funcionalidades que se han ido implementando para la aplicación, por lo que las imágenes que mostramos a continuación pueden sufrir distintas modificaciones que realizaremos posteriormente para mejorar el diseño y la usabilidad final de la aplicación.

Figura 5.6: Prototipo final. Vista principal

Figura 5.7: Prototipo final. Reconociendo actividad

Figura 5.8: Prototipo final. Vista del calendario.

Figura 5.9: Prototipo final. Vista del historial de un día en concreto.

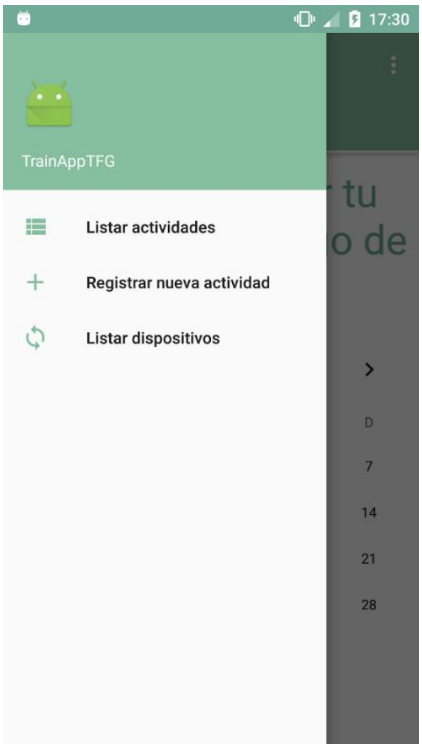


Figura 5.10: Prototipo final. Menú lateral.

Figura 5.11: Prototipo final. Lista de actividades registradas en el sistema.

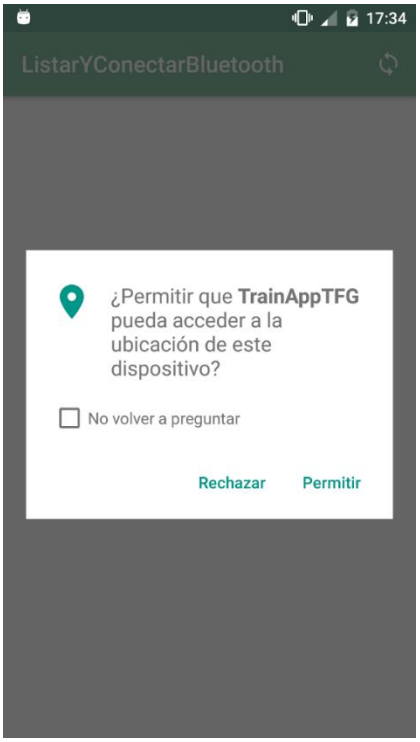


Figura 5.12: Prototipo final. Solicitud de permisos para activar el Bluetooth.

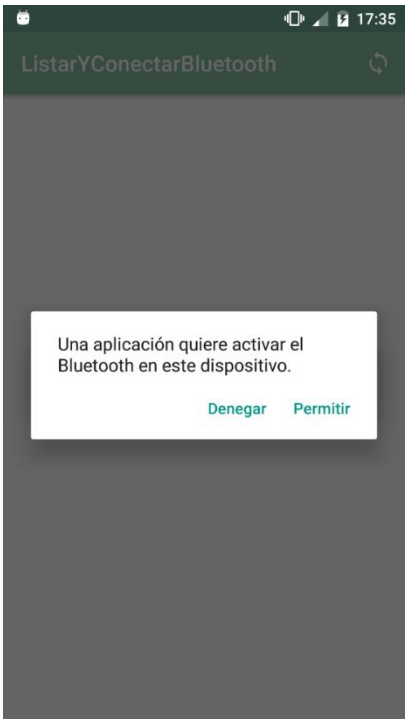


Figura 5.13: Prototipo final. Solicitud de activar el Bluetooth si está desactivado.

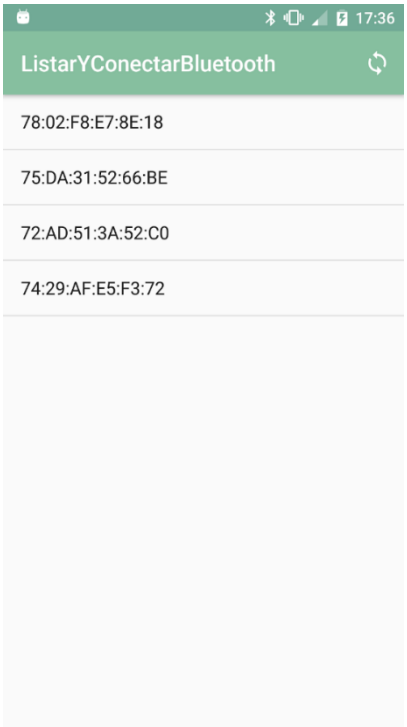


Figura 5.14: Prototipo final. Listado de dispositivos Bluetooth

Figura 5.15: Prototipo final. Vista de bienvenida del asistente de grabación de actividades.

Figura 5.16: Prototipo final. Formularios del asistente de grabación de actividades.

Figura 5.17: Prototipo final. Solicitud de permisos para poder guardar archivos en el teléfono del usuario.

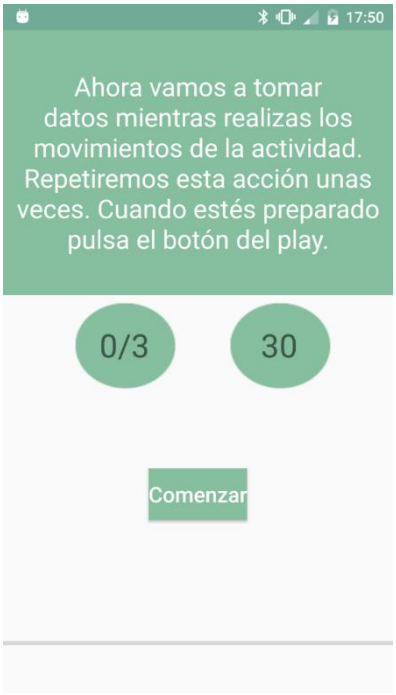


Figura 5.18: Prototipo final. Vista inicial del asistente listo para empezar a grabar para recoger datos.

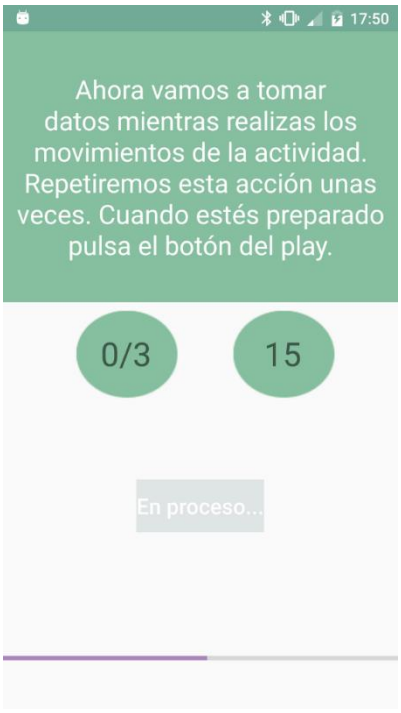


Figura 5.19: Prototipo final. Asistente en proceso de recogida de datos.

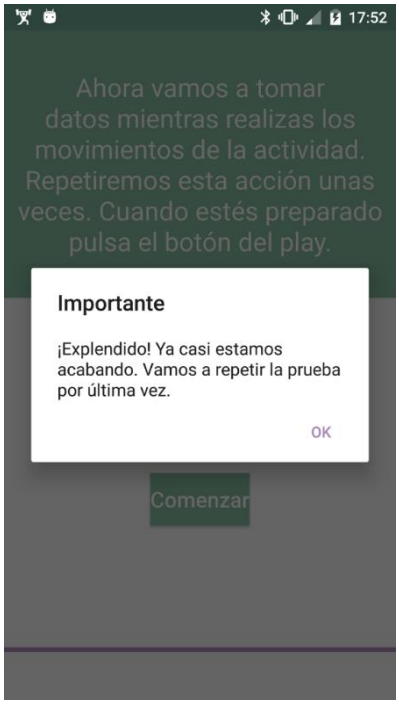


Figura 5.20: Prototipo final. Aviso por parte del asistente de que se ha grabado 2/3 de los archivos de datos.

5.2 Estructura de la aplicación del smartphone

En este apartado explicaremos cómo está estructurada la aplicación, de qué partes se compone y de qué funcionalidad se encarga cada parte del sistema. Para explicarlo utilizaremos un gráfico en el que se muestran los distintos componentes del sistema y cómo están relacionados:

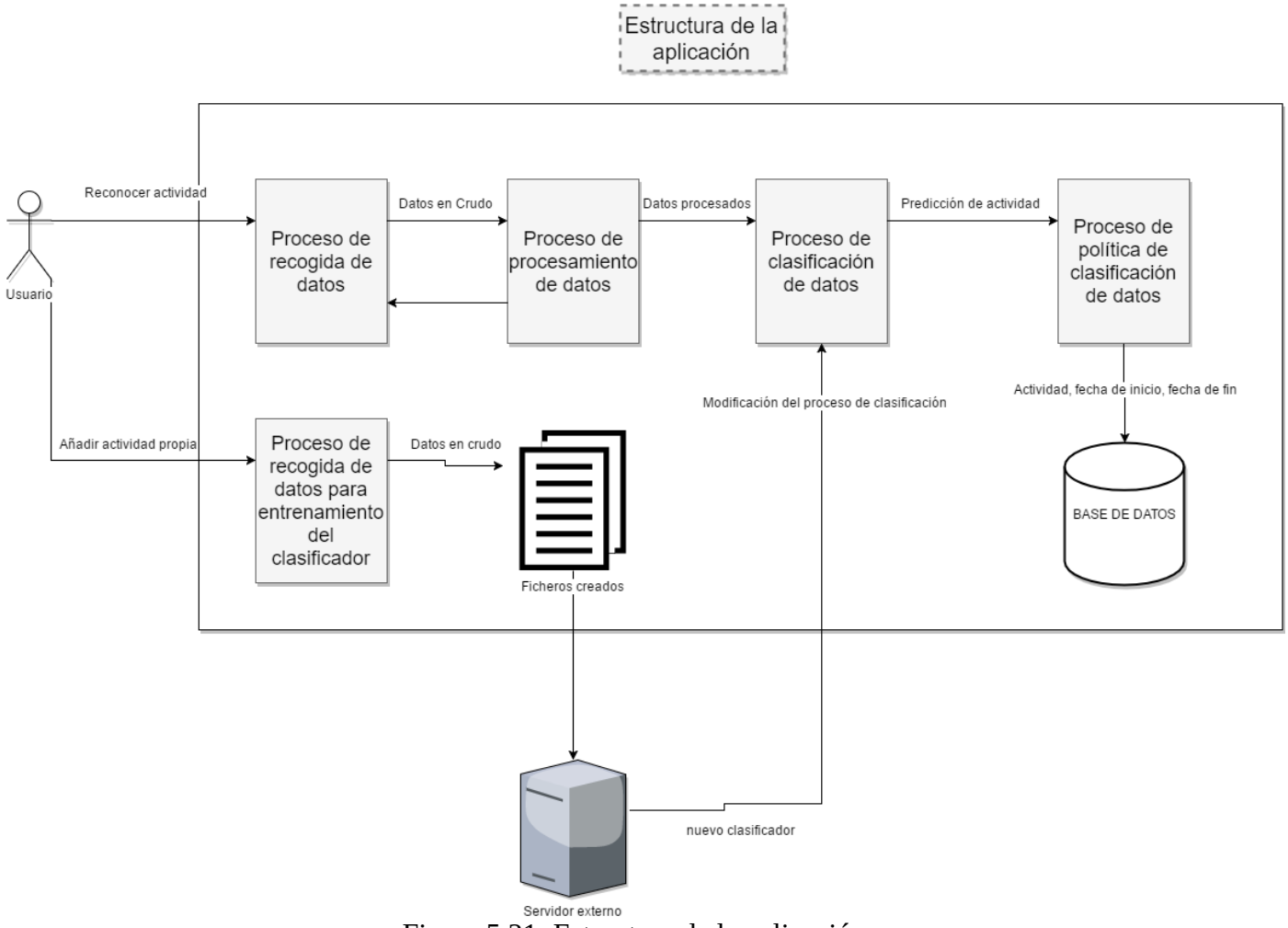


Figura 5.21: Estructura de la aplicación

A la vista del gráfico anterior, comentaremos brevemente en qué consisten cada una de las partes del sistema, ya que se explicarán en mayor profundidad en apartados posteriores.

Como podemos observar en el gráfico, hemos representado el funcionamiento a alto nivel de lo que son dos actividades bien diferenciadas:

- La primera funcionalidad consiste en que el usuario desea que la aplicación comience a monitorizar sus movimientos y que quede registro de las distintas actividades que ha realizado en el historial. Como podemos observar este proceso está compuesto por otros subprocesos que se encargan de realizar distintas funciones para finalmente poder reconocer la actividad que está realizando el usuario:
    - El primer subproceso es el de recogida de datos, el cual básicamente se encarga de recoger datos en crudo/bruto o RAW de los sensores del teléfono o de la pulsera (como ya hemos explicado anteriormente, nosotros tomamos los datos del acelerómetro y del giroscopio) en ventanas de un segundo de duración y le pasa estos datos al siguiente subproceso. Este primer subproceso se repite de forma continua desplazando la ventana medio segundo de duración, es decir, el 50% del tamaño de la ventana.
    - El segundo subproceso se encarga de procesar los datos del primer proceso, calculando las distintas características (media, mediana, desviación típica…) necesarias para utilizar el árbol de decisión entrenado de manera offline y envía estos datos al tercer subproceso.
    - El tercer subproceso recibe los datos procesados (es decir, con las características calculadas) del subproceso de procesamiento, pasa estos datos procesados por el árbol de decisión y como resultado proporciona un número que corresponde con la clase de actividad que se estaba realizando (por ejemplo, un uno es correr, un dos andar…).
    - El cuarto y último subproceso se encarga de recibir la la actividad que el árbol de decisión ha calculado que se está haciendo y ejecuta la política de clasificación de estados que hemos definido, que como explicaremos posteriormente es una máquina de estados. También se encarga de almacenar en la base de datos de la aplicación datos relevantes para el historial, como son la fecha de inicio, la fecha de fin y la actividad que se estaba realizando el usuario. Por ejemplo: (10/06/2016 13:15 – 13:20 Andar, 10/06/2016 13:20 – 13:25 Correr…).

- En cuanto a la segunda funcionalidad principal, esta consiste en que el usuario quiere añadir su propia actividad al sistema para que este sea capaz de reconocer

esta nueva actividad cuando el usuario la realice. Al igual que en la primera funcionalidad, se compone de distintos subprocesos:

- El primer subproceso, cuya funcionalidad es similar al primer subproceso de la primera funcionalidad, se encarga de recoger datos del usuario realizando la nueva actividad durante treinta segundos, en tres ocasiones. La diferencia está en que tras la finalización este subproceso genera tres ficheros de datos (en total un minuto y medio de duración) del acelerómetro y del giroscopio.
- El segundo subproceso, se encarga de enviar los ficheros de datos al servidor para crear y entrenar al nuevo árbol de decisión.
- El tercer y último subproceso se encarga de recibir el nuevo árbol de decisión, el cual sustituirá al anterior, siendo ya capaz de reconocer la nueva actividad que el usuario quiere que la aplicación reconozca.

En la aplicación del smartphone toda la parte de recogida, segmentación, procesamiento de los datos y clasificación las estamos realizando localmente en el smartphone, por lo que según explican en [3] nuestro tipo de clasificación entra dentro de la categoría de ”clasificación online”, que se diferencia de la “clasificación offline” en que en esta aproximación los datos son recogidos por el smartphone, pero estos son enviados a un servidor o cloud que se encarga de realizar todo el procesamiento y clasificación, y finalmente, le manda al cliente (en este caso el smartphone) el resultado final. Esta última aproximación requiere de conexión a Internet continua, lo cual implica un gasto de energía en la batería bastante elevado, por lo tanto, decidimos usar la aproximación de “clasificación online”, ya que actualmente los smartphones cuentan la necesaria potencia de cálculo para realizar estas operaciones sin ningún tipo de problema.

Aparte de las funcionalidades básicas definidas en la estructura principal de la aplicación, la aplicación también cuenta con otras funcionalidades que no hemos definido en este apartado ya que consideramos no son esenciales para el funcionamiento básico de la aplicación, y, por lo tanto, las explicaremos y definiremos en apartados sucesivos.

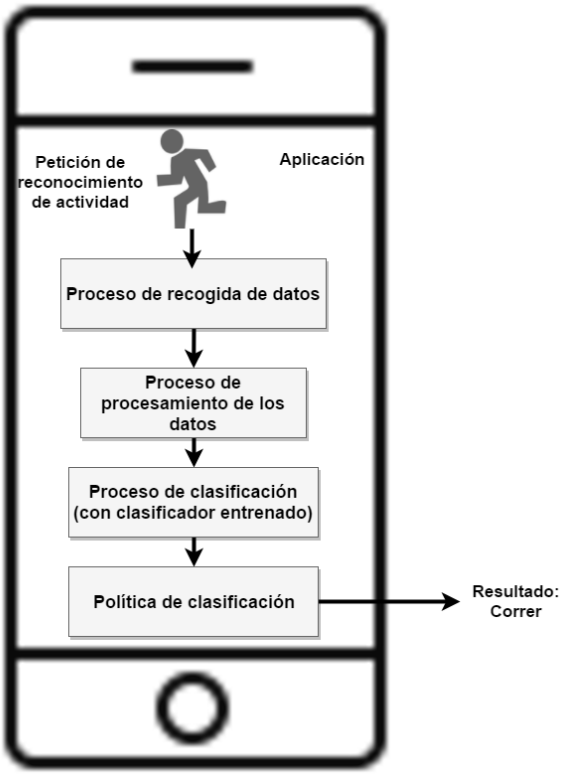


Figura 5.22: Aplicación. Clasificación online

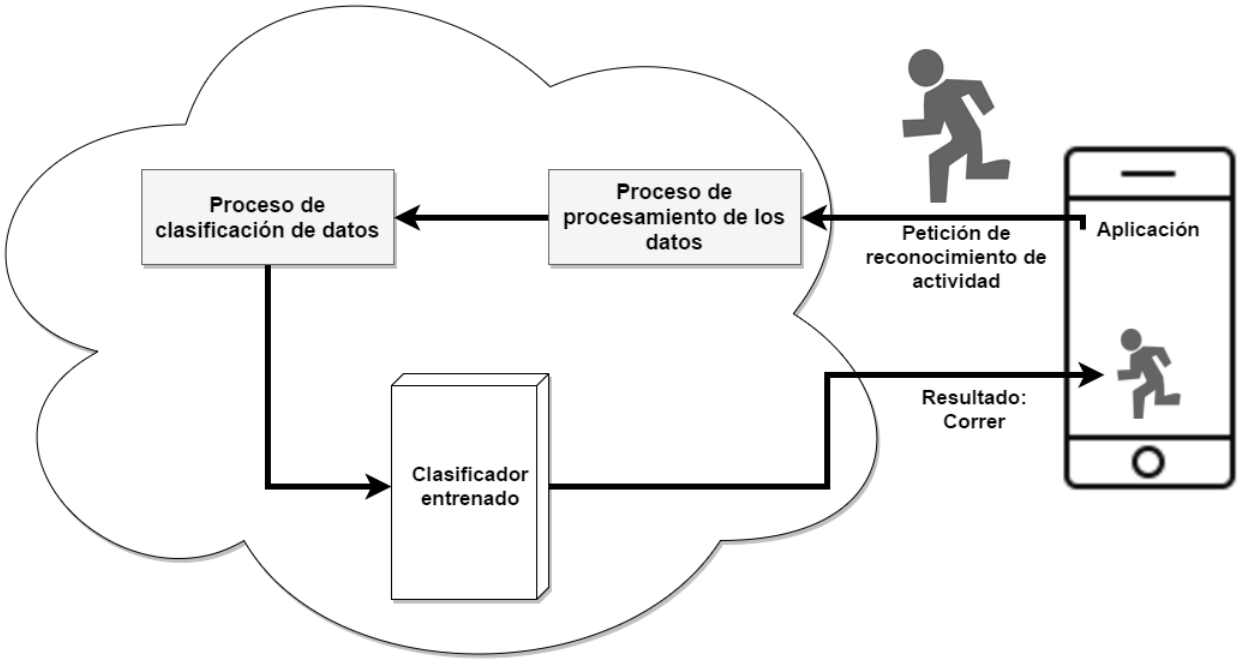


Figura 5.23: Ejemplo de clasificación Offline

5.3 Estructura de la aplicación del servidor

En este apartado explicaremos la estructura de la aplicación del servidor, la cual se encarga de realizar la personalización de las actividades para la aplicación del smartphone.

Para realizar esta personalización hay que entrenar al clasifcador y para realizar este proceso de entrenamiento para permitir al usuario crear nuevas actividades hemos optado por realizarlo en un servidor externo al smartphone (por lo que nuestro entrenamiento entra dentro de la categoría de “entrenamiento offline”), ya que esto es un proceso que según [3] es computacionalmente alto para el smartphone lo cual no es deseable ya que aumentaría el consumo de la batería del teléfono.

El servidor se encarga de recibir los datos de todas las actividades que el usuario quiere que la aplicación reconozca, este los procesa y los añade a un almacén de datos de actividades generalizadas, se inicia el proceso de entrenamiento del clasificador con estos datos y devuelve a la aplicación un nuevo clasificador ya entrenado, el cual es capaz de detectar la nueva actividad que el usuario ha registrado.

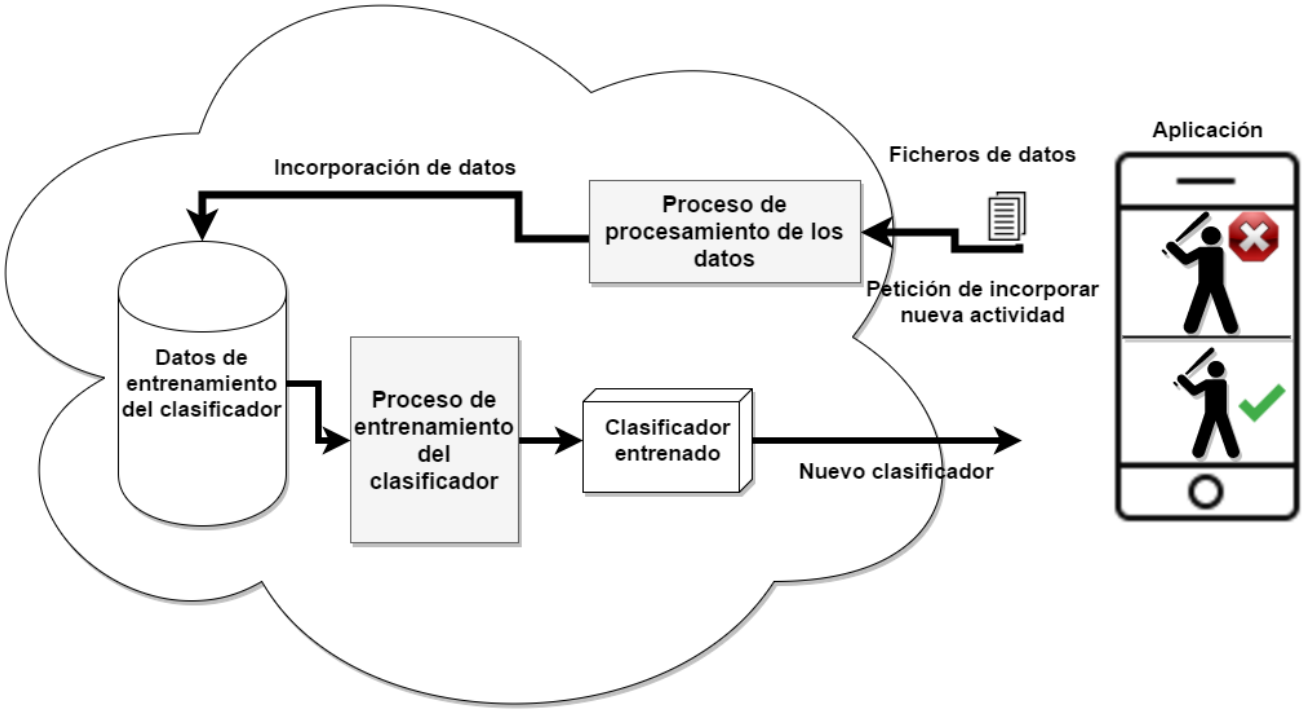


Figura 5.24: Aplicación. Entrenamiento offline

5.4 Implementación del proceso de reconocimiento de actividad

En este apartado explicamos en mayor profundidad y con mayor nivel de detalle la implementación que hemos llevado a cabo para desarrollar nuestra aplicación, dividiendo la implementación en cinco partes principales, cada una de las cuales se ha implementado como un hilo (o thread) que se ejecuta en un segundo plano de la aplicación: la recogida de los datos en crudo, el procesamiento que realizamos de estos datos en crudo, la clasificación, la política de clasificación que hemos implementado y por último cómo realizamos el entrenamiento offline del clasificador para permitir al usuario incorporar a la aplicación sus propias actividades.

5.4.1 Recogida de datos

Comenzaremos explicando cómo hemos implementado el punto de inicio del reconocimiento de actividad de la aplicación: la obtención de los datos necesarios y cómo los hemos obtenido. En cuanto a las fuentes de estos datos, nuestra aplicación soporta dos tipos distintos de fuentes:

- La primera fuente generadora de los datos son los propios sensores del teléfono, en concreto el acelerómetro y el giroscopio.
- El segundo origen de los datos también puede ser de un Wearable externo al smartphone, cómo puede ser la pulsera que nosotros hemos utilizado (SensorTag CC2650 Texas Instruments).

También queremos hacer mención a la librería Joinery [17], la cual hemos utilizado para gestionar los datos de los sensores de forma parecida a los DataFrames que se utilizan en Pandas o en R. Es de código abierto (GPL), sencilla de usar, documentada, y da buenos resultados en los tiempos de ejecución.

Así pues, pasaremos a explicar cada una de las maneras de las que hemos obtenido los datos, ya que aunque en un principio pueden parecer bastante similares, en realidad se diferencian bastante en la manera de acceder a los datos de los distintos sensores.

5.4.1.1 Recogida de datos – Smartphone

Antes de comenzar a explicar la implementación que hemos desarrollado para obtener datos de los sensores del smartphone, explicaremos las distintas funcionalidades o componentes que la API de Android pone a disposición de los desarrolladores para que podamos hacer uso de los sensores [18], información que podemos obtener en el cual también podemos observar códigos de ejemplos bastante útiles. Como hemos comentado, el Sistema Operativo Android consta de un framework para trabajar con los sensores que se llama Sensor Framework pertenece al paquete "android.hardware"), el cual proporciona distintas clases e interfaces que nos ayudan a realizar distintas tareas en cuanto a los sensores se refiere:

- **SensorManager:** utilizamos esta clase para crear instancias del SensorService, clase que proporciona varios métodos para acceder y listar los sensores que tenemos disponibles, registrar y desregistrar events listeners, y permite adquirir información de la orientación.
- **Sensor:** esta clase se utiliza para crear una instancia de un sensor específico y también proporciona diversos métodos para acceder a los datos y a la configuración del sensor.
- **SensorEvent:** el sistema utiliza esta clase para proporcionar información de que ha ocurrido un evento en el sensor. Generalmente este evento indicará que hay datos nuevos. La información que proporciona son los datos en crudo del sensor (raw data), el tipo de sensor que disparó el evento, el accuracy o precisión de los datos y el timestamp en el que ocurrió el evento.
- **SensorEventListener:** esta interfaz se encarga de definir las callbacks necesarias para que cuando ocurra un cambio en los valores de los datos del sensor o se modifique el accuracy del sensor, el listener recoja el evento (el cual recibirá un objeto de la clase SensorEvent).

Así, nosotros hemos seguido los distintos códigos de ejemplo que proporcionan en [18]. Entonces lo primero que tenemos que hacer para empezar a recoger datos de los sensores es obtener una instancia de la clase SensorManager, con la cual obtendremos instancias de Sensor, que representan los distintos sensores que queremos usar (en nuestro caso acelerómetro y giroscopio). Por último, a través de la instancia obtenida de SensorManager tendremos que realizar una llamada al método registerListener, el cual

registra una instancia que implemente la interfaz SensorEventListener, el sensor y el data delay (o frecuencia de muestreo). Este último parámetro, según la información proporcionada en [18], es meramente orientativo, ya que el sistema puede que use un delay menor que al especificado programáticamente.

En cuanto al parámetro de delay, nosotros realizamos distintas pruebas probando distintos delays, ya que deseábamos tener una frecuencia de muestreo de 10 Hz, es decir conseguir diez registros de datos por cada segundo, pero las pruebas realizadas cambiando el delay en la llamada al registerListener no dieron los resultados esperados y los timestamps entre los distintos registros de datos eran totalmente distintos, lo cual no cumplía con nuestros requisitos. Tras lo cual decidimos implementar una clase que almacenase el último valor de los datos de los sensores (y realizamos el registerListener esta vez con un valor de delay 0, es decir lo más rápido posible) y así controlamos la frecuencia exacta con la cual queremos acceder a los datos del sensor, aunque esto implica un mayor consumo de la batería. Esto viene motivado porque el framework de Android no permite modificar frecuencias de los sensores más allá de las pocas opciones que hay ya definidas. Es por ello, que aunque en la clase auxiliar anterior se almacene el último valor de los sensores a una frecuencia elevada, nosotros accederemos a estos valores desde afuera a una frecuencia igual o menor, permitiéndonos, ahora sí, controlar la frecuencia.

Una vez implementada la solución anterior, comprobamos que ya sí obtenemos los datos de los sensores con la frecuencia de muestreo exacta que nosotros queremos, ya que las diferencias entre los timestamps de los distintos registros de datos difieren aproximadamente 100 milisegundos entre dato y dato, lo cual cumple con nuestro requisito de tener una frecuencia de muestreo de 10 Hz.

Para obtener un dato cada 100 ms de la instancia que almacena el último valor proporcionado por el sensor y su timestamp, hemos utilizado un TimerTask [19] en combinación con un Timer [20]. La instancia de Timer se encarga de ejecutar repetidamente, con una frecuencia de 100 milisegundos (parámetro modificable a nuestra elección), la instancia del TimerTask, la cual posee el código que hay que ejecutar en segundo plano y que en nuestro caso será obtener los datos del acelerómetro y del

giroscopio, comprobar si ya tenemos una ventana de un segundo de datos y mandarle este conjunto de datos al proceso de extracción de características.

También realizamos una comprobación en la diferencia de tiempos de los timestamps entre los sensores del acelerómetro y del giroscopio, ya que ambos sensores en los smartphones se suelen encontrar separados (a diferencia del SensorTag CC2650 como explicaremos posteriormente), por lo que cada sensor tiene timestamps distintos y en nuestra aplicación los datos de ambos sensores tienen que estar en una única fila, es decir de la siguiente manera:

- timestamp, accel-x, accel-y, accel-z, gyro-a, gyro-b, gyro-g.

Tras realizar la comprobación de que la diferencia de timestamps entre registros de ambos sensores no era sumamente significativa decidimos juntar ambos datos en un único registro y el timestamp de este único registro es el generado tras la obtención de ambos registros. Tras esta "unificación de los datos" realizamos una última comprobación entre los tres registros de datos: el acelerómetro, el giroscopio y el registro con los datos unificados. Tras esta última comprobación, dimos por válido "unificar" con un único timestamp los datos de ambos sensores, ya que las diferencias de tiempos eran insignificantes.

5.4.1.2 Recogida de datos – Pulsera

Para la recogida de datos desde la pulsera SensorTag CC2650 el procedimiento es bastante similar al explicado para recoger datos de los sensores del teléfono, con la única diferencia de que para obtener datos de la pulsera tenemos que conectarnos vía Bluetooth al dispositivo para obtener los datos y realizar la configuración apropiada del dispositivo, además de configurar distintos permisos necesarios para que la aplicación pueda utilizar la funcionalidad del Bluetooth y sea capaz de conectarse a otros dispositivos a través de él. Ahora explicaremos cómo lo hemos implementado para la aplicación, ya que es un proceso que hay que seguir paso a paso. Comentaremos brevemente los elementos que son necesarios configurar para establecer una conexión Bluetooth entre el smartphone y la pulsera.

Para ello nos hemos basado en un tutorial [21] en el cual explica de manera detallada cómo podemos obtener los datos de humedad de una pulsera con sensores bastante similar a la nuestra.

5.4.1.2.1 Conexión Bluetooth

La conexión por Bluetooth de dispositivos a través de Android se caracteriza por dos puntos. Por un lado, se ha de seguir los pasos que el framework de Android proporciona para la conexión genérica de cualquier dispositivo. Por otro lado, se debe obtener, a partir de la documentación del fabricante del aparato, los identificadores necesarios para poder interactuar con las diferentes características del dispositivo. Este último punto limita a nuestra aplicación el poder conectar otras pulseras, pues hasta dentro de un mismo fabricante, diferentes productos de la misma categoría disponen de identificadores diferentes. Por lo tanto, en este apartado se explicará el proceso de conexión con la pulsera Sensor Tag CC2650.

La conexión se realiza mediante Bluetooth Low Energy. ¿Por qué el uso de esta tecnología? Todo viene motivado por el consumo. La pulsera se alimenta de una pila botón. El uso de este tipo de Bluetooth permite al sensor funcionar durante meses sin necesidad de cambiar la batería. Para poder interactuar con esta tecnología tendremos que hacer uso del paquete “android.bluetooth”.

En primer lugar, el usuario debe permitir el uso del Bluetooth por parte de la aplicación. Del mismo modo, deberá tener el Bluetooth activado en todo momento.

Con el Bluetooth activado ya podemos empezar a trabajar desde la aplicación. La app debe tener acceso al chip Bluetooth del teléfono móvil donde esté instalada. Para ello, Android nos proporciona la clase “BluetoothAdapter” [22]. A partir de esta clase, nosotros podremos escanear los dispositivos Bluetooth que tengamos a nuestro alrededor. Tenemos a nuestra disposición herramientas que nos avisen de cuándo se empieza a escanear o en qué momento se ha encontrado algún dispositivo. Estos métodos son muy útiles de cara a la implementación de la parte gráfica de la aplicación, pues cuando un dispositivo cercano se encuentra, se informa al usuario de tal evento.

Entre los dispositivos Bluetooth encontrados, el usuario deberá elegir con cual quiere conectarse. Para poder conectarnos, haremos uso de la clase "BluetoothGatt" [23], la cual está optimizada para trabajar con la tecnología Bluetooth Low Energy. Esta clase está muy relacionada con la clase "BluetoothGattCallback" [24], pues nos proporciona una serie de métodos que nos permite estar informados en todo momento del estado de la conexión con el dispositivo, así como las distintas interacciones que hagamos con él.

Una vez estemos conectados a un dispositivo tendremos que entender diferentes partes del mismo. En concreto, tres partes relacionadas entre sí: concretar un Service, para poder descubrir las distintas Características del dispositivo, y los Descriptores.

Estos elementos tienen su origen en la naturaleza "GATT" (Generic Attributes Profile), la cual expone una estructura de datos que proporciona el acceso jerárquico del dispositivo. El sensor al que nos conectamos contiene un servidor GATT, con el cual establecemos conexión. Este pequeño servidor contiene varios servicios (Service), los cuales representan diferentes tipos de datos que podemos intercambiar con el servidor.

La pulsera utilizada en nuestro proyecto dispone de multitud de sensores: acelerómetro, giroscopio, humedad, sensor de luz… Cada Service se encarga de trabajar con cada uno de estos sensores. A su vez, un Service contiene varias Características. Nosotros podemos obtener datos y comunicarnos con la pulsera gracias a este elemento, al cual le puede acompañar varios Descriptores. Con el siguiente cuadro queremos ilustrar esta jerarquía:

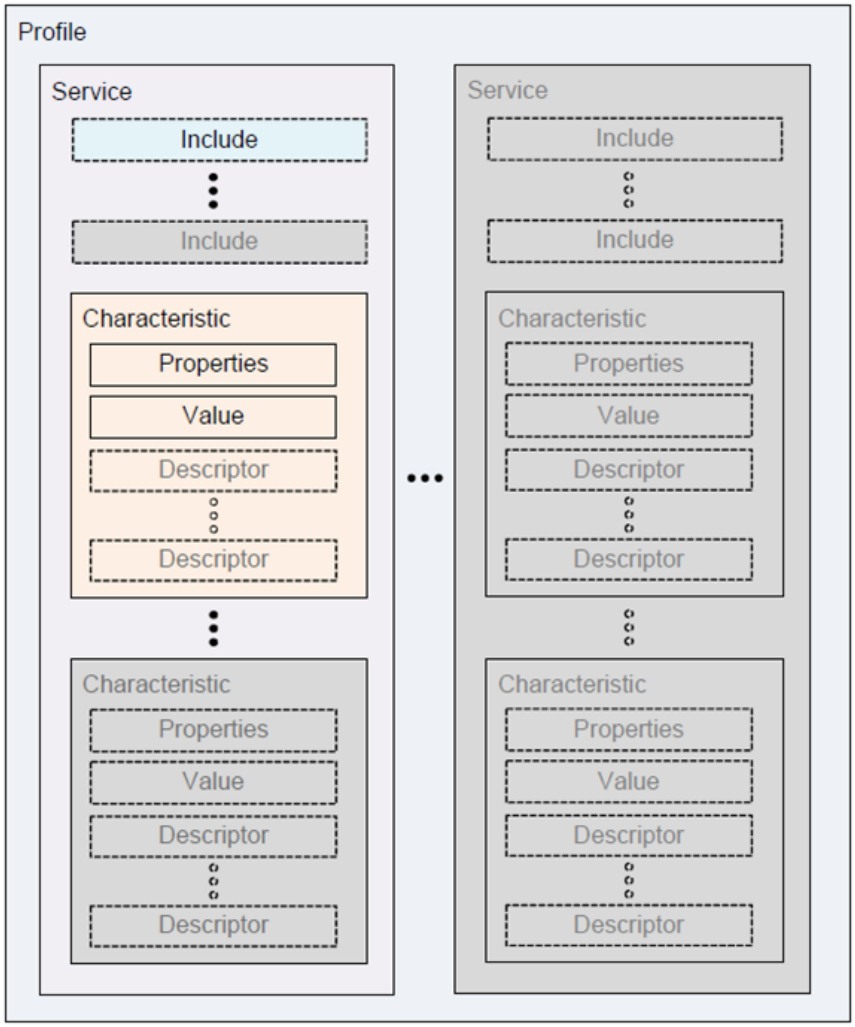


Figura 5.25: Jerarquía Gatt Profile. Fuente: Bluetooth.com

Como nosotros solamente vamos a trabajar con el acelerómetro y con el giroscopio, deberemos obtener el Service que se encargue de dichos sensores. Para poder obtener este Service deberemos de echar mano a la documentación que nos proporciona el fabricante del hardware al cual queremos conectarnos. Observamos que el modelo de pulsera con el que vamos a trabajar, el acelerómetro y el giroscopio conviven en el mismo chip (MPU-9250). Para conseguir detectar al Service, así como todas sus características, debemos buscar una identificación, denominada UUID. En la tabla proporcionada por el fabricante disponemos de todos estos identificadores: enlace.

Una vez localizado el UUID del Service correspondiente para trabajar con nuestro par de sensores, procederemos a obtenerlo. Una vez tenemos el Service, obtendremos sus Características (debemos buscar sus correspondientes UUID. Trabajaremos con tres

Características: los datos, la configuración y el periodo (la frecuencia) sobre el que trabajará el sensor. Una vez hayamos obtenido toda esta información, estaremos listos para poder poner toda la maquinaria a funcionar.

5.4.1.2.2 Configuración de sensores

Cuando ya disponemos de todas las características, deberemos hacer uso de ellas. Por ejemplo, antes de usar los sensores, debemos de configurarlos a nuestra necesidad.

Bits	Usage
0	Gyroscope z axis enable
1	Gyroscope y axis enable
2	Gyroscope x axis enable
3	Accelerometer z axis enable
4	Accelerometer y axis enable
5	Accelerometer x axis enable
6	Magnetometer enable (all axes)
7	Wake-On-Motion Enable
8:9	Accelerometer range (0=2G, 1=4G, 2=8G, 3=16G)
10:15	Not used

Tabla 6: SensorTagCC2650. Configuración de sensores. Fuente: processors.wiki.ti

En la tabla anterior podemos observar que podemos indicar a la Característica para configurar el sensor. Como se comentaba en el apartado anterior, el acelerómetro y el giroscopio se encuentran ubicados en el mismo chip, por lo tanto, comparten el mismo Service. Esto implica que debemos configurarlos a la vez. Para la configuración disponemos de 16 bits (dos bytes). Los 6 primeros nos permiten habilitar los ejes de los sensores del acelerómetro y del giroscopio. Nosotros vamos a trabajar con los tres ejes de cada sensor, por lo tanto, deberemos marcar los correspondientes bits a 1. El magnetómetro no será utilizado, por lo tanto, su bit valdrá 0. Lo mismo ocurre con la funcionalidad "Wake-On-Motion". Los bits 8 y 9 nos sirven para establecer el rango de los sensores (como de preciso van a ser los movimientos). Utilizamos el rango 8G al asemejarse al de los smartphones. Este rango le corresponde el valor 2, por lo tanto, el bit

9 será un 1 y el bit 8 será un 0 (10 = 2 en binario). Como se indica en la tabla, los bits comprendidos entre el 10 y el 15 no se utilizan, por lo que tendrán un valor de 0.

Para entender de una manera más clara la explicación anterior, en la siguiente figura se puede observar los diferentes bits, con ceros (0) para desactivar y con unos (1) para activar.

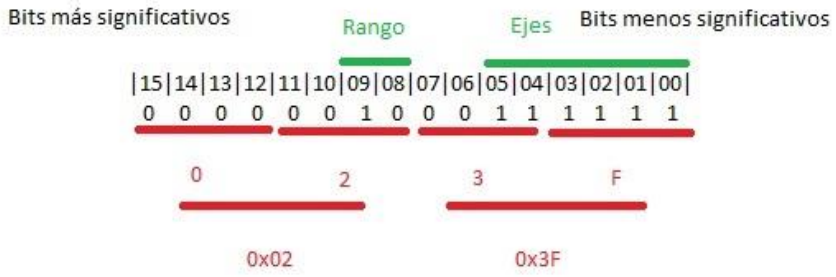


Figura 5.26: SensorTagCC2650. Bits configuración sensores.

Las características trabajan con número hexadecimales, por lo que hay que realizar la correspondiente conversión.

Por lo tanto, si nosotros trabajamos con la configuración de la imagen, debemos de escribir en la Característica “Configuración” el array de bytes 0x3F, 0x02 (en el array de bytes, los menos significativos van primero). Una vez hayamos escrito la configuración, en cuanto activemos el sensor por medio de la escritura de otra Característica, el sensor empezará a funcionar tal y como lo hayamos configurado.

5.4.2 Procesamiento de los datos

Tras haber recogido los datos, ya sea de los propios sensores del smartphone o de la pulsera, enviamos estos datos en crudo o raw al proceso de extracción de características, el cual recibirá registros de datos en ventanas de un segundo de duración y además los datos están solapados un 50%, es decir, medio segundo de solapamiento entre ventana y ventana, como ya explicamos en el apartado 5.2.

Para mantener la consistencia, la coherencia y para que la clasificación funcione adecuadamente, tenemos que realizar los mismos cálculos (ya explicados anteriormente en el apartado 5.2.) que se realizan en el entrenamiento offline del árbol de decisión en un PC de sobremesa, lo cual nos supuso un pequeño inconveniente, ya que en el entorno de sobremesa utilizábamos diversas funciones de Pandas que te calculaban de una manera muy sencilla las características. El inconveniente es que en el entorno de Android no podemos utilizar el lenguaje de Python sin hacer uso de terceras aplicaciones (QPython [25], por ejemplo), y además, estas aplicaciones están pensadas para poder desarrollar otras aplicaciones en Python en vez de en Java para Android, que es en lo que nosotros estamos desarrollando la aplicación.

Intentamos buscar diferentes soluciones para tratar de solucionar este pequeño inconveniente, a ver si existía alguna solución para invocar scripts de Python desde el código de Java de Android (algo similar a Jython [26], pero para Android, ya que también probamos a utilizar Jython pero no es compatible con el compilador de Java de Android). También probamos otras soluciones como SL4A [27], pero no se consiguieron los resultados esperados, ya que el cambio de contexto del código de Java al código de Python consumía un tiempo considerable que no podíamos permitirnos, por lo que a falta de una solución adecuada para este inconveniente, decidimos realizar los cálculos en el propio código de Java de Android, utilizando para ello la librería Apache Math [28], la cual nos proporciona clases y métodos para realizar distintos cálculos matemáticos complejos, como pueden ser la transformada rápida de Fourier (FFT) o la correlación entre varias variables.

Una vez que ya podemos calcular las distintas características de los datos, que en nuestro caso recordamos, como ya comentamos en el apartado 5.2, son el mínimo, el

máximo, la media, la mediana, la desviación estándar, correlación entre los tres ejes del acelerómetro y la transformada rápida de Fourier (FFT) también de los tres ejes del acelerómetro. Entonces, ya podemos proceder a enviar estos datos procesados al proceso de clasificación, el cual describimos con mayor detalle en el siguiente apartado.

5.4.3 Clasificación de los datos

Una vez hemos recogido los datos y los hemos procesado para extraer las características, podemos enviar estos valores al clasificador entrenado previamente para que nos indique qué actividad ha calculado que estamos realizando. Esto es, si el clasificador dice si estamos andando, barriendo, aplaudiendo o quietos…. El árbol de decisión se puede representar de manera sencilla mediante código Java en una estructura de if-elses que dada una entrada de datos procesados nos retorna la salida, es decir, la actividad que el clasificador detecta que estamos realizando. Si el usuario ha añadido nuevas actividades personalizadas a la aplicación, el clasficador que empezará a utilizar es el último árbol que devolvió el servidor.

Tras obtener la salida del clasificador (árbol de decisión), enviamos este resultado al siguiente proceso, el cual implementa la política de clasificación. En el apartado 5.4.5 describiremos con mayor detalle cómo hemos desarrollado e implementado el entrenamiento offline del clasificador que estamos utilizando en este proceso.

Comentar que no tomamos directamente la salida del clasificador para realizar las distintas gestiones de la aplicación ya que (mostrar la actividad que está realizando, almacenamiento en el historial…), ya que si no el sistema sería muy susceptible a cambiar de forma muy frecuente, por ejemplo, con una tasa de aciertos del 80% en el clasificador es de esperar que una de cada cinco ventanas analizadas sea clasificada incorrectamente, por lo que para controlar esta situación y hacer la aplicación más resistente a este tipo de fallos realizamos un proceso de "refinado", para lo cual hemos definido una política de clasificación mediante máquina de estados que explicaremos en mayor profundidad en el siguiente apartado. En resumen, lo que queremos explicar en este párrafo es que, cuando una persona realiza una actividad, la finaliza y de seguido comienza a realizar otra actividad, los movimientos que la persona hace durante la transición entre actividades puede provocar que el clasificador clasifique de manera incorrecta actividades que el

usuario no está realizando, por lo que con la ayuda de una política de clasificación podemos suavizar estas transiciones para que, por ejemplo, si una persona camina, y de repente empieza a aplaudir, entre media no nos salga barrer, entre otros posibles resultados inesperados.

5.4.4 Política de clasificación implementada

Como explicamos en el apartado anterior, hemos optado por implementar una política de clasificación que controle de una manera más apropiada los resultados proporcionados por el proceso de clasificación, consiguiendo así un efecto de control sobre lo que finalmente se va a mostrar al usuario por la pantalla de la aplicación y permitiéndonos la posibilidad de cambiar o modificar esta política a nuestro criterio, pudiendo ser esta, más estricta o más laxa. La política puede ser más estricta o menos dependiendo de diversos factores, como puede ser el porcentaje de precisión del clasificador entrenado o por criterios que queremos imponer.Para implementar esta política hemos definido una máquina de estados, que para poder explicarla mejor representaremos en el siguiente gráfico, simplificado a dos actividades, aunque se extiende trivialmente a N actividades:

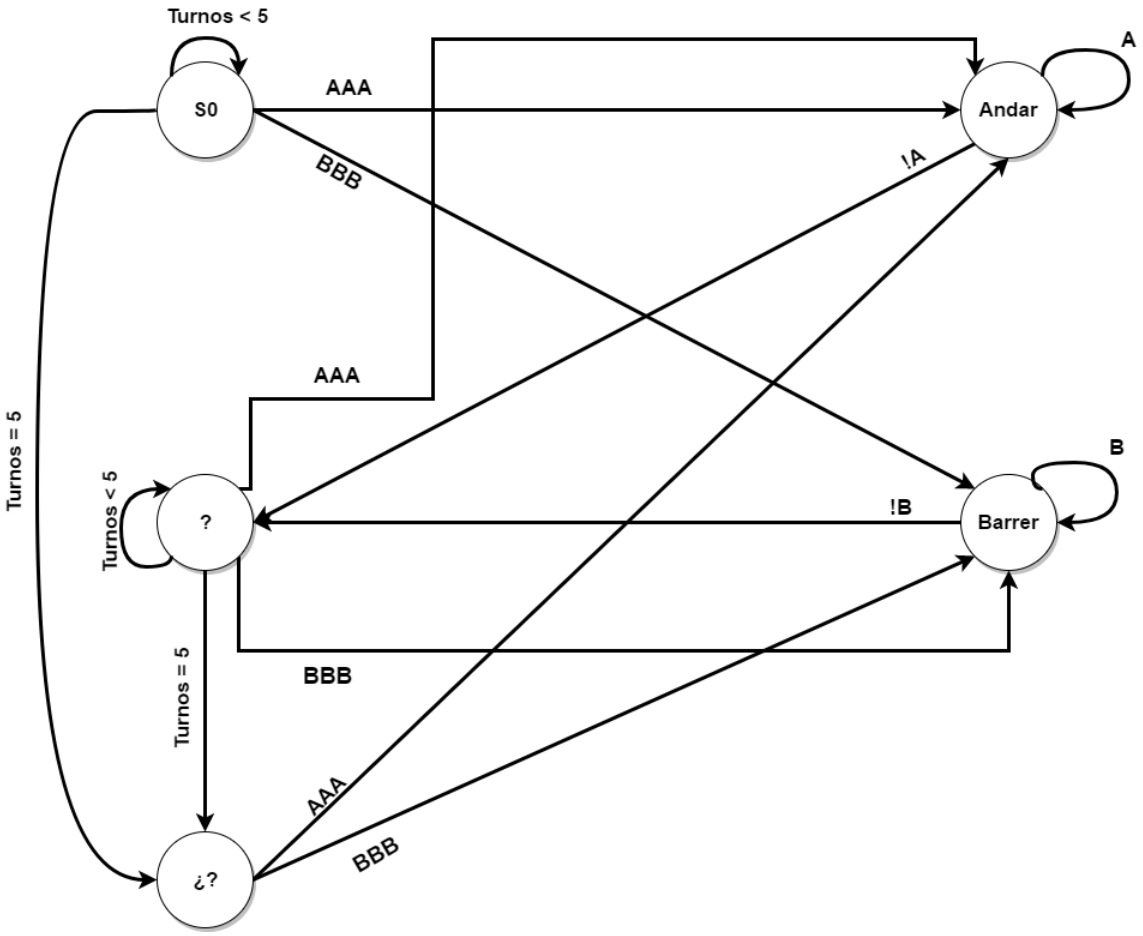


Figura 5.27 Diagrama de la máquina de estados de la política utilizada.

A la vista del gráfico anterior explicaremos más detalladamente los estados y la transición entre los distintos estados:

- **Estado S0 o inicial:** este es el estado inicial de la aplicación. No se sale de este estado hasta que lleguen tres actividades clasificadas iguales (entonces se pasa al estado de la actividad correspondiente), en cambio si llegan 5 letras (es decir, que se han clasificado los datos y se ha alimentado a la máquina de estados con 5 actividades clasificadas) y no hay tres seguidas iguales se pasa al estado "¿?", el cual nos indica que no sabemos qué actividad está realizando.
- **Estado "?" o dudoso:** este estado nos indica que estábamos ya en un estado de actividad reconocida, pero que estando en ese estado se ha detectado la llegada de un cambio de actividad. Entonces en este estado permanecemos en "alerta" para un posible cambio de actividad (pero no se actualiza la interfaz gráfica hasta que no vuelvan a llegar otras tres letras iguales seguidas y volvamos a uno de los estados de actividad). Igual que el anterior si analizamos 5 letras y no hay 3 letras

iguales seguidas, el siguiente estado es el estado' ¿?'. Con este estado conseguimos evitar diferentes interferencias que se puedan evitar mientras se realiza una actividad. Por ejemplo, si una persona está corriendo y mueve una mano para secarse la frente de sudor (movimiento corto, rápido y repentino), con este estado evitaríamos que el clasificador mostrara otra actividad entre medias.

- **Estado "¿?" o muy dudoso:** si hemos llegado a este estado, quiere decir que no hemos podido detectar adecuadamente qué actividad se está realizando según la política definida, es decir, durante una duración de cinco ventanas o tres segundos (ya que las ventanas se desplazan de medio segundo en medio segundo). No se sale de este estado hasta que lleguen tres letras iguales seguidas y vayamos nuevamente a otro estado de actividad reconocida.

También comentar que la implantación de esta política de estados, nos ha facilitado el desarrollo del historial de actividades, ya que cuando entramos en un estado de actividad reconocida, podemos almacenar el timestamp en el cual se entró a dicho estado y en cambio si hemos llegado al estado "?" o dudoso desde un estado de actividad reconocida y nos vamos a otra actividad distinta de la anterior, o al estado "¿?" o muy dudoso, podemos marcar también el timestamp de finalización de dicha actividad, y con estos datos ya contamos con todo lo necesario para la implementación del historial.

5.4.5 Conexión con el servidor para la actualización del clasificador

Para realizar la sustitución del árbol viejo por el nuevo, hacemos uso de BeanShell [29], una herramienta que parsea cadenas de String y que, posteriormente, lo transforma a un lenguaje script orientado a objetos, que es transformado a un AST (árbol de sintáxis abstracta) [30]. Cuando un usuario se graba realizando una actividad nueva, los archivos de datos se envían al servidor para que se procesen. Una vez se ha generado el nuevo árbol que incluye la nueva actividad, esta se envía en formato String a la aplicación embebido en un método y se almacenará en la base de datos. De tal modo que, cuando el usuario arranque el clasificador, la aplicación cogerá de la base de datos el último árbol actualizado. Se hará uso de un intérprete que nos proporciona la herramienta BeanShell. El intérprete parseará y evaluará el método que contiene el nuevo árbol una sola vez (con

el objetivo de ganar en rendimiento, ya que así no tiene que construir un nuevo AST cada vez que se ejecute, sino que utilizarrá ya el construido anteriormente) antes de comenzar a clasificar y una vez finalizado el proceso, la aplicación ya estará lista para funcionar con el nuevo árbol entrenado.

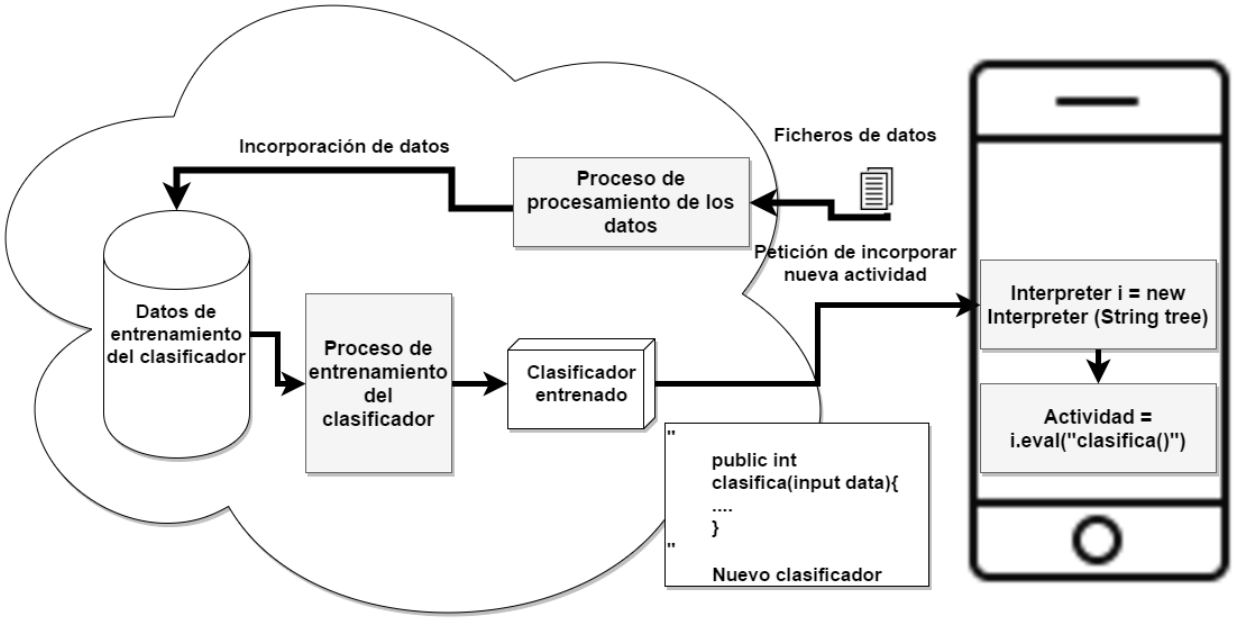


Figura 5.28: Aplicación. Proceso de incorporación de nueva actividad

5.5 Otros aspectos importantes de la aplicación

Una vez explicada todas las partes esenciales para el funcionamiento de la aplicación y tras haber detallado cómo hemos implementado el proceso de reconocimiento de actividad en la aplicación, también explicaremos en este apartado otros aspectos que consideramos importantes y que han ido surgiendo según hemos ido implementando las distintas funcionalidades.

5.5.1 Interfaz gráfica

Como bien sabemos, es enorme la diversidad de tamaños y resoluciones de las pantallas que tienen los distintos smartphones, por lo que las distintas aplicaciones tienen que ser capaces de visualizarse correctamente en la mayoría de dispositivos para abarcar

una mayor cuota del mercado y ampliar el espectro de dispositivos que puedan utilizar la aplicación adecuadamente.

Por los motivos anteriores, hemos decidido que nuestra aplicación también sea compatible en distintos tipos de pantallas. De entre varias soluciones que ofrece Android para tratar de que las aplicaciones nosotros decidimos usar el layout de organización de la vista "PercentiveLayout", mediante el cual podemos indicar el tamaño que queremos que ocupe cada elemento gráfico de la interfaz respecto al elemento padre, lo cual nos posibilita que la aplicación funcione en distintos tamaños de pantalla, ya que los elementos gráficos se adaptan automáticamente según el porcentaje de ocupación que le hayamos otorgado. Encontramos bastante información sobre cómo utilizarlo en [31].

5.5.2 Ficheros

Para el almacenamiento de los datos recogidos para el posterior entrenamiento offline del árbol de decisión (es decir, el envío de los datos para realizar entrenamiento del clasificador en el servidor), necesitamos almacenar estos datos en distintos ficheros en local (de aproximadamente 30 KB cada fichero de treinta segundos), es decir, dentro del smartphone.

Para almacenar estos ficheros, necesitamos pedir permiso al usuario cuando este esté utilizando la aplicación. No basta con declarar estos permisos en el manifiesto, ya que a partir de la versión Marshmallow de Android (6.0), se llevó a cabo un cambio importante en lo relacionado a la gestión de los permisos del sistema.

Además, decidimos almacenar estos ficheros en la memoria interna del teléfono, ya que actualmente los dispositivos cuentan con la capacidad interna de almacenamiento suficiente y así no tenemos que depender de que el smartphone tenga o no una tarjeta externa de almacenamiento.

5.5.3 Bases de datos

Como ya hemos explicado en apartados anteriores, nuestra aplicación proporciona al usuario la opción de visualizar un historial con las distintas actividades que ha realizado,

así como su duración. En esta ocasión de todas las opciones de persistencia que ofrece Android, elegimos almacenar estos datos en una base de datos relacional para facilitar las distintas gestiones con los elementos del sistema.

Android incorpora un motor de bases de datos que es muy popular actualmente, ya que ofrece distintas características, como su reducido tamaño, no necesita estar implementado en un servidor, necesita poca configuración, es transaccional y además de código libre como el propio Android. Nos referimos al motor de bases de datos SQLite.

Su uso es bastante sencillo y permite realizar las distintas operaciones de inserción, eliminación y actualización de modo parecido a MySql. Además, utilizarlo a través del cliente que proporciona para Java es bastante sencillo de utilizar.

Para la aplicación hemos definido dos tablas:

- **Tabla de actividades:** esta tabla tiene dos columnas que son de distintos tipos: la primera almacena un identificador numérico y representa a la actividad y la segunda almacena en modo texto el nombre de esa actividad.
  CREATE TABLE actividades(id INTEGER PRIMARY KEY AUTOINCREMENT,actividad TEXT NOT NULL UNIQUE, fechaCreacion DATETIME DEFAULT (datetime('now', 'localtime'))))
- **Tabla de historial de actividades:** en esta tabla almacenamos la información necesaria para representar el historial de actividades que el usuario ha realizado. Tiene cuatro columnas, la primera es un identificador numérico, la segunda se corresponde a la actividad, la tercera a la fecha de inicio y la cuarta a la fecha de fin.

  CREATE TABLE historial(id INTEGER PRIMARY KEY AUTOINCREMENT,actividad INTEGER NOT NULL,fechaIni DATETIME NOT NULL,fechaFin DATETIME, FOREIGN KEY (actividad) REFERENCES actividades(id) ON DELETE CASCADE)

Además, también hemos utilizado el patrón Transfer para gestionar y representar los objetos de las tablas.

5.5.4 Servidor

Como hemos comentado en reiteradas ocasiones, en nuestro trabajo hemos optado por realizar el entrenamiento de nuestro árbol de decisión de manera offline, ya que entre otros motivos está el elevado coste computacional que supone esta tarea para el smartphone. También ha influido en la toma de esta decisión que para realizar este entrenamiento hayamos utilizado clases que proporciona el paquete Scikit-learn [13], el cual solamente está disponible para ejecutar en Python y no para Java, apareciendo el mismo inconveniente que en la fase de procesamiento de datos, en la cual no podíamos utilizar Pandas en el código Java para realizar los distintos cálculos de las características.

Por las razones anteriores hemos decidido montar un servidor que pueda ejecutar Python (y, por lo tanto, también Scikit-learn), el cual se encarga de recibir los ficheros con los datos, los cuales procesaremos y entrenaremos al nuevo árbol de decisión que enviaremos a la aplicación y sustituiremos por este nuevo árbol por el antiguo. Hemos utilizado Bottle [32] (Framework Web) como herramienta para nuestro servidor ya que es rápido, sencillo y es un WSGI ligero para Python el cual nos ha resultado muy fácil de usar. Para realizar las correspondientes peticiones al servidor, hemos utilizado la librería asynchttp [33], la cual gestiona adecuadamente de realizar la petición POST al servidor con los ficheros necesarios para realizar el entrenamiento del clasificador.

5.5.5 Comunicación entre Threads

Como hemos explicado anteriormente, hemos dividido la funcionalidad principal de la aplicación en distintos Threads que se ejecutan en un segundo plano de la ejecución principal, pero y ¿cómo se comunican y se envían datos de unos de unos procesos a otros en Android? Bueno, para responder a esta pregunta hemos encontrado y probado distintas soluciones que Android ofrece, como por ejemplo pueden ser la clase Parcelable [34] para los Services, enviar los datos directamente en el Intent al Service correspondiente (el problema de esta última solución es que solamente se pueden enviar los tipos básicos definidos en Java como String, Boolean, Integer..., no se puede mandar un Object general) o también que el objeto que queramos enviar implemente la clase Serializable y enviar la instancia “envuelta” en la clase que implemente esta interfaz (esto último cumple la

misma funcionalidad que Parcelable, pero Serializable no está optimizado para transferencias de objetos en Android).

Como vemos, las anteriores soluciones nos sirven para comunicación entre distintos servicios de Android, pero ¿y para la comunicación entre distintos Threads?.

Para implementar la comunicación entre los distintos Threads que forman la cadena de trabajo de la aplicación queríamos utilizar algo similar a tuberías o pipes, las cuales aprendimos cómo están implementadas y cómo usarlas en la asignatura de Arquitectura Interna de Linux y Android (LIN), y además tenían que implementar también algún mecanismo de sincronización. En este aspecto, los distintos Threads que componen nuestra aplicación siguen un esquema similar al problema del productor-consumidor (con sus correspondientes aspectos de sincronización a tener en cuenta y a tratar) en el sentido de que un Thread se encarga de producir nuevos datos y otros de consumirlos, procesando por ejemplo y enviándolo a su vez al siguiente Thread.

Investigando, encontramos una interfaz que cumplía con todos nuestros requisitos para poder establecer la comunicación entre los distintos Threads: la interfaz BlockingQueue [35]. Esta interfaz nos ofrece diversos métodos para añadir instancias de Object a la cola y además implementa el mecanismo de sincronización que requerimos para que los Threads solo se ejecuten en caso de que haya datos en su cola de consumición de datos (obligando a esperar al Thread hasta el momento en el que disponga datos a consumir). Esta última solución nos ha parecido una solución bastante sencilla, a la vez que útil y completa, lo único que hay que hacer es inicializar estas colas, pasarselas a los distintos Threads y utilizar métodos de inserción y extracción (en nuestro caso utilizamos las llamadas bloqueantes put() y take()) . Hemos usado estas colas para la comunicación y envío de datos entre los distintos procesos que componen la aplicación del smartphone: recogida de datos, procesamiento, clasificación y política de clasificación. Así tenemos una cola entre cada uno de los procesos que se ejecutan de manera continua mientras se esté realizando el reconocimiento de actividad.

5.5.6 Wakelocks

Por último, comentar que hemos utilizado Wakelocks para mantener a la CPU del Smartphone activa mientras se está realizando el reconocimiento de actividad en segundo plano, para mantener al dispositivo "despierto" aún cuando el usuario no lo esté usando directamente (es decir, con la pantalla apagada), ya que los dispositivos Android cuando llevan un rato sin utilizarse entran en modo de bajo consumo pudiendo así dejar de ejecutar aplicaciones que estaban ejecutándose en segundo plano, lo que ocasionaría la detención del proceso de reconocmiento de nuestra aplicación. Hay distintos tipos de Wakelocks en Android [36]:

- **Parciales:** mantienen la CPU activa, en cambio la pantalla no.
- **Totales:** mantienen la CPU y la pantalla activas.

A la vista de la anterior diferenciación, nosotros hemos usado un Wakelock de tipo parcial ya que solamente queremos mantener la CPU activa mientras estamos ejecutando el proceso de reconocimiento de actividad en segundo plano y no la pantalla también (un ejemplo de Wakelock de tipo total es la aplicación de Youtube, que mantiene además de la CPU, la pantalla activa mientras estamos reproduciendo un vídeo). Aunque puede parecer de gran utilidad, hay que tener especial cuidado y controlar su uso, ya que si por ejemplo adquirimos un Wakelock que no liberamos nunca aunque hayamos cerrado la aplicación, esto va a ocasionar que la CPU se mantenga activa aún cuando no es necesario, lo cual implica un mayor gasto de batería del dispositivo. A continuación mostramos distintos gráficos de usos correctos e incorrectos de Wakelocks [36]:

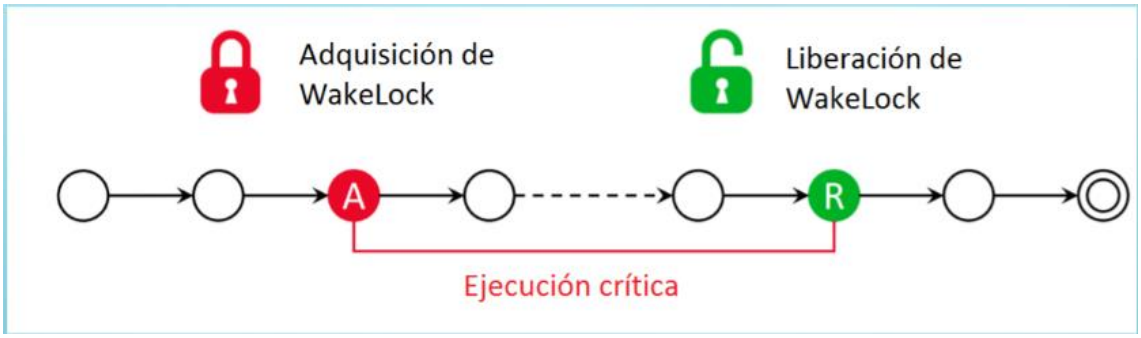


Figura 5.29: Uso correcto Wakelock

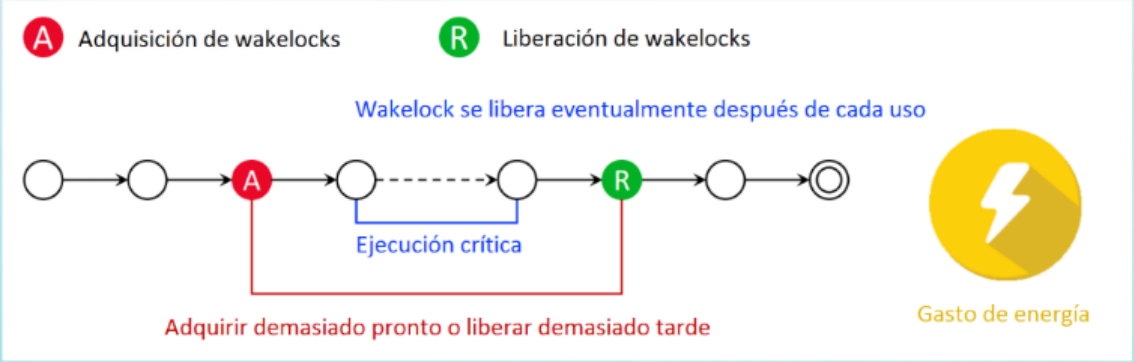


Figura 5.30: Uso incorrecto Wakelock. Liberación destiempo.

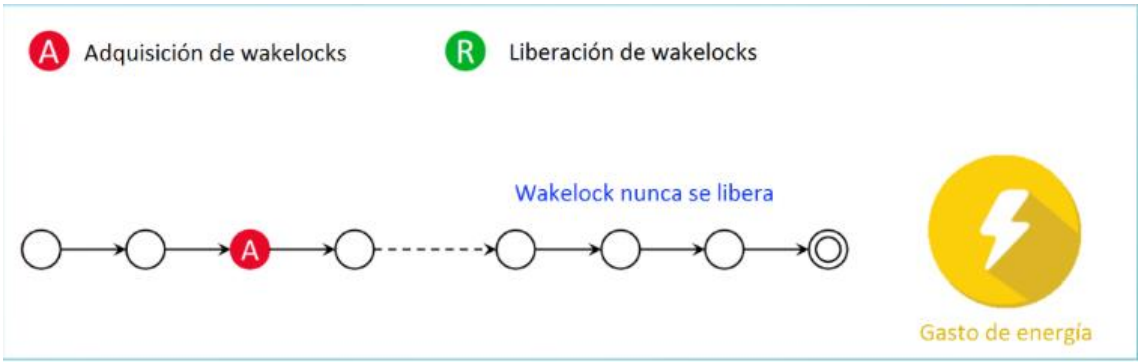


Figura 5.31: Uso incorrecto Wakelock No se libera.

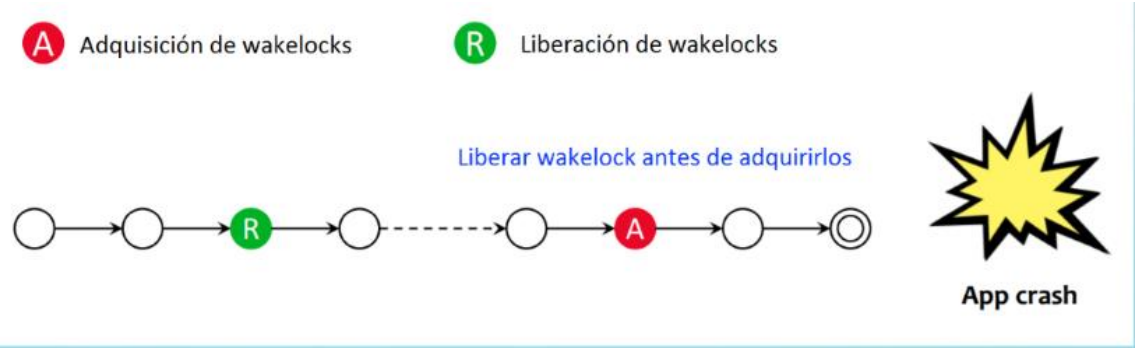


Figura 5.32: Uso incorrecto Wakelock. Liberar antes de adquirir.

6 Implementación del clasificador de actividad mediante árboles de decisión

En este capítulo vamos a explicar todas las pruebas que hemos realizado para ver la precisión de nuestro sistema de reconocimiento de actividades, detallando el dataset utilizado, las métricas empleadas y las matrices y resultados obtenidos, tanto con los datos obtenidos con la pulsera, como los obtenidos a través de los sensores del móvil. El primer apartado tratará sobre los experimentos realizados en el ordenador y el segundo sobre la experiencia de usuario a través del móvil.

6.1 Experimentación en PC

6.1.1 Datos de la pulsera

En este apartado se va a explicar el primer experimento que hicimos: obtener datos de la pulsera conectada al ordenador por conexión Bluetooth.

6.1.1.1 Dataset

Los datos se obtuvieron de cuatro personas diferentes. La pulsera estaba colocada en la mano izquierda (se hizo bastante hincapié en que la posición de la pulsera fuera exacta en todas las personas) y se obtuvieron datos del giroscopio (tres ejes: gamma, betta y Alpha) y del acelerómetro (tres ejes: X, Y y Z). Se realizaron cinco actividades:

- Andar
- Barrer
- Estar de pie
- Subir escaleras
- Bajar escaleras

Los datos, como explicamos en el capítulo cinco, fueron tomados tres veces durante 30 segundos por cada persona y actividad.

6.1.1.2 Medidas extraídas

Los datos fueron procesados en ventanas deslizantes, como explicamos en el capítulo 3, de tamaño de ventana de 1 segundo y con solapamiento de 0.5 segundos. Se extrajeron en total siete medidas:

- Media.
- Mediana.
- Desviación típica.
- Máximo.
- Mínimo.
- Correlación sólo en los acelerómetros
- Transformada de Fourier, solo en los acelerómetros.

6.1.1.3 Resultados

Como técnica para clasificar utilizamos los árboles de decisión. Para realizar la prueba, dividimos el dataset, dejando a tres personas para datos de entrenamiento y una para validación, este proceso se repitió rotando a la persona que se queda en validación. Esto es una forma de implementar k-fold cross validation de una forma realista ya que permite ver el impacto de una persona nueva a usar el clasificador. A continuación, mostramos los resultados obtenidos.

Lo primero que hicimos fue graficar el árbol. En cada nodo se muestra el número de ejemplos de cada actividad y la métrica que se ha utilizado para hacer la división. En verde se puede distinguir la actividad de barrer, en azul la de subir escaleras, en naranja la de andar, en morado bajar escaleras y estar de pie de un azul verdoso. Este árbol nos permite ver de manera gráfica qué métricas son más importantes para realizar una correcta clasificación y qué actividades se distinguen más fácilmente o por el contrario, cuáles se confunden más.

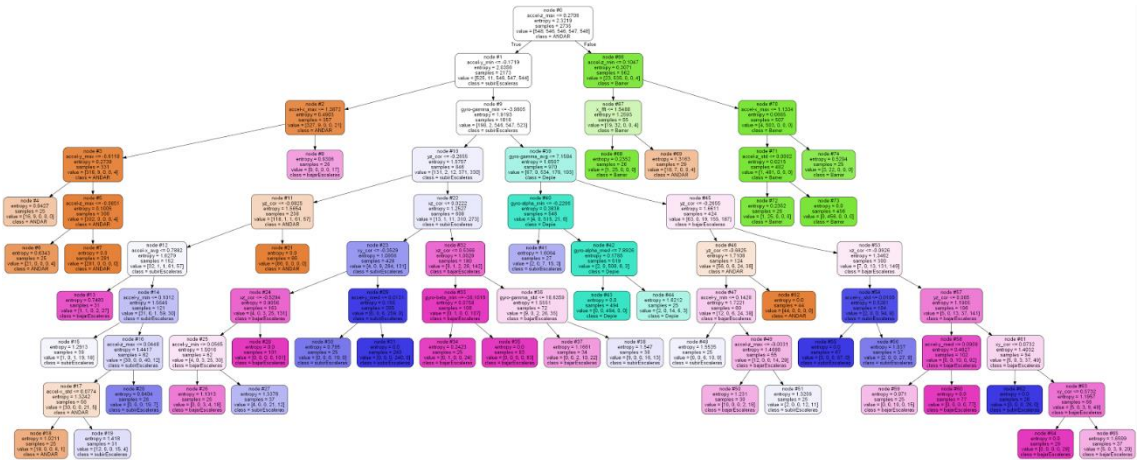


Figura 6.1: Árbol graficado. Pulsera. Experimento PC

A continuación, calculamos la “la salida del árbol de decisión, que es la actividad que el árbol pronostica. Entrenamos el árbol con los datos de las actividades de tres personas y utilizamos los datos de la cuarta persona para comprobar o validar qué actividad predecía. El resultado obtenido fue que se acertaba un 68 %, es decir su accuracy. También calculamos las métricas explicadas en el capítulo 4, recall, accuracy y f1-score. Los resultados son los siguientes:

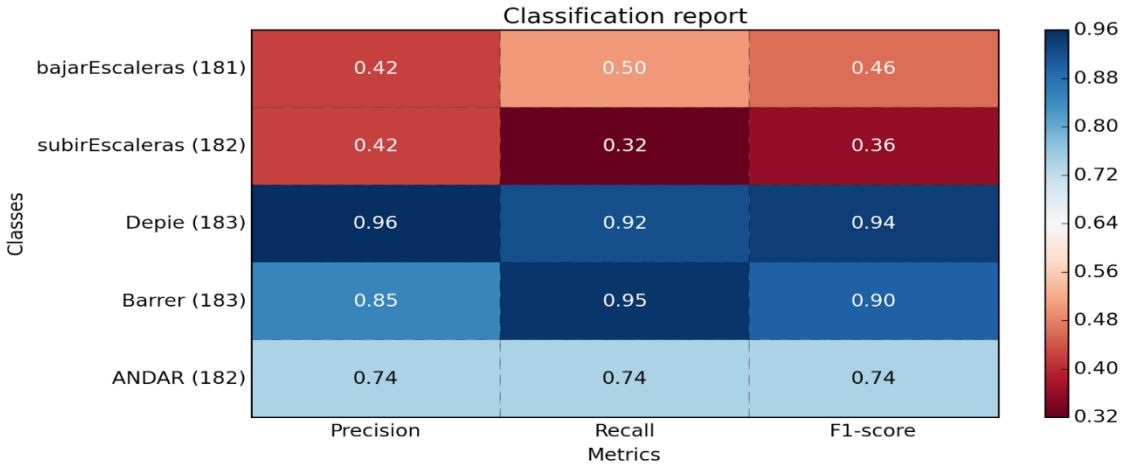


Figura 6.2: Recall, Accurracy, F1-score. Pulsera. Experimento en PC

Como podemos ver subir y bajar escaleras falla (actividades que se confunden con otras actividades), mientras que las otras tres las distingue bastante bien (cuanto más azul oscuro mejor tasa de acierto). Este alto grado de confusión de las actividades se debe a que las actividades de bajar y subir escaleras tienen bastante similitud entre ellas y a la vez los movimientos para realizarlas son muy parecidos a otras como andar. Otra causa de la confusión es que hemos contado con una cantidad de datos muy baja.

A continuación, mostramos los resultados obtenidos al calcular la matriz de confusión con los datos de entrenamiento. Podemos ver qué actividades se confunden con más frecuencia y cual se distinguen más fácilmente.

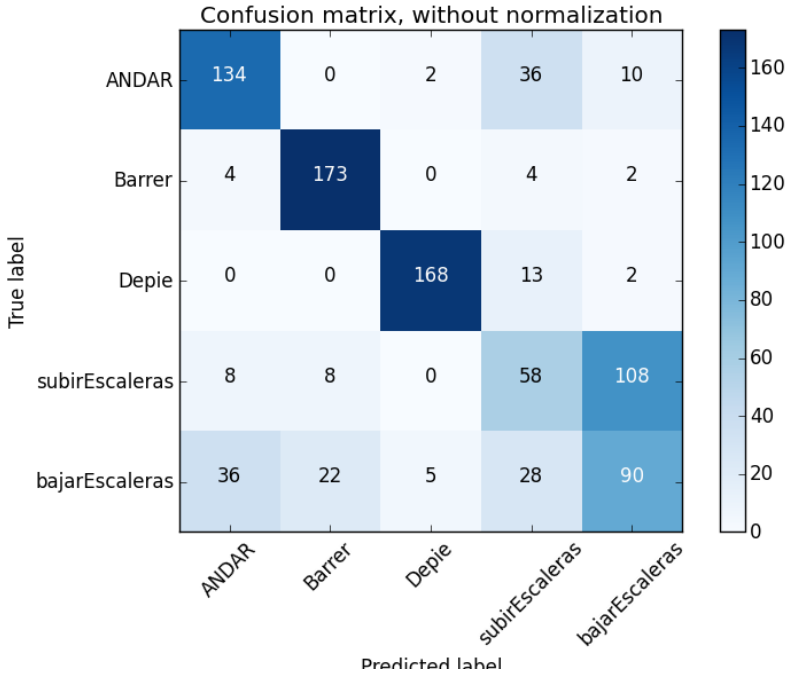


Figura 6.3: Matriz de confusión. Pulsera. Experimento en PC.

Como podemos ver, subir escaleras dice 108 veces que es bajar escaleras, y solo 58 veces de 183 dice que correctamente es subir escaleras. Pero por el contrario, por ejemplo de 183 ejemplos de barrer, distingue 173 bien.

Por último, para saber qué métricas eran más importante, y así poder reducir el número de características y disminuir el tiempo de clasificación y el uso de recursos para conseguir ser más eficientes, decidimos sacar el porcentaje de uso de cada una de ellas.

	gyro-alpha	gyro-betta	gyro-gamma	accel-x	acce-y	accel-z
Media	0	0	0.12	0.006	0	0
Minimo	0	0.013	0.0003	0.09	0	0.14
Máximo	0.012	0	0	0	0.012	0.006
Std	0.325	0	0	0.002	0.0008	0.0029
Corr	0.0007			0.0007	0.063	0.061
Fft				0.097	0.004	0
Mediana	0	0.009	0	0	0.008	0

Tabla 7: Porcentaje de uso de métricas. Pulsera. Experimentación en PC

Utilizamos también como técnica de clasificación, Random Forest. Su precisión fue del 70%. Solo mejoraba 0.2% a los árboles de decisión. Decidimos quedarnos con los primeros debido a que no daba un aumento considerable de la precisión y sin embargo si son más costosos, hablando en medidas de recursos utilizados y debido a su facilidad para convertir a código que pudiera leer una aplicación Android.

6.1.2 Datos del móvil

En este segundo apartado vamos a explicar la creación del árbol de decisión que posteriormente utilizamos en la aplicación del móvil.

6.1.2.1 Dataset

Tras analizar los resultados anteriores, nos dimos cuenta que podíamos mejorar mucho la precisión de nuestro reconocedor de actividad. Para el siguiente experimento decidimos realizar las actividades con un número mayor de personas, en total 16, y de diferente género y edad para que los resultados fueran más genéricos y pudiera analizar mejor cualquier usuario que no estuviera entre los anteriores.

Como en el experimento anterior se realizaron las actividades durante 30 segundos tres veces, pero esta vez el dispositivo utilizado fue el móvil, con la app que nosotros mismos hemos creado. La posición del teléfono siempre era en la muñeca de la mano izquierda y se realizaron cuatro actividades distintas:

- Andar
- Aplaudir
- Barrer
- Estar quieto

6.1.2.2 Medidas extraídas

Los datos fueron procesados en ventanas deslizantes de 1 segundo, con solapamiento de 0.5. Las medidas extraídas fueron las siguientes:

- Media
- Mediana
- Máximo
- Mínimo
- Desviación típica
- Correlación
- Energía de Fourier
- RMS
- RFFT

- Media de los valores absolutos de Fourier
- Suma de los valores absolutos de Fourier
- Desviación de los valores absolutos de Fourier
- Skew
- Kurtosis

Al final decidimos quedarnos con las siete primeras métricas solamente, lo que permite tener un árbol más ligero de procesar en el móvil y no afecta a la precisión del sistema.

6.1.2.3 Resultados

Como hicimos en el experimento anterior, dividimos los datos en dos conjuntos distintos. Los datos de 15 personas serían para entrenamiento y los datos de una única persona para validación del clasificador. Con esto queremos simular el comportamiento de una persona que nunca haya utilizado nuestro sistema y así poder comprobar cómo reacciona nuestro clasificador ante otras personas y datos nuevos. Este proceso lo repetimos varias veces como explicamos en el apartado anterior de los datos de la pulsera.

Primero pintamos el árbol resultante con todas las métricas antes descritas:

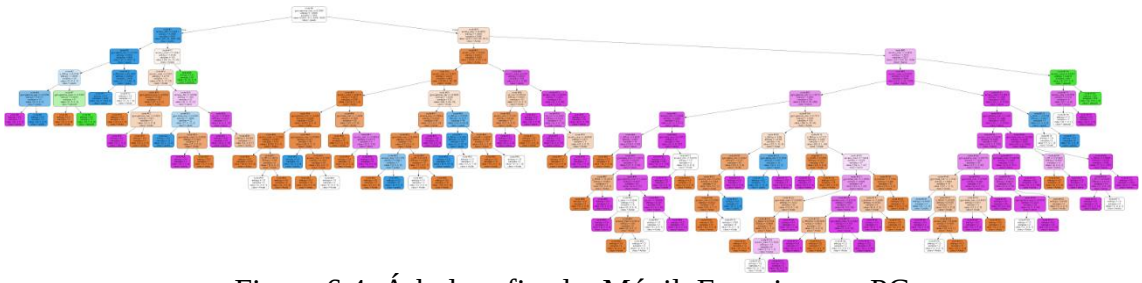


Figura 6.4: Árbol graficado. Móvil. Experimento PC

En azul se puede ver estar quieto, en morado barrer, en naranja andar y en verde aplaudir.

A continuación, calculamos la “y” predicha con los datos de validación para calcular la precisión del sistema. Esta dio un 98%. Habíamos mejorado considerablemente respecto al experimento anterior.

Tras calcular la precisión del sistema, calculamos las métricas recall, accuracy y f1-score y la matriz de confusión que a continuación mostramos:

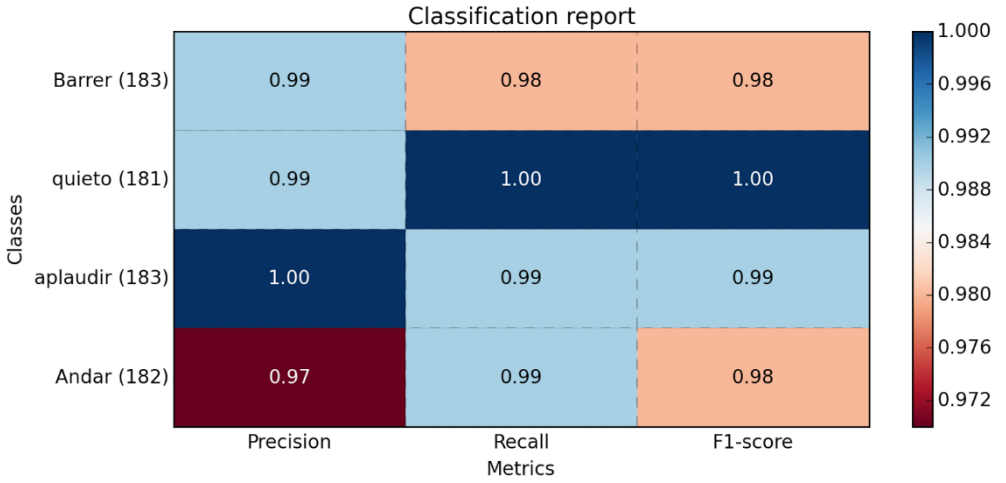


Figura 6.5: Recall, Accurracy, F1-score. Móvil. Experimento en PC

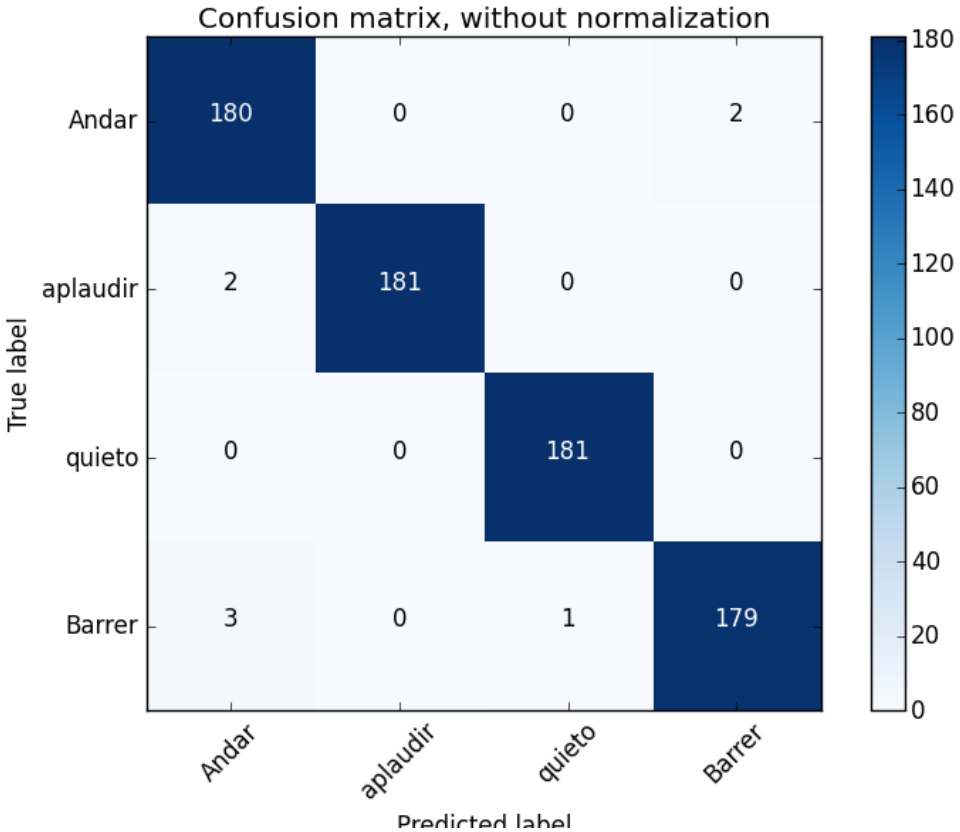


Figura 6.6: Matriz de confusión. Móvil. Experimento en PC

Como podemos ver nuestro reconocedor apenas falla, debido a la variedad de personas y que son actividades que se pueden diferenciar entre sí.

Para saber qué métricas eran más importantes realizamos la siguiente tabla:

	gyro-alpha	gyro-betta	gyro-gamma	accel-x	accel-y	accel-z
Media	0.00000000e+00	2.26613258e-03	1.33995092e-03,	1.76825945e-03	2.54541126e-01	2.98178414e-03
Minimo	6.16646659e-03	5.04188120e-03	1.39381305e-02	5.55077132e-04	2.02608996e-02	2.43462186e-03
Máximo	3.19348074e-04	2.95577680e-03	4.06506781e-01	4.48006900e-02	1.59399988e-01	1.38438932e-03
Std	5.71032585e-04	0.00000000e+00	0.00000000e+00	5.03210701e-04	3.27762879e-02	0.00000000e+00
Corr				0.00000000e+00	8.78760027e-04	9.96713959e-04
FFT(Numpy)				9.37507426e-04	0.00000000e+00	0.00000000e+00
Skew				4.20505348e-04	6.91849484e-04	3.40717033e-04
Kurtosis				0.00000000e+00	3.79826386e-04	2.66794941e-04
FFT (Scipy)				0.00000000e+00,	0.00000000e+00,	0.00000000e+00
sum_fft				7.37487426e-04	6.07533950e-04	0.00000000e+00
media_fft				6.16433055e-03	1.33868499e-03	1.53284343e-03
std_fft				6.16293177e-04,	4.16505531e-04	0.00000000e+00
Rms				0.00000000e+00	9.44082568e-04	0.00000000e+00,
Rfft				0.00000000e+00	3.86222318e-04	1.30989238e-04
Mediana	4.74136479e-04	6.15522204e-04	5.26365186e-04	6.29133126e-03	1.24968737e-02	1.29628966e-03]

Tabla 8: Porcentaje de uso de métricas. Móvil. Experimentación en pc

Como comprobación, para saber si en nuestro sistema podíamos reducir el número de características sin afectar a la precisión, decidimos calcular la precisión solamente con las siete primeras métricas y también realizar las pruebas con el clasificador Random Forest. En el primero de los casos fue del 97,5% y en el segundo de 99%. Con estos resultados pudimos ver que con muchas menos características la precisión apenas disminuyó y que utilizando Random Forest mejoraba, pero no lo suficientemente considerable como para utilizarlo.

6.2 Personalización del clasificador

Para implementar la funcionalidad de crear un clasificador personalizado con nuevos datos de un usuario, hemos desplegado un servidor (explicado en mayor detalle en el capítulo cinco) que se encarga de recibir datos nuevos, procesarlos y entrenar el clasificador con estos nuevos datos procesados.

En el servidor disponemos de unos datos procesados que hemos recogido de varias personas que han realizado las cuatros actividades que por defecto tiene la aplicación (barrer, andar, estar quieto y aplaudir). Cuando un usuario quiere añadir una nueva actividad con datos personalizados, lo que hacemos es recibir estos nuevos datos directamente de la aplicación (en forma de ficheros), los procesamos extrayendo las mismas características que las calculadas en los datos procesados de las actividades por defecto (los cuales están almacenados en una ubicación fija del servidor) y almacenamos en otra ubicación del servidor los archivo de datos procesados por cada actividad extra, (distinta a las actividades por defecto), tras lo cual los unificamos en un único dataset de datos nuevos procesados. Tras procesar y unificar los nuevos datos, los concatenamos con los datos procesados de las actividades por defecto, obteniendo así un último dataset unificado con todas las actividades.

Tras haber procesado los datos nuevos y haberlos unificado con los datos de las actividades por defecto, ya tenemos todos los “ingredientes” necesarios para entrenar nuestro nuevo árbol de decisión personalizado, cuyo entrenamiento se encuentra explicado en el capítulo cuatro. Después de realizar el entrenamiento del clasificador,

obtenemos la representación del árbol en formato string de if-elses de Java para su posterior utilización en la aplicación Android, explicado en el capítulo cinco.

7 Conclusiones y trabajo futuro

Para finalizar este proyecto analizaremos todo el trabajo realizado y las posibles mejoras que se pueden llevar a cabo.

7.1 Conclusiones

Tras finalizar este proyecto y haber analizado todo lo recogido en la presente memoria, es la hora de realizar las conclusiones finales, que esperemos que sirvan para ayudar a otras personas en el proceso de reconocimiento de actividades, así como a nosotros mismos para poder mejorar nuestras técnicas de clasificación, a la vez que nuestra app móvil.

El proyecto comenzó con una investigación inicial sobre varios artículos que trataban el tema del reconocimiento de actividad desde varias perspectivas. Tras haberlos leído nos dimos cuenta que no había un método genérico para reconocer actividades explicado en detalle. Por tanto, con este trabajo, queríamos reflejar todo el proceso realizado para implementar un clasificador de actividades. Tras analizar las aplicaciones en el mercado, observamos que no existe casi ninguna que te permita hacer un clasificador adaptado y personalizado a una persona en concreto, pudiendo añadir actividades propias.

A lo largo de este proyecto, hemos podido ver las dificultades que conlleva un trabajo de estas dimensiones. Hemos tenido que comprender toda la parte relacionada con el aprendizaje automático, así como entender las distintas métricas que mejor se adaptan a cada actividad, las complicaciones que trae conectarse a dispositivos con sensores como pulseras y móviles, así como la orientación de los ejes de estos, que dependiendo de los dispositivos, pueden tener diferente orientación.

Tras todas las complicaciones que hayamos podido tener, hemos podido solucionar cada una de ellas para poder cumplir nuestros objetivos finales del proyecto. En primer lugar, hemos implementado un reconocedor de actividad con una tasa bastante alta de acierto. En segundo lugar, queríamos que nuestro trabajo se reflejara de una forma más visual y pudiera llegar a todo el mundo, por lo tanto, implementamos una aplicación en

Android que te dice que actividad estás realizando y te permite visualizar un historial con el tiempo empleado en cada una de ellas. Para finalizar nuestro proyecto, nuestro deseo era que a través de la app del móvil, el usuario pudiera crear nuevas actividades para crear un clasificador personalizado. Este era nuestro último objetivo y lo logramos.

En definitiva, con este trabajo hemos aprendido a utilizar toda una serie de tecnologías y herramientas que seguro que nos van a ser muy útiles en nuestro futuro y que va a poder ser utilizado por otras personas, ya que se ha creado una app totalmente funcional.

7.2 Trabajo futuro

Las posibles mejoras que se pueden realizar en un futuro son las siguientes:

- **Mayor número de actividades.** El número de actividades actualmente reconocidas en el clasificador genérico es reducido. Convendría añadir un mayor número de actividades por si el usuario no quiere registrarlas, ahorrando así el paso de añadir actividades personalizadas.
- **Retos y objetivos.** Tras realizar las entrevistas a usuarios potenciales de la app, nos dimos cuenta que para ellos era muy importante tener unos objetivos, como, por ejemplo, correr 5 km antes de un día, y que le avise mediante notificaciones de si lo había logrado o no. Por tanto, queremos que a partir de la pestaña calendario el usuario pueda marcarse objetivos.
- **Integración de plano social**. Consideramos que para que nuestra aplicación tenga una mayor repercusión, es necesario que integre un plano social para que los usuarios que lo deseen puedan compartir sus objetivos o retar a otros usuarios.
- **App multiplataforma**. Ahora mismo nuestra aplicación solo funciona en Android. Debido al número elevado de usuarios que tienen dispositivos de Apple sería conveniente migrar la aplicación a un entorno multiplataforma que dé soporte a distintos sistemas operativos.

8 Conclusions and future work

To finish this project, we will analyze all the work done and the possible improvements that can be carried out.

8.1 Conclusions

After finish this project and have analyzed everything gathered in this document, it is time to make the final conclusions, which we hope that serves to help other people that want to deep in the process of activity recognition, as well as ourselves to improve our classification techniques, as well as our mobile app.

The project began with an initial investigation into several articles dealing with the subject of recognition of activity from various perspectives. After reading them we realized that there was no generic method to recognize activities explained in detail. Therefore, with this work, we want to reflect the whole process carried out to implement a classifier of activities. After analyzing the market of applications, we observe that there isn't apps that allows you to make a classifier tailored and personalized to a specific person, and that you can also add your own activities.

Throughout this project, we have been able to see the difficulties involved in work of these dimensions. We have had to understand all the part related to the Machine Learning, as well as to understand the different metrics that best adapt to each activity, the complications that brings to connect devices with sensors like wrist bands and mobiles, as well as the orientation of the axes of these, which depending on the devices, may have a different orientation.

After all the complications that have appeared while the development, we have been able to solve each of them in order to meet our final project objectives. First, we have implemented an activity recognizer with a fairly high rate of success. Secondly, we wanted our work reflected in a more visual way and could reach the whole world, so we implemented an application on Android that tells you what activity you are doing and allows you to visualize an historical with the time spent in each of them.

To finalize our project, our desire was that through the mobile app, the user could create new activities to create a custom classifier. This was our last objective and we succeeded.

In short, with this work we have learned to use a whole series of technologies and tools that are sure to be very useful in our future and that can be used by other people, since a fully functional app has been created.

8.2 Future work

The possible improvements that can be made in the future are the following:

- **Increased number of activities**. The number of activities currently recognized in the generic classifier is small. It would be useful to add a greater number of activities in case the user does not want to register them, thus saving the step of adding personalized activities.
- **Challenges and objectives**. After interviewing potential users of the app, we realized that it was very important for them to have goals, such as running 5 km before a day, and to warn him/her by notifying him/her if he/she had or did not. Therefore, we want that from the calendar tab the user can set goals.
- **Integration of social plan**. We believe that for our application to have a greater impact, it is necessary to integrate a social plan so that users who wish they can to share their goals or challenge other users.
- **Multiplatform App**. Now our app only works on Android. Due to the large number of users who have Apple devices it would be convenient to migrate the application to a cross-platform environment that supports different operating systems.

9 Anexos

9.1 Guía de uso de la aplicación

Figura 9.1: Logo de la aplicación

9.1.1 Instalación de la aplicación

Para poder hacer uso de la aplicación necesitaremos disponer de un teléfono móvil cuyo sistema operativo sea Android, en su versión 5.1 (Lollipop), como mínimo. Para poder instalar la app, necesitaremos el archivo apk correspondiente, el cual lo podemos obtener del siguiente enlace:

https://github.com/TfgReconocimientoPulseras/TrainAppTFG/tree/master/apk

La instalación consiste únicamente en pulsar el archivo .apk y seguir el asistente de instalación de aplicaciones de Android. Una vez tengamos instalada la aplicación, la abrimos y veremos la ventana principal:

Figura 9.2: Vista principal

9.1.2 Utilización del clasificador por defecto

Como podemos observar en la figura anterior, la pantalla principal está formada por dos secciones principales. Por un lado, tenemos la vista del clasificador, que es lo primero que se ve al arrancar la aplicación. Si pulsamos el icono del reloj situado en la parte superior de la pantalla, accederemos al historial de actividades.

Si queremos comenzar a reconocer las actividades que estemos realizando con la configuración por defecto, bastará con colocarse el teléfono móvil en la muñeca de la mano izquierda, de tal modo que la pantalla la podamos observar correctamente. Una vez tengamos colocado el teléfono móvil en la posición adecuada, procederemos a pulsar el botón de “COMENZAR A RECONOCER”. A modo de recordatorio, la aplicación nos mostrará una imagen de cómo debe ir colocado el dispositivo en nuestra muñeca.

Una vez confirmado el recordatorio, el clasificador comenzará a funcionar. Mientras estamos realizando unos movimientos, en la pantalla del dispositivo podremos observar una imagen de las actividades que el clasificador está reconociendo y el nombre de dicha actividad.

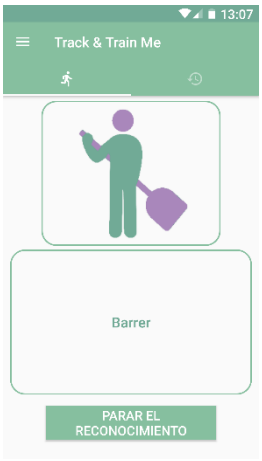


Figura 9.3: Clasificador en funcionamiento

Figura 9.4: Clasificador en funcionamiento

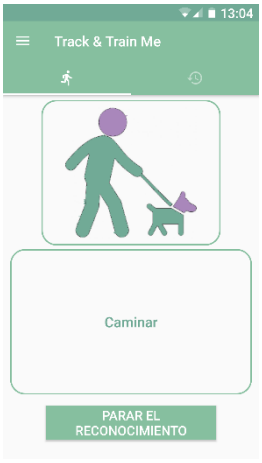


Figura 9.5: Clasificador en funcionamiento

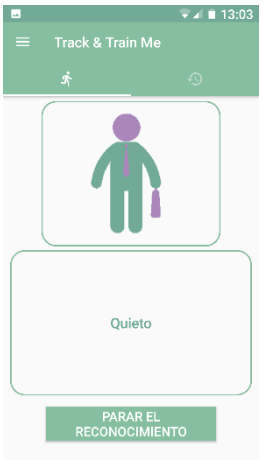


Figura 9.6: Clasificador en funcionamiento

Si durante el proceso realizamos algún movimiento extraño, el clasificador puede no saber que estás haciendo, y lo notificará con su imagen correspondiente.

Figura 9.7: Clasificador en funcionamiento

Si queremos pausar el clasificador, bastará con pulsar el botón “PARAR EL RECONOCIMIENTO”.

9.1.3 Consultar el historial de actividades realizadas

La aplicación almacena un historial de las actividades que se van realizando a lo largo de los días. Si queremos consultar qué actividades se han realizado, por ejemplo, el día 15 de junio de 2017 bastará con ir a la sección del historial (como se comentó anteriormente, el icono del reloj en la parte superior de la vista principal). En esta sección veremos un calendario dinámico, en el cual podemos navegar a través de los meses y los años. Si queremos consultar el día 15 de junio de 2017, deberemos pulsar en las flechas de los laterales que aparecen justo al lado del mes y año en el que se encuentra el calendario hasta llegar a junio de 2017. Una vez estemos en el mes adecuado, deberemos pulsar sobre el día 15.

Figura 9.8: Vista del calendario

Al pulsar sobre el día podremos observar una lista de actividades realizadas, donde por cada actividad se indicará la hora de inicio y la hora final. Podremos navegar por la lista subiéndola o bajándola.

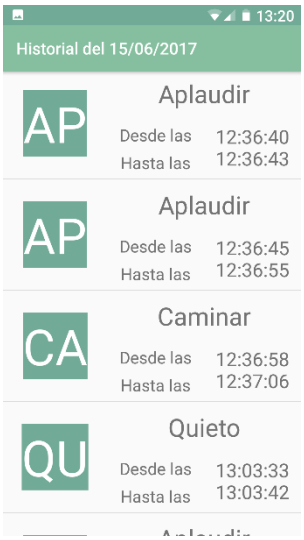


Figura 9.9: Lista del historial de un día concreto

Si pulsamos sobre un día en el que no hemos realizado ninguna actividad, la lista aparecerá vacía.

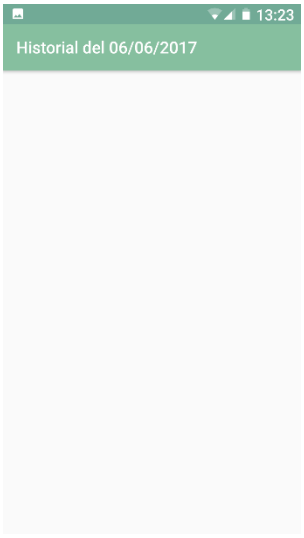


Figura 9.10: Lista del historial de un día concreto vacío

9.1.4 Añadir nuevas actividades a nuestro clasificador

Como ususario de la aplicación, podemos pararnos a pensar que el número de aplicaciones que dispone el sistema es bastante reducido. Por defecto, la aplicación tiene instaladas las actividades “Caminar”, “Barrer”, “Estar quieto” y “Aplaudir”. Si queremos que nuestro clasificador clasifique más actividades deberemos grabarnos realizando la actividad que queramos añadir. Para poder realizar este paso, debemos dirigirnos al Asistente de grabación de actividades. Para ello, desde la vista principal de la aplicación, deberemos pulsar el “menú deslizante” que se encuentra situado en la esquina superior izquierda de la aplicación.

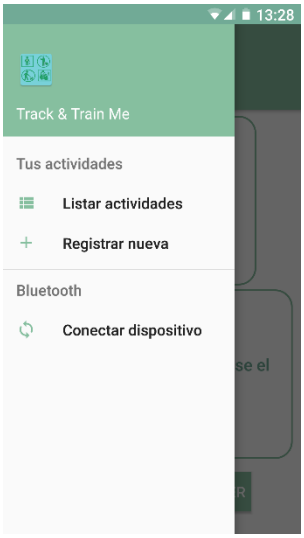


Figura 9.11: Menú hamburguesa

Una vez tengamos desplegado el menú lateral, deberemos pulsar en el botón “Registrar nueva”. A partir de aquí, ya estaremos en el Asistente de grabación de actividades.

Figura 9.12: Vista de bienvenida al asistente de grabación

Tras pulsar el botón de “SIGUIENTE”, nos aparecerá un formulario, obligatorio de rellenar para poder añadir una nueva actividad. En el formulario se nos pregunta que nombre queremos poner a la actividad y que fotografía (campo opcional) queremos que aparezca cuando la nueva actividad sea clasificada por el clasificador. Nada más acceder al formulario se nos solicitará permisos para poder acceder a ficheros.

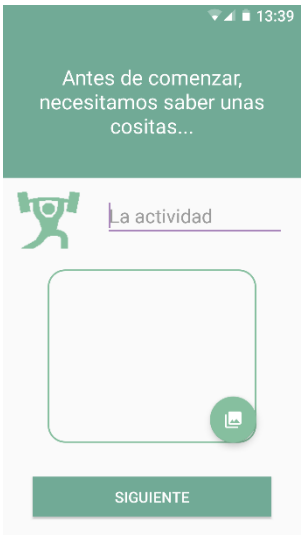


Figura 9.13: Formulario del asistente de grabación

Al pulsar sobre el icono con forma de “galería”, se nos abrirá una ventana donde nos dejará seleccionar una imagen que tengamos. Si no seleccionamos ninguna imagen, la aplicación utilizará una imagen por defecto. Crearemos la activdad “Agitar”.

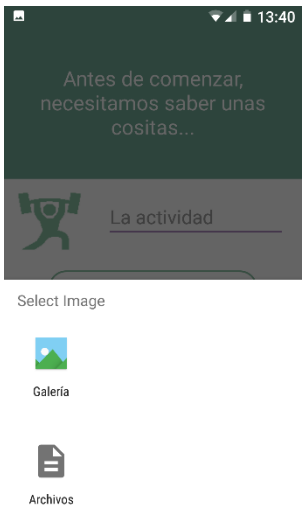


Figura 9.14: Selector de imágenes del asistente de grabación

Tras finalizar el formulario, pulsamos el botón “SIGUIENTE” y accederemos al último paso del asistente. Cuando tengamos colocado el teléfono móvil en la misma posición que cuando hemos utilizado el clasificador, podremos dar al botón de “COMENZAR”. Para que el proceso de grabación sea lo más eficaz posible, solo deberemos de realizar los movimientos correspondientes a la nueva actividad. Cualquier otro movimiento provoca que el clasificador clasifique peor esta nueva actividad, confundiéndola con otras o no sabiendo qué estás realizando.

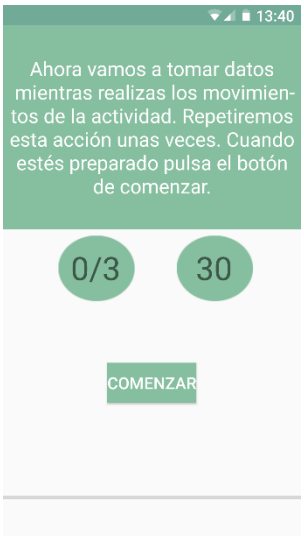


Figura 9.15: Asistente de grabación parado

Tras pulsar el botón de “COMENZAR”, el grabador comenzará a tomar datos de tus movimientos. Durante 30 segundos deberá de ralizar los movimientos de la actividad. Este paso se tiene que repetir tres veces.

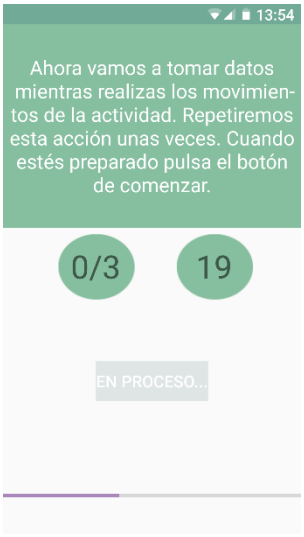


Figura 9.16: Asistente de grabación en funcionamiento

Cuando hayan transcurrido 30 segundos, la aplicación nos avisará a través de una de notificación de que el usuario puede descansar y que tiene que volver a repetir el mismo proceso (así tres veces).

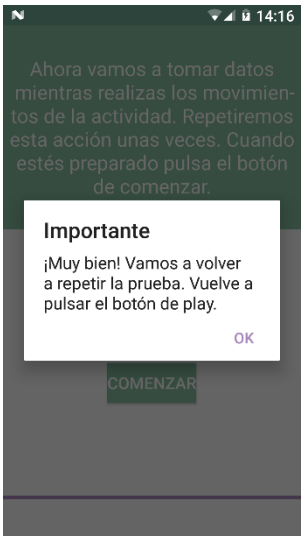


Figura 9.17: Notificación para poder continuar

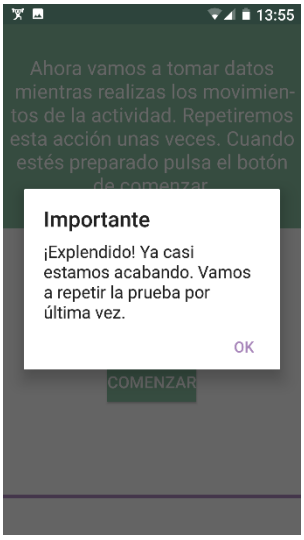


Figura 9.18: Notificación para poder continuar

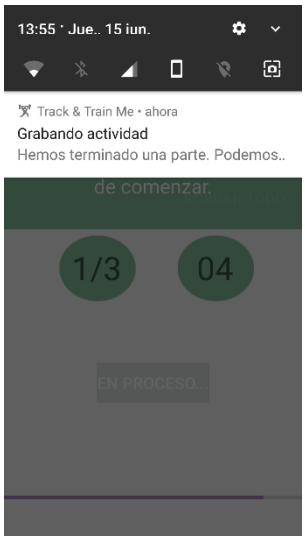


Figura 9.19: Notificación para poder continuar

Cuando se haya realizado este paso tres veces, el asistente enviará los datos recogidos a un servidor para que sean procesados. Mientras se construye un clasificador observará un mensaje de espera.

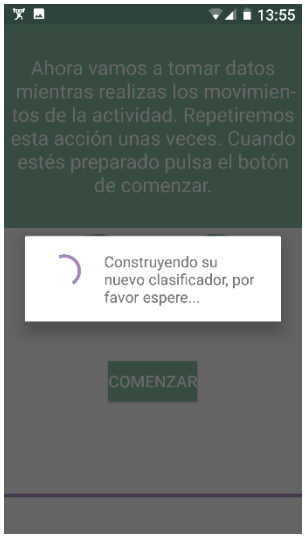


Figura 9.20: Mensaje de construcción en proceso

Cuando el servidor haya terminado de construir su nuevo clasificador, se nos notificará por pantalla el éxito de la operación, preguntándonos si queremos añadir alguna actividad más a la aplicación.

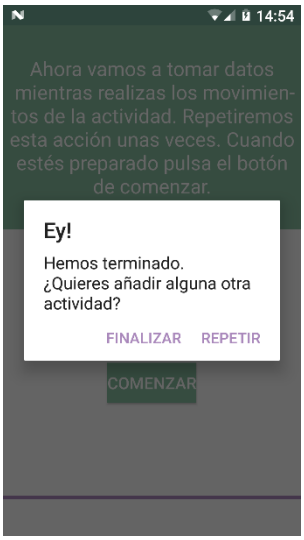


Figura 9.21: Mensaje de finalización con éxito

Si durante este proceso ha habido un error, una notificación nos advertirá del problema y nos preguntará si queremos volver a intentar el proceso.

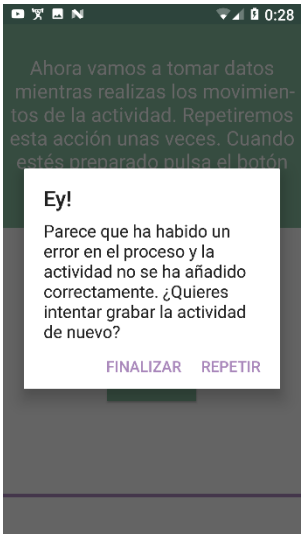


Figura 9.22: Mensaje de finalización con error

Si todo ha ido bien, tendremos una nueva actividad en el sistema. Para comprobar que se ha añadido correctamente, deberemos ir a la vista principal de la aplicación. Desde allí, accederemos al menú principal desde el cual hemos entrado al Asistente de grabación. Dentro del menú, pulsaremos el botón “Listar actividades”. En esta pantalla

visualizaremos las actividades que tiene el sistema registrado y la fecha de su creación. Podemos observar que la actividad nueva “Agitar” se ha creado correctamente.

Figura 9.23: Lista de actividades registradas

9.1.5 Conectarnos con una pulsera vía Bluetooth

Tal vez nos pueda resultar incómodo llevar nuestro teléfono móvil en la muñeca. Ante esta problemática, la aplicación permite utilizar la pulsera SensorTag CC2650, la cual tiene un tamaño más reducido. Para poder conectar la pulsera con la aplicación, deberemos ir al menú deslizante de la vista principal. Una vez lo hayamos abierto, deberemos pulsar en “Conectar dispositivo”. Una vez pulsado, accederemos a la sección encargada de conectar nuestra pulsera con la aplicación. Para comenzar buscar dispositivos a nuestro alcande, deberemos pulsar sobre el icono situado en la esquina superior derecha. Se nos solicitará permisos para poder utilizar el Bluetooth, se los concedemos. Si tenemos el Bluetooth del teléfono apagado, nos preguntará si queremos encenderlo. Una vez esté el Bluetooth preparado, nos aparecerá en la pantalla los dispositivos a nuestro alcance.

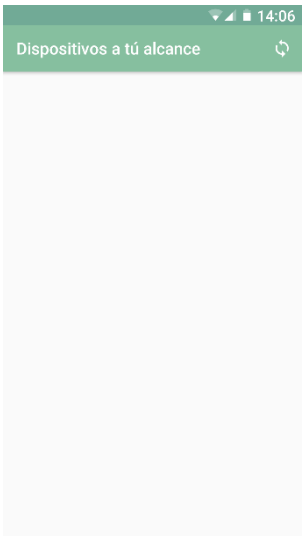


Figura 9.24: Vista del asistente Bluetooth

Pulsamos sobre nuestro dispositivo y la aplicación comenzará el proceso de conexión.

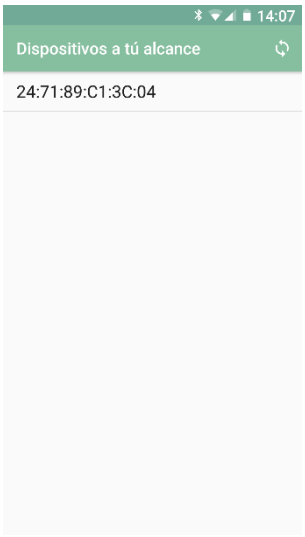


Figura 9.25: Lista de dispositivos a nuestro alcance

Si la conexión ha sido todo un éxito, se nos redirigirá a la vista principal de la aplicación, donde nos aparecerá un mensaje indicándonos que la conexión ha sido todo un éxito. A partir de este punto, podremos usar el Asistente de grabación de actividades

y el clasificador del mismo modo que con el teléfono, recordando en todo momento que ambos dispositivos deben ir siempre bien colocados.

Figura 9.26: Vista principal anunciando que la conexión Bluetooth es correcta

10 Glosario

- **API**: conjunto de subrutinas, funciones y procedimientos que ofrece una biblioteca para ser utilizado en otro software proporcionando una capa de abstracción.
- **AR**: reconocimiento de actividad.
- **ASD**: trastorno del espectro autista. Una modalidad de autismo.
- **AST**: estructura sintáctica abstracta en forma de árbol de un código fuente, donde cada nodo denota una construcción que ocurre en el código fuente.
- **BLE**: nueva tecnología de Bluetooth que permite la conexión inalámbrica de dispositivos móviles y computadoras. Se caracteriza por su bajo consumo.
- **CPU**: hardware encargado de interpretar instrucciones. Es la parte central de toda computadora, ya que su objetivo es procesar todas las funciones.
- **Datos RAW**: datos vírgenes que no han sido procesados. Tal y como se han obtenido de su origen.
- **DFT**: tipo de transformada discreta de Fourier utilizada en el análisis de Fourier.
- **DGO**: metodología del proceso de ingeniería de usabilidad propuesto por Alan Cooper, donde se diseña a partir de los objetivos que el usuario persigue a la hora de utilizar un sistema.
- **DSI**: asignatura del Grado en Ingeniería Informática de la Universidad Complutense de Madrid.
- **FFT**: algoritmo que permite calcular la transformada de Fourier discreta (DFT) y su inversa.
- **GATT**: tipo perfil Bluetooth relacionado con el Bluetooth de baja energía. Específica, pues, una interfaz de alto nivel para su uso entre dispositivos.
- **GPL**: licencia pública general que garantiza a los usuarios finales la libertad de usar, estudiar, compartir y modificar el software.
- **GPS**: sensor que permite determinar la posición de un objeto en toda la Tierra.
- **GSM**: estándar que permite la comunicación de teléfonos móviles por toda Europa.
- **HMM**: modelo oculto de Márkov. Es un modelo estadístico en el que se asume que el sistema a modelar es un proceso de parámetros desconocidos.

- **KNN**: algoritmo de aprendizaje automático usado para la clasificación de objetos basado en un entrenamiento mediante ejemplo cercanos.
- **MAC**: dirección que sirve como identificación única. Es proporcionado por el fabricante.
- **MB**: unidad que sirve para mediar una cantidad de datos informáticos.
- **QDA**: método utilizado en técnicas de aprendizaje automático para separar medidas en dos o más a clase de objetos mediante cuadrados.
- **Random Forest**: técnica de aprendiza automático utilizado para clasificar que utiliza la combinación de árboles predictores, de tal manera que cada árbol depende de los valores de un vector aleatorio probado independientemente y con la misma distribución para cada uno de estos.
- **RFFT**: algoritmo que calcula la transformada discreta de Fourier de una serie de valores reales.
- **RMS de Fourier**: raíz cuadrada de media de cuadrados.
- **SVM**: conjunto de algoritmos de aprendizaje automático que están relacionados con problemas de clasificación y regresión.
- **TFG**: proyecto final de carrera que se realiza al finalizar los estudios de grado.
- **Timestamp**: marca temporal que denota la fecha en la que ocurrió un determinado evento.
- **UUID**: número de 16 bytes utilizado para funcionar como un identificado único universal.
- **Wakelock:** proceso que se ejecuta en segundo plano y que fuerza a la CPU a estar activa.
- **Wearable**: tipo de dispositivo que se incorpora en alguna parte de nuestro cuerpo. Interactúa de manera continua con el usuario y otros dispositivos con el objetivo de realizar alguna función.
- **WSGI:** especificación sencilla e interfaz universal entre los servidores web y las aplicaciones web o frameworks de Python.

11 Bibliografía

[1] «2002IEEE.pdf». [Online]. Disponible en: http://web.sfc.keio.ac.jp/~ohgi/Research/2002IEEE/2002IEEE.pdf. [Accedido: 26-may-2017]

[2] Ozlem Durmaz Incel , Mustafa Kose , Cem Ersoy, «A Review and Taxonomy of Activity Recognition on Mobile phones».

[3] Muhammad Shoaib , Stephan Bosch , Ozlem Durmaz Incel , Hans Scholten y and Paul J.M. Havinga, «A Survey of Online Activity Recognition Using Mobile Phones».

[4] «Sensor», *Sensor*. [Online]. Disponible en: https://es.wikipedia.org/wiki/Sensor

[5] Varun Nagpal, «Android Sensor Programming By Example», en *Android Sensor Programming By Example*, [Online]. Disponible en: http://proquest.safaribooksonline.com/book/programming/android/9781785285509/firstchapter#X2ludGVybmFsX0h0bWxWaWV3P3htbGlkPTk3ODE3ODUyODU1MDklMkZjaDAxbHZsMXNlYzdfaHRtbCZxdWVyeT0=

[6] «El giroscopio y su importancia». [Online]. Disponible en: https://elandroidelibre.elespanol.com/2016/07/giroscopio-movil-android.html

[7] Muhammad Shoaib, Stephan Bosch, Ozlem Durmaz Incel, Hans Scholten, y Paul J. M. Havinga, «Fusion of Smartphone Motion Sensors for Physical Activity Recognition». [Online]. Disponible en: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4118351/

[8] Antonsson E.K y Mann R.W, «The frequency content of gai».

[9] Pekka Siirtola y Juha Röning, «Recognizing Human Activities User independently based on accelerometer data».

[10] Xing Su, Hanghang Tong, Ping Ji, «Activity Recognition with Smartphone Sensors».

[11] «Decision tree learning», *Wikipedia*. 12-jun-2017 [Online]. Disponible en: https://en.wikipedia.org/w/index.php?title=Decision_tree_learning&oldid=785249911

[12] Peter Harrington, *Machine Learning IN ACTION*. .

[13] «sklearn.tree.DecisionTreeClassifier — scikit-learn 0.18.1 documentation». [Online]. Disponible en: http://scikit-

learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier. [Accedido: 23-may-2017]

[14] «Random forest vs Simple tree», *Quantdare*. 07-ene-2015 [Online]. Disponible en: https://quantdare.com/random-forest-vs-simple-tree/. [Accedido: 22-may-2017]

[15] «Alan Cooper». [Online]. Disponible en: https://www.cooper.com/people/alan_cooper. [Accedido: 27-may-2017]

[16] Pablo Moreno Ger, Guillermo Jiménez Díaz, Antonio Sánchez Ruiz-Granados, y Luis Garmendia, «Tema 3: Diseño guiado por objetivos». .

[17] «Joinery». [Online]. Disponible en: https://cardillo.github.io/joinery/v1.8/api/reference/joinery/DataFrame.html. [Accedido: 06-jun-2017]

[18] Android, *Sensors Overview*. [Online]. Disponible en: https://developer.android.com/guide/topics/sensors/sensors_overview.html

[19] Android, *TimerTask*. [Online]. Disponible en: https://developer.android.com/reference/java/util/TimerTask.html

[20] Android, *Timer*. [Online]. Disponible en: https://developer.android.com/reference/java/util/Timer.html].

[21] *Bluetooth Tutorial*. [Online]. Disponible en: https://blog.stylingandroid.com/bluetooth-le-part-1/

[22] (primero), *Android - BluetoothAdapter*. [Online]. Disponible en: https://developer.android.com/reference/android/bluetooth/BluetoothAdapter.html

[23] *Android - BluetoothGatt*. [Online]. Disponible en: https://developer.android.com/reference/android/bluetooth/BluetoothGatt.html

[24] *Android - Bluetooth GattCallback*. [Online]. Disponible en: https://developer.android.com/reference/android/bluetooth/BluetoothGattCallback.html

[25] *QPython*. [Online]. Disponible en: http://qpython.com/

[26] *Jython*. [Online]. Disponible en: http://www.jython.org/

[27] *S4LA*. [Online]. Disponible en: https://www.tutorialspoint.com/sl4a/index.html

[28] *Apache Math*. [Online]. Disponible en: https://commons.apache.org/proper/commons-math/

[29] «BeanShell». [Online]. Disponible en: http://www.beanshell.org/

[30] *AST*. [Online]. Disponible en: https://es.wikipedia.org/wiki/%C3%81rbol_de_sintaxis_abstracta

[31] Kyle Mew, «Android Design Patterns and Best Practice», .

[32] *Bottle*. [Online]. Disponible en: http://bottlepy.org/docs/dev/

[33] *Asynchttp*. [Online]. Disponible en: http://loopj.com/android-async-http/

[34] *Android - Parcelable*. [Online]. Disponible en: https://developer.android.com/reference/android/os/Parcelable.html

[35] *Android - BlockingQueue*. [Online]. Disponible en: https://developer.android.com/reference/java/util/concurrent/BlockingQueue.html

[36] Iván Aguilera Calle y Daniel García Moreno, «Wakelock» [Online]. Disponible en: http://wikis.fdi.ucm.es/ELP/Wakelock

12 Aportaciones

La realización de este proyecto consta de tres partes principales. Por una parte, la elaboración, redacción y revisión de los distintos capítulos y apartados de la memoria, en la que todos los componentes hemos participado. Por otra parte, el desarrollo, implementación y análisis del proceso de Machine Learning para lograr obtener un buen clasificador que realice adecuadamente el proceso de reconocimiento de actividad, y por último, el desarrollo de la aplicación para Android.

12.1 Aportaciones de Iván

En cuanto a mis aportaciones al proyecto, he colaborado, de mayor a menor porcentaje, en el desarrollo de la aplicación Android, en la memoria y por último en el proceso de aprendizaje.

12.1.1 Memoria

En cuanto a la memoria mis aportaciones han sido las siguientes (orden cronológico):

- Participación en la búsqueda de información sobre el estado del arte en artículos científicos, así como la realización de resúmenes traducidos, junto con mis tres compañeros.
- Participación en la creación del esquema de la memoria, junto con mis tres compañeros.
- Redacción de la introducción (castellano) junto con mi compañero Daniel.
- Recopilación de distinta información sobre los componentes de Android para su posterior inclusión en el apartado de la aplicación Android.
- Recopilación de información sobre el funcionamiento de los sensores en Android.
- Recopilación de información sobre los AST y BeanShell.
- Redacción y elaboración del apartado y sub apartados del proceso de diseño de la aplicación Android.
- Redacción junto a mi compañero Daniel de los demás apartados de la aplicación Android.
- Revisión del capítulo de introducción y sus apartados.

- Revisión del capítulo de estado del arte y sus correspondientes sub apartados junto a mi compañera Verónica.
- Revisión del capítulo de implementación del sistema de reconocimiento, junto a mis tres compañeros.
- Revisión del capítulo de la aplicación Android junto a mis compañeros Daniel y Verónica.
- Revisión del capítulo de experimentación con mis compañeros Daniel y Alejandro.
- Revisión del capítulo de conclusiones y trabajo futuro a mis otros tres compañeros.
- Migración de la memoria de Google Docs a Word, junto a mis tres compañeros.
- Revisión de formato de la memoria en Word, junto a mis tres compañeros.
- Elaboración de los distintos gráficos del capítulo de la aplicación Android utilizando la herramiento draw.io.
- Inclusión y revisión de referencias de todos los capítulos de la memoria, utilizando el plugin de Zotero para Word.
- Creación de índices de figuras y tablas, asignando títulos a cada uno de estos elementos.
- Revisión del resumen y el abstract tras el feedback proporcionado por el borrador.
- Revisión del capítulo de introducción (castellano e inglés) tras el feedback proporcionado por el borrador.
- Revisión del capítulo de aplicación Android tras el feedback proporcionado por el borrador.
- Revisión del capítulo de guía de instalación/uso de la aplicación.
- Colaboración en la redacción del apartado de personalización del clasificador del capítulo seis junto con Daniel.

12.1.2 Machine Learning

Respecto a la parte de aprendizaje automático, aunque yo he colaborado en mayor parte en la aplicación Android, si que realizamos un proceso de aprendizaje inicial entre todos los componentes del grupo para tener una primera toma de contacto:

- Participación en el desarrollo de la aplicación web inicial que contamos en el apartado 4.1 para la captura de datos inicial. Cambios en la interfaz y funcionalidad de envío de datos al servidor.
- Realización de scripts en Python para realizar el procesamiento de los datos obtenidos (realizando segmentación de ventanas de un segundo y con un 50% de solapamiento entre ventanas). Esto lo hicimos cada uno por separado para coger soltura en el uso de Python y de Pandas, aunque finalmente alcanzamos una solución entre todos.
- Participación en el desarrollo y adaptación del script de Python para conectarnos y obtener datos de la pulsera SensorTagCC2650, junto a mi compañero Daniel.
- Colaboración en la recogida de datos de distintas actividades mediante la pulsera SensorTagCC2650 para el posterior entrenamiento del clasificador, junto a mis tres compañeros.
- Colaboración en la búsqueda de distintos estudios o proyectos ya realizados, y la obtención de métricas que podrían sernos útiles.

12.1.3 Aplicación Android

Por último, la mayor aportación al proyecto por mi parte ha sido la participación en el desarrollo de la aplicación Android, en cuanto a esfuerzo se refiere. Mis aportaciones, que se han realizado en la inmensa mayoría junto con mi compañero Daniel son las siguientes:

- Búsqueda de libros, tutoriales y material explicativo sobre Android.
- Desarrollo de la interfaz principal de la aplicación.
- Conexión y obtención de datos de los sensores del smartphone.
- Gestión de permisos para Android: archivos, bluetooth…
- Almacenamiento de datos en ficheros locales del teléfono.
- Desarrollo e implementación de la solución para solventar el problema de los retardos entre el acelerómetro y el giroscopio.
- Creación e implementación del servicio que implementa el reconocimiento de actividades de la aplicación: recogida, procesamiento y clasificación.
- Fragmentación del servicio en distintos procesos independientes mediante Threads: proceso de recogida, proceso de prosamiento y proceso de clasificación.

- Diseño, desarrollo e implementación del cuarto proceso o Thread: la política de clasificación.
- Implementación del sistema de comunicación entre los distintos procesos o Threads: BlockingQueues.
- Implementación de la funcionalidad del historial: creación de tablas de bases de datos, diversas consultas y adaptación del código.
- Participación en la búsqueda de información para realizar la conexión a la pulsera SensorTagCC2650.
- Colaboración en la implantación de la conexión de la pulsera al smartphone para obtener datos de la pulsera en vez de los sensores del smartphone.
- Colaboración en la configuración de los distintos parámetros de la pulsera (frecuencia de muestreo, registro de sensores…).
- Participación en la implantación de la funcionalidad de permite añadir nuevas actividades al usuario, creando así un árbol personalizado.
- Búsqueda de información para establecer conexión desde Android a un servidor remoto para enviar los distintos ficheros de datos al servidor mediante petición POST.
- Colaboración en el desarrollo del script del servidor que se encarga de arrancarlo, de recibir datos de la aplicación en forma de ficheros, de entrenar un nuevo clasificador y de devolver un nuevo clasificador a la aplicación.
- Participación en la implementación de la recepción del nuevo clasificador e inclusión en el proceso de clasificación de datos (BeanShell).
- Implementación de subida de imagen para la actividad.
- Revisión del funcionamiento del script del servidor.

12.2 Aportaciones de Daniel

El proyecto podemos estructurarlo en tres partes: el proceso de Machine Learning, que conlleva el procesado de datos, la elaboración de la aplicación Android (muy entrelazada con el proceso de Machine Learning) y por último la realización de la memoria.

En lo referido al proceso de Machine Learning y procesado de datos mis aportaciones no han sido tan elevadas como en otras partes del proyecto, pues los responsables fueron Verónica y Alejandro. Colaboré de la siguiente manera:

- Al igual que mis compañeros, tuve que experimentar con una aplicación web (orientada a dispositivos móviles) la recogida de datos de los sensores.
- Investigación y utilización de la primera pulsera de marca Huawei (junto con su correspondiente aplicación móvil).
- Como el desarrollo de esta parte del proyecto está realizada en Python, realicé diferentes tutoriales para aprender a utilizar este lenguaje de programación, los cuales incluían aspectos básicos, y más posteriormente, el tutorial se dirigía más al aspecto de tratamiento de datos (uso de bibliotecas como Pandas…).
- Una vez comprendido los aspectos básicos de Python, de igual modo que mis compañeros, implementé un Script cuyo objetivo era procesar datos, implementando de manera manual ventadas deslizantes.
- Desarrollo de un Script, junto a mi compañero Iván, que consiguiera conectarse mediante Bluetooth, la pulsera final (Sensor Tag CC2650) y que mostrara por pantalla los valores del acelerómetro y del giroscopio.

En lo referido al proceso de desarrollo de la aplicación Android, aquí se encuentra mi mayor número de aportaciones, pues he participado de manera conjunta con mi compañero Iván. Colaboré de la siguiente manera:

- En primer lugar, desarrollamos Iván y yó el asistente de recogida de datos, teniendo en mente el script de recogida de datos anterior, así como el diseño realizado en el proyecto de la asignatura de DSI. Para ello, tuvimos que buscar información acerca de como interactuar con los sensores del teléfono móvil (clases necesarias, tipos de datos utilizados…).

- Manejo de Timers para poder realizar operaciones durante un tiempo determinado.
- Nos documentamos sobre como almacenar datos en la memoria del teléfono (por ejemplo, los ficheros que contienen los datos de los sensores).
- Colaboración en la implementación del historial del reconocimiento de actividades.
- Una vez implementado de manera correcta el asistente de recogida de datos y habiendo cogido soltura en el uso de los sensores, en conjunto implementamos el clasificador partiendo de la idea del asistente. Para ello, tuvimos que diseñar una máquina de estados para poder implementar una política de clasificación. En folios, tuvimos que imaginarnos varios casos, y apartir de ahí, diseñar la máquina de estados. Posteriormente, traducimos el diseño en un conjunto de clases, de tal manera que su utilización en el clasificador resultura lo más sencillo eficaz posible.
- Búsqueda de información acerca de como conectar un dispositivo Bluetooth de baja energía al teléfono.
- Tras encontrar un tutorial sobre como conectar la pulsera (al principio se practicó con el sensor de humedad) implementamos una clase que nos devolviera los datos de los sensores.
- Implementación de un servicio (Service) que se encargue de todo lo relacionado con la conexión Bluetooth de la pulsera, de tal modo que todo concentrado, encapsulado (esto es, realizar peticiones a un objeto, como puede ser encender, apagar, darme datos…).
- Colaboración en la implementación de la petición al servidor que se encarga de enviar los datos de los sensores y de recibir el nuevo árbol de clasificación para su posterior almacenamiento en la base de datos.
- Colaboración en conjunto de todos los compañeros del grupo en la implementación del script que se ejecuta en el servidor y se encarga de recibir los datos y enviar el nuevo árbol de decisión.

Por último, en la parte referida a la redacción de la memoria, la colaboración ha sido en sincronización con todos los compañeros del proyecto. Entre las aportaciones realizadas en la memoria se encuentran las siguientes:

- Al igual que todos mis compañeros, lectura de diferentes ártículos científicos que trataban el tema del reconocimiento de actividad, con el objetivo de que entendiéramos a que nos estábamos enfrentando y ponernos al día de como se encuentra este tema.
- Colaboración en la redacción del capítulo que trata sobre la aplicación Android.
- Colaboración en la redacción del capítulo introductorio del proyecto.
- Colaboración en la redacción del capítulo que explica el estado del arte.
- Revisión de todos los apartados que forma la memoria, corrigiendo errores y añadiendo contenido en cualquier apartado con el objetivo de precisar más una explicación o hacerla más sencilla de cara al lector.
- Revisión final tras las primeras correcciones del borrador de la memoria, modificando aquellas partes que no estaban expuestas de manera correcta.
- Colaboración en el formateado del documento para que tenga un aspecto lo más profesional posible.
- Redacción de la guía de instalación y uso de la aplicación.
- Colaboración en la redacción del apartado de personalización del clasificador.

12.3 Aportaciones de Verónica

El proyecto comenzó en octubre. Lo primero que teníamos que hacer era entender que era el reconocimiento de actividad y todo lo que se había hecho en este tema. Cada uno de nosotros leyó diferentes artículos científicos para sacar las ideas principales y ponerlas en común. Posteriormente, tuvimos que hacer nuestro primer experimento para recoger datos a través de una aplicación web y del móvil. Probé al igual que mis compañeros que los datos se recogieran correctamente con familiares y amigos.

Tras una primera toma de contacto, tuvimos que hacer un script en Python que nos ayudará a hacer solapamiento entre los datos, al principio nos costó bastante, pero tras investigar las bibliotecas de Python, entendí la manera de hacerlo correctamente a través de dos funciones de Pandas.

A partir de este momento, surgieron dos ramas principales de este proyecto, la parte de crear la aplicación Android y otra relacionada con toda la parte de aprendizaje automático y de hacer las pruebas para encontrar los mejores resultados. De la parte de Android se encargaron Daniel e Iván, mientras que Alejandro y yo nos encargamos de la parte de los datos. Tras obtener los datos a través de la pulsera y haber hecho el solapamiento, teníamos que encontrar las mejores métricas. Me encargué de investigar otros estudios y averiguar qué métricas había utilizado. Probé con casi todas las métricas y comprobé la tasa de acierto de cada una de ellas. También tuve que entender con la ayuda de Javier, que dependiendo de cada actividad se tenían que utilizar unas métricas u otra como esta explicado en el capítulo cuatro.

También colaboré en la parte de automatización de los archivos, creando scripts en Python que permitieran procesar una gran cantidad de archivos de manera automática, sin tener que ir uno a uno. Escribí diferentes informes en los notebooks de ipython que reflejaban los datos usados, las métricas más importantes, así como la exactitud del sistema, matrices de confusión y métricas como recall, accuracy, etc. Todo esto necesario para crear los árboles de decisiones que mejor clasificaran las actividades sin tener una gran cantidad de métricas excesivas que ralentizaran posteriormente el análisis en la aplicación Android.

Por último, con la ayuda de Alejandro, creamos un script en Python con la librería Bottle que iniciara un servidor de manera local y que tras recibir datos nuevos a través del móvil, los procesara y creara un árbol de decisión con las nuevas actividades.

Para finalizar, he participado en casi todos los capítulos de la memoria, centrándome sobre todo en capítulos como el de reconocimiento de actividad o el de experimentación, aunque he colaborado en todos los generales, y he revisado todos los capítulos como el de Android.

En definitiva, en la parte de machine learning he colaborado obteniendo datos, creando script para procesarlos, análisis de documentación, automatización para la gestión de ficheros, análisis de métricas, creación de arboles de decisión y random forest, obtención de resultados, buscando las mejores métricas, creación de matrices de confusión, calcular precisión, sensibilidad y exactitud, creación de notebook con los resultados y análisis de actividades con diferentes métricas.

En la parte de Android, he colaborado en el primer diseño inicial que hicimos en DSI, como en el analisis de otras aplicaciones, objetivos de la aplicación y entrevistas a usuarios

En la parte de memoria, he escrito parte del estado del arte, resumen, abstract, implementación del reconocimiento de actividad, experimentación y conclusiones y trabajo futuro. He colaborado en casi todas las partes del documento, leyendo y revisando todos los capítulos.

Para terminar, agradecer a mis compañeros por el gran esfuerzo que hemos hecho los cuatro, ya que todos hemos colaborado en todas las partes de este trabajo, especializándose cada uno en una, y con el trabajo de cada uno ha sido posible terminarlo de forma satisfactoria.

12.4 Aportaciones de Alejandro

- Al principio, junto con mis compañeros, me encargué de la búsqueda de información sobre el reconcomiendo de actividades. En mi caso, yo nunca había trabajado con nada relacionado con el reconocimiento de actividades. Si había realizado trabajos con el aprendizaje automático y eso me permitía tener una base en la materia, pero aun así al principio tuve que tener una fase de investigación bastante elaborada.
- Después de una primera investigación teórica, junto a mis compañeros tuve que estar probando cómo se realiza la captura de datos. Hicimos la modificación de un script para realizar la recogida de datos.
- Luego llega la segunda investigación en la que participé. Se tuvo que investigar sobre cómo realizar la recogida de los datos con solapamiento. Realicé notebooks con la librería de Python, Pandas.
- También, junto con mi compañera Verónica, estuvimos analizando qué características eran más propensas a utilizarse para el reconocimiento de las actividades que en un principio queríamos incluir.
- Estuve investigando sobre los modelos de predicción, en concreto, los árboles de decisión, que fue el método que decidimos que íbamos a utilizar. En esta parte entra también el estudio de todos los parámetros que estos disponen para poder realizar la implementación más óptima. Para ello realicé un script en Python que iba cambiando parámetros en los árboles hasta encontrar los parámetros que creaban un árbol que devolvía la actividad reconocida con mayor exactitud.
- Para la visualización de forma gráfica cómo afectaban el cambio de los parámetros en el árbol, cree un script que realizase un mapa de calor y mostrase que porcentaje de exactitud devolvía el árbol.
- Como a la par que Verónica y yo íbamos realizando las tareas relacionadas con el aprendizaje automático, nuestros compañeros Iván y Daniel iban creando la app Android, programé un script que dados unos datos de entrenamiento, crease un árbol, ya con los mejores parámetros implementados y que este devolviese un string que representase una serie de “if-else” encadenados y que estos cuando fuesen implementados en la app devolviese que actividad se había reconocido.

- Como se ha podido observar, tiendo bastante a crear scripts que puedan facilitar bastante las tareas. Como nos encontrábamos con la tediosa tarea de que cada vez que obteníamos datos nuevos de alguien debíamos pasar los nuevos ficheros uno por uno, decidí crear un script que nos ayudase en esa tarea y que fuese leyendo todos los ficheros e ir llamando a funciones de otros scripts para el procesado de datos. De esta manera nos ahorrábamos tener que estar pasando fichero por fichero.
- Para finalizar, entre mi compañera Verónica y yo creamos un script que permitiese iniciar un servidor y así poder conectarse a él para enviarle los ficheros nuevos de datos y que este pudiera crear un nuevo árbol reconocedor. Para ello utilizamos la librería de Python, Bottle.
- Para el apartado del estado del arte de la memoria, junto a mis compañeros realice una investigación que nos permitiese realizar el trabajo óptimo que hemos presentado.
- Colaboré en la elaboración del apartado del estado del arte y realicé las revisiones de los apartados de este capítulo en los cuales no fui partícipe en su elaboración.
- También participé en la elaboración del cuarto capítulo de la memoria, implementación del sistema de reconocimiento de actividad.
- Colaboré también en la elaboración del capítulo implementación del clasificador de actividad mediante árboles de decisión.
- Realicé la traducción al inglés de la parte de la memoria que habla de la conclusión del trabajo y del trabajo futuro que nos queda por implementar.
- Como mis compañeros, me he encargado de la revisión de los apartados de la memoria que no me he encargado de escribir.
- Junto con mis compañeros cree los títulos de las tablas y de las figuras para poder crear un índice para estas.
- También me ha tocado, junto a mis compañeros, tener que darle formato a la memoria, para poder darle un aspecto profesional.