UNIVERSIDAD CARLOS III DE MADRID

Escuela Politécnica Superior

Grado en Ingeniería Informática

Continuous authentication based on data from Smart Devices

Autor: Silvia Barbero Rodríguez
Tutor: José María de Fuentes García-Romero de Tejada

ABSTRACT

As technology moves forward to offer the user custom information, based on their activity, their habits, their hobbies… User authentication advances and provides them with ways to verify their identity with data that no one else possess, such as fingerprints, and it is becoming more popular. More devices are entering in the market, wearables, which provide an enhanced user experience by expanding the functionalities offered by a computer or a smartphone, and they are being incorporated to the user identification process.

This thesis presents an approach to user authentication based on their movement, collecting data from an accelerometer placed on a smartwatch, to find out if it is a valid metric to distinguish among users when they are performing day to day activities. The recognition task is relied on artificial intelligence techniques, employing machine learning algorithms to generate a model that recognises a user and the activity that is being carried out, and a graphical user interface is provided so that users can try the system and incorporate new information.

Using Waikato University developed software for machine learning algorithms, the system is developed using Python and Texas Instruments eZ430-Chronos smartwatch, and has been tested on a real environment, where several users were asked to perform different activities while wearing the watch.

KEYWORDS

Sensor, biometric identification, accelerometer, user identification, smartphone, smartwatch.

RESUMEN

Según se mueve la tecnología hacia la personalización de la información basada en la actividad de los usuarios, sus hábitos y hobbies… La identificación de los usuarios avanza para proporcionar formas de verificación particulares para cada uno y que no posee nadie más, como es el caso del reconocimiento mediante la huella dactilar, y cada día gana más aceptación entre los usuarios. Los constantes lanzamientos de nuevos dispositivos *weareables*, que proporcionan una experiencia de usuario mejorada, ofrecen nuevas funcionalidades que extienden las que ya proporcionan los ordenadores o dispositivos móviles, y su uso se está incorporando a la verificación de usuarios.

A lo largo de este trabajo se presenta un nuevo enfoque a la identificación de usuario basado en su movimiento, recolectando datos de un acelerómetro situado en un smartwatch, para averiguar si es una forma válida de diferenciar entre usuarios cuando están realizando actividades comunes del día a día. La tarea de identificación se confía a la inteligencia artificial, utilizando algoritmos de aprendizaje automático para generar modelos que sean capaces de reconocer a un usuario y la actividad que están realizando. Además, se proporciona una interfaz de usuario para que los usuarios puedan probar el sistema y ampliarlo con nuevos datos.

Empleando el software de aprendizaje automático desarrollado por la Universidad de Waikato, el sistema está realizado en Python, usando el smartwatch Texas Instruments eZ460-Chronos, y ha sido probado en un entorno real donde se pidió a distintos usuarios que realizasen varias actividades mientras llevaban puesto el reloj.

PALABRAS CLAVE

Sensor, identificación biométrica, acelerómetro, verificación de usuarios, Smartphone, smartwatch.

ACKNOWLEDGEMENTS

After finally finishing this Bachelor Thesis, there are several people to be thanked for their support throughout these years.

First, to my family, for their support and their attempt to understand endless nights confined with my computer to finish projects, assignments and deliveries, for their help and their encouragement throughout all these years.

To Alberto, for his infinite patience and support, for helping me through the most stressful periods and encouraging me to go on and to aim higher.

To my lab rats: Alberto, Natalia, Pablo and Pablo, for helping me with the project and agreeing to test it before even knowing what it was about and exercising while we were suffering a heat wave, your support has been essential for this project.

To the Fantastic 4, the clowder, who have helped me get through the degree, with their continuous support during all these years, confronting our first year together and all the changes coming with it has shown me how important is to have people like you to support each other. The rest of our degree years, although lighter, have been equally hard and your constant encouragement has been always there even when we have been scattered over different countries. I know I can count on you as you can count on me.

To Chema, my tutor, for helping and guiding me throughout all this project, coping up with my crazy ideas and encouraging me to continue with this field. You have boosted my interest on computer security.

And last but not least, thanks to my haters, for fuelling my thrive to improve, and leave them behind.

TABLE OF CONTENTS

ABSTRACT ..... **I**

KEYWORDS I

RESUMEN ..... **II**

PALABRAS CLAVE II

ACKNOWLEDGEMENTS ..... **III**

INDEX OF TABLES ..... **VI**

INDEX OF FIGURES ..... **VII**

1. INTRODUCTION **1**

1.1. MOTIVATION 1
1.2. GOALS 2
1.3. DOCUMENT'S STRUCTURE 3

2. STATE OF THE ART **4**

2.1. BIOMETRIC AUTHENTICATION 4
2.1.1. Fingerprint *5*
2.1.2. Heart rate *7*
2.1.3. Iris and retina scanner *8*
2.1.4. Accelerometer *10*
2.2. MACHINE LEARNING 11
2.3. PUBLISHED ARTICLES AND SIMILAR WORK 12

3. ANALYSIS **14**

3.1. SCOPE OF WORK 14
3.2. FIELD STUDY 15
3.2.1. Smart devices *15*
3.2.2. Development tools *16*
3.2.2.1. Data treatement and reception 17
3.2.2.2. Machine learning tools 17
3.2.2.2.1. WEKA 17
3.2.2.2.2. MOA 18
3.2.2.2.3. MEKA 18
3.2.2.3. Programming languages 19
3.3. REQUIREMENTS 19
3.3.1. Functional requirements *20*
3.3.2. Non-functional requirements *20*
3.4. HARDWARE AND SOFTWARE RESTRICTIONS 22
3.5. USE CASES 23
3.5.1. Traceability matrix *24*
3.6. TEST CATALOGUE 24
3.6.1. Traceability matrix *26*

4. DESIGN **28**

4.1. SYSTEM ARCHITECTURE 28
4.2. USER INTERFACE MOCK-UPS 29
4.3. COMPONENT DIAGRAMS 30
4.4. CLASS DIAGRAM 32
4.5. SEQUENCE DIAGRAMS 34

5. IMPLEMENTATION AND EVALUATION **36**

5.1. SYSTEM DESIGN ADAPTATION 36
5.2. USER INTERFACE 37

5.3. DATA ACQUISITION 41
5.3.1. Rest *42*
5.3.2. Walk *43*
5.3.3. Run *44*
5.3.4. Jump *45*
5.4. TEST USERS 46
5.5. MOA AND MEKA EXPERIMENTATION 48
5.6. EVALUATION RESULTS 55
5.7. EVALUATION FORM 55
6. MANAGEMENT 59
6.1. PLANNING 59
6.2. BUDGET 61
6.3. COMMERCIALIZATION 62
7. LEGAL 63
8. CONCLUSIONS AND FUTURE WORK 64
REFERENCES 66
ANNEX 68
A. GLOSSARY 68

INDEX OF TABLES

TABLE 2.1: PAPERS COMPARISON 13
TABLE 3.1: FUNCTIONAL REQUIREMENT EXAMPLE TABLE 19
TABLE 3.2: REQUIREMENT RSF-01: USER RECOGNITION 20
TABLE 3.3: REQUIREMENT RFS-02: TASK RECOGNITION 20
TABLE 3.4: REQUIREMENT RFS-03: DATA ACQUISITION 20
TABLE 3.5: REQUIREMENT RFS-04: MONITORIZATION 20
TABLE 3.6: REQUIREMENT RFS-05: DATA IDENTIFICATION 20
TABLE 3.7: REQUIREMENT RSF-06: USER INCORPORATION 20
TABLE 3.8: REQUIREMENT RSNF-01: OPERATING SYSTEM 20
TABLE 3.9: REQUIREMENT RSNF-02: PROGRAMMING LANGUAGE 21
TABLE 3.10: REQUIREMENT RSNF-03: PYTHON LIBRARIES 21
TABLE 3.11: REQUIREMENT RSFN-04: ACCELEROMETER 21
TABLE 3.12: REQUIREMENT RSNF-05: SMART DEVICE 21
TABLE 3.13: REQUIREMENT RSFN-06: PORT CONNECTION 21
TABLE 3.14: REQUIREMENT RSNF-07: SYSTEM PORTS 21
TABLE 3.15: REQUIREMENT RSNF-08: SMARTWATCH DRIVERS 21
TABLE 3.16: REQUIREMENT RSNF-10: DATA TRANSMISSION 22
TABLE 3.17: REQUIREMENT RSNF-10: LEARNING SOFTWARE 22
TABLE 3.18: REQUIREMENT RSNF-11: TRAINING 22
TABLE 3.19: USE CASE EXAMPLE TABLE 23
TABLE 3.20: USE CASE CU-01: ADD USER 23
TABLE 3.21: USE CASE CU-02: RECOGNIZE USER 23
TABLE 3.22: TRACEABILITY MATRIX BETWEEN REQUIREMENTS AND USE CASES 24
TABLE 3.23: TEST EXAMPLE TABLE 25
TABLE 3.24: TEST TS-01: INSTALL SOFTWARE 25
TABLE 3.25: TEST TS-02: DATA RECEPTION 25
TABLE 3.26: TEST TS-03: ADD USER 25
TABLE 3.27: TEST TS-04: CANCEL ADD USER 25
TABLE 3.28: TEST TS-05: RECOGNIZING RESTING USER 25
TABLE 3.29: TEST TS-06: RECOGNIZING WALKING USER 26
TABLE 3.30: TEST TS-07: RECOGNIZING RUNNING USER 26
TABLE 3.31: TEST TS-08: RECOGNIZING JUMPING USER 26
TABLE 3.32: TEST TS-09: CANCEL USER RECOGNITION 26
TABLE 3.33: TEST TS-10: ERROR ON UNKNOWN USER RECOGNITION 26
TABLE 3.34:TRACEABLITY MATRIX BETWEEN REQUIREMENTS AND TESTS 27
TABLE 5.1: USERS PROFILE 46
TABLE 5.2: MEKA EXPERIMENTATION RESULTS 51
TABLE 5.3: BCC J48 EXPERIMENTATION RESULTS 53
TABLE 5.4: BCC HOEFFDING TREE EXPERIMENTATION RESULTS 54
TABLE 5.5: EVALUATION RESULTS 55
TABLE 6.1: DIRECT COST IN PERSONAL 61
TABLE 6.2: DIRECT COST ON EQUIPMENT 61
TABLE 6.3: TOTAL BUDGET FOR THE PROJECT 61

INDEX OF FIGURES

FIGURE 1.1: RELATION BETWEEN USERS ACTIVITIES AND AVAILABLE TECHNOLOGY 2
FIGURE 2.1: HUMAN FINGERPRINT 5
FIGURE 2.2: SMARTPHONE FINGERPRINT SCANNER (SOURCE) 6
FIGURE 2.3: SMARTPHONE HEART RATE MONITOR (SOURCE) 8
FIGURE 2.4: SMARTPHONE IRIS SCANNER (SOURCE) 9
FIGURE 2.5: RETINA'S BLOOD VESSELS (SOURCE) 9
FIGURE 2.6: RETINA SCANNER (SOURCE) 9
FIGURE 2.7: TWO AXIS ACCELEROMETER (SOURCE) 10
FIGURE 2.8: THREE AXIS ACCELEROMETER (SOURCE) 10
FIGURE 3.1: USE CASES DIAGRAM 24
FIGURE 4.1: SYSTEM ARCHITECTURE 28
FIGURE 4.2: USER INTERFACE MOCK-UP 30
FIGURE 4.3: COMPONENT DIAGRAM (I): SYSTEM 31
FIGURE 4.4: COMPONENT DIAGRAM (II): DATA RECEIVER 31
FIGURE 4.5: COMPONENT DIAGRAM (III): MACHINE LEARNING ENGINE 32
FIGURE 4.6: CLASS DIAGRAM 33
FIGURE 4.7: SEQUENCE DIAGRAM (I): ADD USER 35
FIGURE 4.8: SEQUENCE DIAGRAM (II): RECOGNIZE USER 35
FIGURE 5.1: MAIN SCREEN, ACTION SELECTION 37
FIGURE 5.2: ADD USER SCREEN 38
FIGURE 5.3: RECOGNIZE USER SCREEN (I): DATA COLLECTION 38
FIGURE 5.4: RECOGNIZE USER SCREEN (II): SYSTEM OUTPUT 38
FIGURE 5.5: SCREEN FLOW 39
FIGURE 5.6: ACTION FLOW 40
FIGURE 5.7: SABATINI'S COURTYARD 41
FIGURE 5.8: REST DATA FROM USER 1 42
FIGURE 5.9: WALK DATA FROM USER 3 43
FIGURE 5.10: WALK DATA FROM USER 4 44
FIGURE 5.11: RUN DATA FROM USER 5 44
FIGURE 5.12: JUMP DATA FROM USER 6 45
FIGURE 5.13: FORM RESULTS FOR "DO YOU THINK THAT THE ACTIVITIES PROPOSED ARE ENOUGH?" 57
FIGURE 5.14: FORM RESULTS FOR "DO YOU THING THAT THE TIME TO ACQUIRE DATA WAS ENOUGH?" 57
FIGURE 5.15: FORM RESULTS FOR "WOULD YOU PREFER THE USE OF AN ALTERNATIVE DEVICE OTHER THAN A SMARTWATCH TO ACQUIRE ACCELEROMETER DATA?" 57
FIGURE 5.16: FORM RESULTS FOR "DO YOU THINK THAT USER IDENTIFICATION BY ACCELEROMETER DATA IS A VALID ALTERNATIVE TO OTHER FORMS OF BIOMETRIC IDENTIFICATION?" 58
FIGURE 5.17: FORM RESULTS FOR "WOULD YOU USE A SYSTEM THAT VERIFIED USERS BASED ON ACCELEROMETER DATA?" 58
FIGURE 6.1: GANTT DIAGRAM: PROJECT PLANNIFICATION 60

1. INTRODUCTION

This first section is divided in three parts: the first one includes an introduction to the project and the motivation behind it, the second talks about the goals to be achieved and finally, the last part explains the structure that this document will follow.

1.1. MOTIVATION

As smartphones have become more popular so has been the theft over these devices. This has lead companies to develop alternative ways for user authentication to the password or four-digit code to hinder the burglar attempts to access a phone to wipe its data and resell or use it.

The expansion of the market with the release of wearables such as smartwatches or smart bands have contributed to the introduction of different sensor that are aimed to expand the functionality offered by some apps or the different devices.

The idea of being able to identify a person based on a quality that only they possess has been welcomed and it has been introduced on almost all devices with fingerprint sensors. Recently, retina and iris scanners have been included on smartphones, in addition to face recognition system, although this have reportedly failed if they have to distinguish between siblings, resulting in other projects such as “Recognizing Individual Sib in the Case of Siblings with Gait Biometric” (Mohd-Isa, Junaidi, Jahangir, & Chikkanan, 2011), which is a consequence of the impossibility of face recognition between similar siblings.

After studying the different sensors that are available and that can be integrated either on a smartphone or on a wearable device and the different ways that a user can be identified, a research looking for one that would not require any extra activity or action from the user to be able to be recognized was made, with the condition that it could also be used for continuous authentication.

There has been a lot of research done over biometrics and how it can be applied to user authentication, and several alternative systems to the most commonly and widely spread ones have been proposed, some of them will be explored over the State of the art.

This lead to the accelerometer. Based on the idea that people do not move in the same way and actions are performed differently based on how people are taught and their personal background, the fact that they could be distinguished based only on the way they walk, run or simply rest and unlock their phones or access important data without performing an action or entering a code was worth of the investigation.

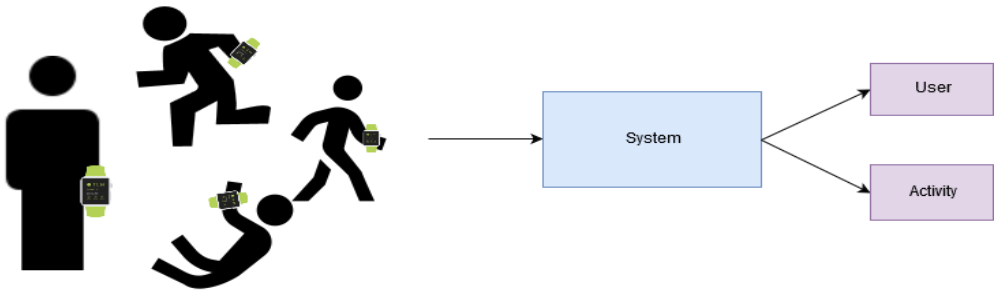

FIGURE 1.1: RELATION BETWEEN USERS ACTIVITIES AND AVAILABLE TECHNOLOGY

1.2. GOALS

The focus collected over the document is to create a system to authenticate people based on their movements independently of the activity being performed using multi-labelling techniques, verifying as well if it can be compared to other forms of biometric identification by obtaining an artificial intelligence model that can achieve a significant range of accuracy.

To achieve this goal, an accelerometer will be used to collect the user movement data and send it over to be processed and used to train an artificial intelligence model. Once trained, the model will be in charge of recognizing data provided by the same user performing the same activity from the training data.

The aim of the project is both to check that an accelerometer can be employed to tell different users apart and to provide a software that is able to so. This means that a research on user identification based on movement is done at the same time a software including this authentication system is being developed. This project will also be used to confirm previous work done over biometric authentication using different users performing different activities, checking that the movement is indeed a unique qualifier of a person and that it is possible to use it on user authentication. This proof of concept is also aimed to be a starting point for future work on biometrics authentication.

1.3. DOCUMENT'S STRUCTURE

This document is divided in several sections which include all the information related with the work, divided per topics that are going to be treated. Shown below, an overall vision of each section is presented.

Over the first section, Introduction, the motivations and goals of the project are stated, as well as the structure that the document will follow.

Section two, State of the art, presents a study of the current state of the technology related with the project.

Section three, Analysis, details the scope of the work, the different requirements that constrain the system and the tests to be performed over the system to ensure that it works according to the requirements.

Section four, Design, displays the system architecture and includes an in-depth study of the pieces which compose it.

Section five, Implementation and evaluation, comprises the most relevant aspects from the implementation process and the evaluation done over the system, including the different tests performed to obtain an artificial intelligence system that is able to correctly tell apart users, data acquisition process and the results obtained after executing the tests proposed on section three.

Section six, Management, goes through the organization and budget of the project.

Section seven, Legal, includes the legal aspects that are related with the project and that affect it directly or indirectly.

Section eight, Conclusions and future work, summarize the work done and the main concepts that can be inferred from it, including future updates that can be done based on it.

Finally, the Annex, a bonus section with additional information relevant for the project, is formed of one chapter devoted to a glossary of useful word used throughout the document.

2. STATE OF THE ART

This section includes all the information obtained after studying the different technologies present and developed that can be useful for the development of the project,

To do so, first some of the most common biometric identification methods are talked through, including fingerprint scanning and heartrate monitorization. The different types of biometric systems and the most common and known ones are defined and explained over this section.

After that, artificial intelligence is defined and the impact it is having on current society, from day to day objects to super computers that use it. This section is centred around one of its disciplines: machine learning algorithms, how they work, why they are useful and why they are the most appropriate way to approach this project.

Finally, similar work and publications will be explored, looking for projects which pursuit a similar approach or present research topics that might be useful for the one presented over this document.

2.1. BIOMETRIC AUTHENTICATION

Nowadays, technology is a key part of most people lives. As it advances, the latest innovations are included with every launch, which results on people turning over this technology more and more.

Having smartphone apps for almost everything, a lot of personal data is being stored on these devices, exposing people if their phones are stolen or accessed without their permission. This has exposed how ordinary locking mechanisms are weak when facing one of these intrusions, and research has been made to achieve locking mechanisms based on user data so that only the owner can access the phone.

This concept is based on biometric information. Biometrics include all metrics related to human qualities of features. Its use to identify people gave place to the concept of biometric authentication, which refers to the use of biometric information to identify individuals.

Some forms of biometric authentication are more effective than others, as they are less error prone or it is more difficult to distort the information sent by the sensor. The key factors to ensure the viability of a biometric recognition system are (Bolle, 1998): universality, uniqueness, permanence, measurability, performance, acceptability and circumvention.

Biometric data can be sorted out on two categories: physiological and behavioural. Physiological data refers to the characteristics a person is born with, such as fingerprints or eye patterns. Behavioural identification methods are still being study, and refers to the ability of uniquely

identifying a person based on their patterns while they perform any activity. This last method is often mentioned as having a lower reliability than physiological biometrics. Some examples of behavioural biometrics are typing rhythm or voice.

Today we can find several biometric authentication systems embedded on smartphones, smartwatches or smart bands. Over the following sections, the most popular and used ones will be explained starting with fingerprints, followed by heart rate monitors, iris and retina scanners and finishing with accelerometers.

2.1.1. FINGERPRINT

A fingerprint is a visible impression left after contact has been made between the friction ridges of a human finger and a surface. They are more perceivable on some materials, for example glass or metal, because of fluids such as sweat or oils produced by the skin. Fingerprints are unique and are developed during the embryo stage.

Figure 2.1 shows the marks left on white paper after applying black ink to the fingertips.

As it is a unique characteristic or property of any human being, it is classified as a physiological biometric.

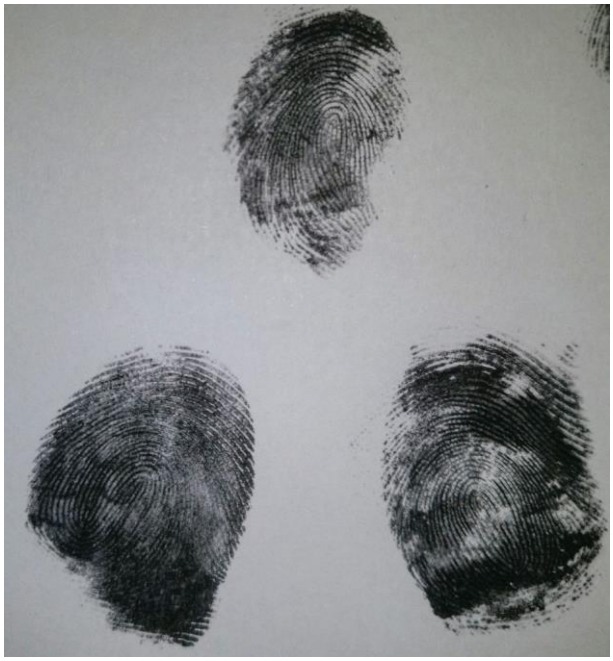

FIGURE 2.1: HUMAN FINGERPRINT

Before being used on mobile devices, the fingerprints began to be used on 1891 (Camgal, 2013), when the fingertips of 23 processed people were taken. Later, on 1894, they were used to verify the identity of 645 jail inmates, which resulted on the police adopting the system. Nowadays, it is used on some countries to register the identity of a person while creating or renewing the national identity document.

Now, its use is generalized on smartphones to verify the owner's identity. Figure 2.2 shows the structure of a fingerprint scanner integrated in a smartphone device. There are several types of fingerprint scanners: optical, capacitive and ultrasonic (Triggs, 2016).

Optical scanners capture an optical image, similarly to how a photograph would be taken, using an algorithm to detect the points that enable user authentication, detecting the fingerprint patterns on its surface. The main drawback of optical scanner is the boundary established by its resolution and the quality of the skin, as it will often produce error if the fingertips are dirty.

Capacitive scanners use capacitor circuits to obtain fingerprint information. The surface of the scanner is composed of conductive plates so that, when the finger is placed on the scanner, the capacitors store the charge transmitted, registering the information about the fingerprints.

Ultrasonic scanners are the most recent addition to fingerprint recognition systems. They are composed of an ultrasonic transmitter and a receiver. When a finger is set upon the scanner, a pulse is transmitted and received by the skin. Some waves are absorbed by the skin and others are reflected. The reflected waves are registered on the receiver. As the reflection of the waves depend on the fingertips ridges forming the fingerprint, they can be recognized using this technology.

FIGURE 2.2: SMARTPHONE FINGERPRINT SCANNER (SOURCE)

2.1.2. HEART RATE

The heart rate is the number of time the heart beats per minute. It is also known as heart pulse (All About Heart Rate (Pulse), 2015). The heart rate varies depending on the person and it is normally between 60 beats per minute and 100 beats per minute.

It is also classified as a physiological biometric, as it is a characteristic we are born with and enables identification.

Its use on user authentication is not as wide spread as fingerprint analysis or other methods such as face or iris recognition, and has been introduced recently on smart devices, mostly on smart bands, as a way to unlock or access critical data on a smartphone.

With the release of the Nymi band, this technology was introduced into the market. Nymi uses an embedded electrocardiogram (ECG) sensor which records user data and store it so that this historical data is use to check if the actual data received from the ECG sensor matches it, in which case the system would unlock.

Samsung Galaxy S5 includes a heart rate monitor embedded on the smartphone, which can be used through an app to consult the heart rate and, on the latest versions, the oxygen level on the blood is also calculated.

The sensor included on this smartphone is an optical device, the most extended way to register a person heart rate monitor. Its process is similar to the optical fingerprint sensor, although instead of registering the creases of the fingertips, blood circulation is monitored. It is also composed a flash and a red LED which calculates the reflectance of the light from the skin.

This method is called pulse oximetry, a non-invasive way to measure how much oxygen is being carried by a person's blood (Bonnie, Suzanne, & Marianna, 2011). This oxygen level is called oxygen saturation level (O_2 or SaO_2) and indicates the percentage of oxygen the blood bears compared to the maximum oxygen that is able to carry.

With every heartbeat, there is a peak on the blood which can be detected by the infrared light, with infers on a new beat of the person using the sensor.

Compared to medical oximeters, the device embedded on mobile phones is really basic, and its situation on the phone would not allow for continuous verification, and the heart rate alone is not a strong enough metric to be able to differentiate among users.

FIGURE 2.3: SMARTPHONE HEART RATE MONITOR (SOURCE)

2.1.3. IRIS AND RETINA SCANNER

The iris is the coloured part of the eye, a membrane with an opening in the centre with variable diameter, depending on how much light needs to pass through, connected to the pupil.

The retina is a photo sensible tissue located on the inner surface of the eye. The images perceived by the eye are projected on it and latter translated into impulses that are sent to the brain through the optic nerve.

Iris and retina scanners have been widely portrayed on films as a maximum-security system, but they are implemented on real life systems.

As both the retina and the iris exhibit unique patterns -the retina with the blood vessels and the iris with the different threads that compose it, they are also classified as physiological biometrics.

The methods used for each are often confused and mixed, but as they are located on different parts of the eye, they need different technology to be able to scan each surface and process the data.

Iris recognition use video cameras with light infrared illumination to obtain the images from the iris with the necessary quality. There are algorithms that look for certain patterns and part of the iris that enable user authentication.

As the technology mentioned is portable, depicted on Figure 2.4, it can be used on smartphones or computers to identify users and is easy to access it.

On the other hand, scanning the retina requires more specific technology, called ocular-based identification technologies. The uniqueness of each retina resides on the pattern that the capillaries in charge of supplying it with blood form on it. The retinal scan is used to identify the patterns formed by the blood vessels. They absorb more light than the surrounding, as can be

seen on Figure 2.5, so when they are inferred with light, it is easy to pick these veins. Similarly to iris recognition, an algorithm is used to identify and recognize the patterns formed by them to be able to identify people.

On Figure 2.6, the machine used to perform a scanner to the retina is shown. The size of the machine does not allow its portability and so, it is not use for small devices or computers.

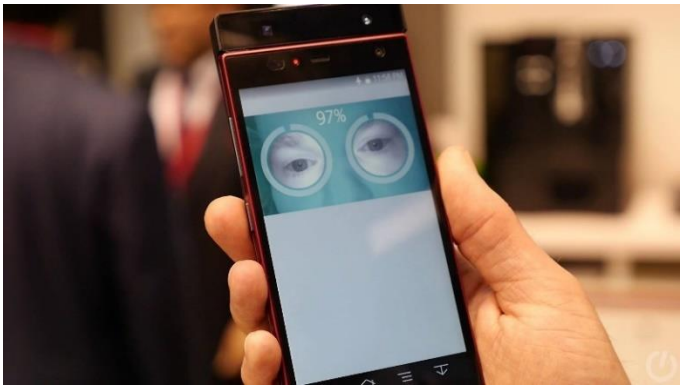

FIGURE 2.4: SMARTPHONE IRIS SCANNER (SOURCE)

FIGURE 2.5: RETINA'S BLOOD VESSELS (SOURCE)

FIGURE 2.6: RETINA SCANNER (SOURCE)

2.1.4. ACCELEROMETER

An accelerometer is a device that measures acceleration forces, the change of velocity, both static and dynamic -such as gravity or movement respectively.

There are three types of accelerometers, depending on the number of axis that it has. One axis accelerometers are the most common type, and are mostly used to calculate the vibration level. Two axis accelerometers record the acceleration or vibration on x and y axis, an example is shown on Figure 2.7. Three axis accelerometers include z axis measurements with respect two axis accelerometers, and are able to take measurements on a 3-dimensional space, a schematic vision of it is shown on Figure 2.8.

There are several ways to measure the acceleration. The most common one is to use the piezoelectric effect: crystal structures are contained which, while experiencing the acceleration forces, get stressed generating voltage.

They are used to track user movements and such, they can be used to identify them based on the patters they unconsciously make while completing a task. As it is not based on characteristics that allow unequivocally user identification, it can be considered as a behavioural biometric.

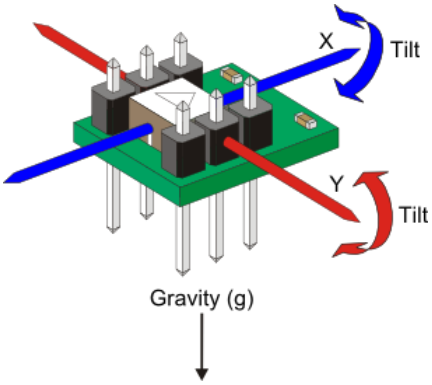

FIGURE 2.7: TWO AXIS ACCELEROMETER (SOURCE)

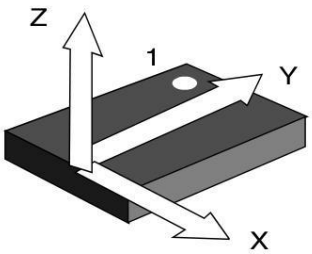

FIGURE 2.8: THREE AXIS ACCELEROMETER (SOURCE)

2.2. MACHINE LEARNING

Artificial intelligence (AI) is the intelligence exhibited by machines. It is a discipline within computer science that deal with the capability of machines to emulate human behaviour and the simulation of intelligence in computers.

Machine learning is a type of Artificial Intelligence in which the means are provided to the computers to endow them with the ability to learn without hardcoding the process. Machine learning is focused on the creation of programs that are able to adapt when they are provided with new data.

The tasks that can be completed by machine learning algorithms can be of regression, if the problem needs to solve estimation or predictions of continuous numerical values; classification, when the system needs to find out the class where an example needs to be sorted for discrete variables and clustering, where the system needs to group data based on the similarity of its attributes.

Considering these tasks, there are different ways to approach the learning process depending on the problem. The most common include supervised, unsupervised and semi-supervised learning and reinforcement learning.

Supervised learning systems use a set of labelled data, called training set, which consist on a series of examples for which the class is known and specified on the input data for the system. Supervised algorithms create models based on the desired output, the specified class, that are able to generalize, for which different methods are introduced to avoid overfitting and such problems derived from the lack of generalization ability.

Overfitting arises when a machine learning algorithms is over trained with data for which the desired output is known. This happens when the system is trained with data that might be filled with noise or which include outliers, which results on the adaptation of the algorithm to be able to fit these examples into it. The goal of the machine learning algorithm is to be able to predict the result of any of the examples, generalizing to be able to sort new data and solve situation where the inputs diverts from the training set examples.

Unsupervised learning systems use unlabelled data, that is, examples for which the class is unknown, to generate the different models.

Semi-supervised learning system combines supervised and unsupervised methods, using both labelled and unlabelled examples to generate the machine learning model.

Reinforcement learning systems introduce the notion of a reward, where the system is either "congratulated" or "punished" depending on the result of an action. They are inspired by psychology and are often used to develop agents.

On an environment where a sensor will be transmitting data to a computer or a portable device so that a system is able to tell, based on said data, the user to whom the data belongs and the activity that is being performed, machine learning algorithms seem to fit the nature of the problem, in particular, it can be sorted into a classification problem.

The sensor data corresponds to the attributes being processed by either of these techniques, with the addition that there would be two different classes: one to sort the data by its user and the other to categorize it based on the activity.

2.3. PUBLISHED ARTICLES AND SIMILAR WORK

Over this section a series of works that have used some form of biometric identification are discussed. First, a summary of all the publications that are used is presented, followed by Table 2.1, where different aspects of them are stated to obtain a global view and compare them. The table includes information about data it identifies, either a user, the activity being performed or both; the algorithm used to perform this recognition, the best accuracy achieved by the system and the number of users that have tested or used the system.

The first proposed paper is "Ph.D. Activity-based Implicit Authentication for Wearable Devices" (Yunze, 2016), in which different smart devices are placed on several parts of the body and user information is recorded while they perform mundane activities.

The second proposed paper is "Authentication Using Pulse-Response Biometrics" (Rasmussen, Marc, Ivan, & Gene, 2014). A pulse system is developed to recognize users who are using an ATM, where a mild electric shock is transmitted to the non-dominant hand, placed on the metallic surface of the ATM, and the remains of the signal are collected from the other hand while it is using the keypad, measuring the variation on the intensity.

A system that has integrated accelerometer authentication is "MotionAuth: Motion-based Authentication for Wrist Worn Smart Devices" (Junshuang Yang, 2015), where users can unlock a device after performing a predefined action, for example, motioning a circle.

On the article "Touch me once and I know it's you! Implicit Authentication based on Touch Screen Patterns" (Alexander, Alina, Frederik, Christian, & Heinrich, 2012) they approach user authentication based on the patterns made by a person while using a smartphone.

Over “Unobtrusive user-authentication on mobile phones using biometric gait recognition” (Derawi, Claudia, & Patrick, 2010) a similar system to the one proposed over this document is researched, where user authentication is done by gait recognition through floor sensor or accelerometers.

The last system that has been researched is “Image-based Ear Biometric Smartphone App for Patient Identification in Field Settings” (Bargal, y otros, 2015), where an application is developed to identify users based on ear patterns using a smartphone camera.

On Table 2.1 several systems that follow the same line of research as the one proposed on this document are stated, exploring different ways of user authentication based on different biometrics. Although movement or activities are employed or used to recognize a user, none of them use it as a way of identification or are centred around recognizing the different activities that can be done through the day.

Reference	**ID**	**Activity**	**Algorithm**	**Best accuracy**	**Users**
(Yunze, 2016)	Yes	No	Random Forest	80.9%	30
(Rasmussen, Marc, Ivan, & Gene, 2014)	Yes	No	Support Vector Machine	100%	15
(Junshuang Yang, 2015)	No	No	Dynamic time warping, Hidden Markov Moddels	N/A	30
(Alexander, Alina, Frederik, Christian, & Heinrich, 2012)	Yes	No	Dynamic time warping	52%	48
(Derawi, Claudia, & Patrick, 2010)	Yes	No	Dynamic time warping	N/A	51
(Bargal, y otros, 2015)	Yes	No	k-NN	100%	84
Current research	Yes	Yes	J48 Classification tree	91% for user and 77% for activity	7

TABLE 2.1: PAPERS COMPARISON

3. ANALYSIS

This section will encompass all tasks related to the analysis of the proposed system. To do so, the scope of the work will be presented first, which will describe the main tasks and functions to be completed by the system. It is followed by a field study of the technology that might be used to complete the goals proposed on the first section. Then, requirements will be talked through followed by hardware and software restrictions, to set the boundaries and constraints of the system. Use cases and its traceability matrix with requirements will be discussed to relate the system requirements with users' possible actions.

Finally, the tests to be performed on the system once it is completed to ensure that the requirements are met are included after the use cases, with a traceability matrix that match them with the system requirements.

3.1. SCOPE OF WORK

Over this section, the main functionalities and responsibilities of the system will be explained, as well as its target.

As shown in Figure 4.1, the system is composed of four main pieces, those being a smart device, a computer, a data base and a machine learning unit. Those pieces will work together to recognize some user and or task based on data obtained from an accelerometer.

To accomplish the established goal, the smart device wirelessly sends accelerometer data measurements to the RF Access Point, and a program developed on the computer accesses said port to extract the received data.

These measurements are then used to create and maintain updated a machine learning model, based on online analysis so that new data can be added to the model on real time. As the amount of data that will be received by the system is constant and can reach a large size, Online Analysis algorithms are contemplated, as they can process large amounts of data and produce a model to predict the output. After a learning period, depending on the activity, its output, the model can be implemented on the desired system and it will update itself based on the new data received.

Thus, the system must fulfil the following points:

- The developed system must obtain and process data from an accelerometer.
- A machine learning model must be obtained to decide whether received data belongs to a certain user or not, independently of the activity that is being performed by the user.
- The machine learning model must be able to update itself based on accelerometer data.
- A machine learning model must be obtained to differentiate between different activities performed by the same user, being resting, walking, running and jumping the chosen for this project.

3.2. FIELD STUDY

The system is composed of several independent pieces connected to work together and can recognize a user and the task being performed. These components are: a smart device, a series of scripts and a machine learning engine that oversees processing new data and get a user based on a predefined algorithm, already tested to obtained the best results after a training period.

To select the best fit option for each of them, research was made beforehand to choose, from the wide variety of available tools, the ones that accommodated better to the problem. Thus, research was done over the different devices that could transmit accelerometer data, the open source machine learning engines and the different programming languages and what each of them provide that could prove useful to the proposed system.

3.2.1. SMART DEVICES

Referring to those electronic devices usually connected to other devices using some form of wireless protocol, such as Bluetooth, NFC or Wi-Fi, smart devices have become increasingly popular due to the applications they provide to increase the service offered by mobile phones or tablets and their portability.

Smartwatches are one of the many smart devices available. They include sensors and functions that mobile phones do not usually include, like pulsometers or accelerometers, which, joined with an application, can implement functionalities as step counter or heart rate monitor. The advantage of a smartwatch over other smart devices is its easy acquisition and the ability to wear it without it interfering with one's daily life.

There is a wide range of smartwatches available to purchase, each offering a set of functions and including a series of sensors. To be able to distinguish one user from another, an accelerometer was chosen as a requirement, as monitoring the movement of a person does not

affect the way a task is completed and people can be told apart from the way they move, walk or run.

After assessing different smartwatches and their features, the Texas Instruments eZ430-Chronos was chosen. It is designed to enable developers to create and test applications for the clock, including a set of sensors that provides user data such as a three-axis accelerometer, pressure sensor, temperature sensor and a battery/voltage sensor. Texas Instruments provides references and guides for the development of applications on it, as well as several examples, and a software to connect the watch to a computer (Control Center Software). The transmission of information is done over <1GHz of wireless transceiver integrated on the system, supplying an USB access point that, once connected to a computer with the correct Texas Instruments drivers installed, can receive the data transmitted from the watch.

There are several projects that have been developed for the Texas Instruments clock, including systems to unlock doors based on accelerometer data and Arduino receivers. Among this projects, there is one that stands out because of its relevance for the development of this project: A Python library that connects to the USB Access Point and opens ACC (accelerometer) communication between the Access Point and the smartwatch. User rlabs published on GitHub the library based on the research done over the Texas Instruments Community (Source), where the connexions done over the watch were monitored to extract the instructions that were exchange between the Control Center Software and the watch.

This library provides a way of communicating with the computer directly to the clock, access accelerometer data and develop a system based on it, which was the goal of the project and thus, the Texas Instruments smartwatch was chosen.

3.2.2. Development tools

To create an intelligent system that can recognize people based on biometric data, artificial intelligence systems that could perform this discrimination were looked through. It also raised the question about the communication between all components, connecting the data transmitted by the smartwatch and its use on the system.

Machine learning algorithms provide mechanisms to classify and cluster data, turning them into the most appropriate artificial intelligence algorithms for the system. There are several machine learning tools that implement different algorithms and functions to be used.

There are also several programming languages that are available and can integrate the different pieces together and each of them offering different tools and IDEs to use.

On this section, these points will be explored.

3.2.2.1. DATA TREATEMENT AND RECEPTION

Using Texas Instruments smartwatch implies the need of the Access point to communicate between the computer and the watch. As mentioned before, there are already several projects developed over the watch which provide useful information, but being rlabs' Python library the most relevant one for this project, data exchange is done using his software.

The library is composed of several functions that oversee opening and closing communication between the watch and the computer and almost all of them are used on the system.

The two main functions are in charge of opening and closing the port, thus allowing further communication once the protocol is initiated on both ends of the system. To open the RF Access Point to ACC communication, there is a chain of hexadecimal instructions that need to be issued to the correct registers. A third function allows to access the accelerometer data sent from the watch and the last one from the library reads the data transmitted and returns it on a vector, each position storing the data of one of the axis.

3.2.2.2. MACHINE LEARNING TOOLS

The University of Waikato has a machine learning dedicated group who, throughout the years, have developed open source software to be used on artificial intelligence problems, including all major machine learning approaches, such as classification, regression and clustering. They provide a GUI to use the software, but it can also be used from the command line or called from code, as it is written in Java and the source code is available on GitHub on public repositories, as well as the extensions provided by users after forking the project.

3.2.2.2.1. WEKA

Issued under the GNU General Public License, Weka is the first machine learning oriented software developed on Waikato University. It includes a wide collection of machine learning algorithms that can also be applied to big data analysis and data mining.

Weka is also the base for the software developed afterwards, its algorithms and source code included on those projects. There are several courses online from Waikato University and tutorials on YouTube to learn how to properly use this tool.

It includes several modes to be able to use classification, clustering and association. Each algorithm comes with a series of parameters that can be modified to achieve the best fitted algorithm for the problem. Once a model is obtained, it can be stored and evaluated with different data.

Weka was then extended to be able to use it with online learning algorithms and with more than one class, evolving to MOA and MEKA.

3.2.2.2.2. MOA

MOA is another machine learning oriented tool developed by the Waikato University team. Based and including the algorithms provided by WEKA, MOA's main objective is to introduce online analysis for stream data, enhancing the original tool by making it more suitable for massive data processing.

It includes algorithms for both classification and clustering that enable to be updated as the data stream is received. As in WEKA, the obtained models can be stored and be used with new data, updating as well the model to include the new information obtained from the stream. It can also be analysed using the provided mode by the GUI, getting accuracy and error information about its performance. It also includes some algorithms from MEKA, allowing multi-target and multi-label classification.

3.2.2.2.3. MEKA

Coming from WEKA as well, MEKA includes all WEKA classifiers, introduces some MOA updatable classifiers and incorporates MULAN framework. This version enables multi-target and multi-label classification and its evaluation using different techniques such as cross-validation or supplied test set.

A problem can be defined as multi-target when there are more than one class and each class can take more than two values, that is, the classes are non-binary. If the classes are binary, then the problem belongs to the multi-label scope, each label indicating if the example is relevant or irrelevant, taking the value 1 or 0 in each case.

Ultimately, this was the chosen option to develop the system as we count with two classes which have a wide range of options, both users and tasks can be added as the software is used. There can be more than two users and tasks, so the problem is framed on the multi-target scope. Moreover, as it includes WEKA and MOA algorithms, the system can be tested using different models and classifiers to choose the one that obtains the best results.

Multi-target algorithms mostly follow the same structure: for each of the classes, a classification algorithm is used individually for each of the classes and all the models obtained are then chained using different techniques, such as classifier chains, Bayes classifier chains, nearest set replacement…

3.2.2.3. PROGRAMMING LANGUAGES

To develop the software connecting data reception and the implementation of the learning algorithms, the options were reduced to two programming languages: Java and Python.

As MEKA is developed in Java, this was the first language considered for the system. It would allow the integration of the source code on the system, simplifying function calls to obtain the desired model and its evaluation and further implementation as the chosen result.

The other option that was considered was Python. On section 3.2.2.1 a Python library that enables the connection between Texas Instruments smartwatch and the computer is mentioned. This connection needs to be done independently from the chosen language. As Python also provides several libraries to pass instructions to the command line and MEKA can be used both from the source code or from the command line, Python was the language chosen, as one of the needed pieces for the system was already implemented.

3.3. REQUIREMENTS

Over this section the different system requirements will be exposed, considering the analysis done on the previous section, the technology to be used and the environment of the system. These requirements will include all restrictions and necessities to have the system running.

This catalogue will include both functional and non-functional requirements, the first referring to any requirement related to what the system is going to do and the second to how the system carries out some task.

They will be presented in tables that will have the following fields:

ID: unique identifier for each requirement. They will use the format RF-YY for functional requirements and RNF-YY for non-functional requirements, where YY stands for a two-digit number starting in 01.

Title: brief description of the requirement.

Description: detailed explanation of the requirement.

Priority: refers to the importance of the requirement. It will be high, medium or low following its relevance for the system.

ID	*RSF-00*
Title	Example table
Description	
Priority	

TABLE 3.1: FUNCTIONAL REQUIREMENT EXAMPLE TABLE

3.3.1. FUNCTIONAL REQUIREMENTS

ID	*RSF-01*
Title	User recognition
Description	The system recognizes a user and the activity he/she is performing based on accelerometer data obtained from a smartwatch
Priority	High

TABLE 3.2: REQUIREMENT RSF-01: USER RECOGNITION

ID	*RSF-02*
Title	Task recognition
Description	The system must recognize a user independently from the activity that is being performed
Priority	High

TABLE 3.3: REQUIREMENT RFS-02: TASK RECOGNITION

ID	*RSF-03*
Title	Data acquisition
Description	The system collects data from an accelerometer placed on a smartwatch
Priority	High

TABLE 3.4: REQUIREMENT RFS-03: DATA ACQUISITION

ID	*RSF-04*
Title	Monitorization
Description	A computer receives the accelerometer data transmitted from the smartwatch
Priority	High

TABLE 3.5: REQUIREMENT RFS-04: MONITORIZATION

ID	*RSF-05*
Title	Data identification
Description	The system obtains a machine learning model to recognize users and activities
Priority	High

TABLE 3.6: REQUIREMENT RFS-05: DATA IDENTIFICATION

ID	*RSF-06*
Title	User incorporation
Description	The user can incorporate more users to the system to be recognized by the machine learning engine
Priority	**High**

TABLE 3.7: REQUIREMENT RSF-06: USER INCORPORATION

3.3.2. NON-FUNCTIONAL REQUIREMENTS

ID	*RSNF-01*
Title	Operating system
Description	The operating system that must be used to develop and execute the system must be Windows or Ubuntu
Priority	High

TABLE 3.8: REQUIREMENT RSNF-01: OPERATING SYSTEM

ID	*RSNF-02*
Title	Programming language
Description	The system must be developed using Python 2.7
Priority	High

TABLE 3.9: REQUIREMENT RSNF-02: PROGRAMMING LANGUAGE

ID	*RSNF-03*
Title	Python libraries
Description	Python uses the libraries pyserial and chronoslib
Priority	High

TABLE 3.10: REQUIREMENT RSNF-03: PYTHON LIBRARIES

ID	*RSNF-04*
Title	Accelerometer
Description	The data is obtained from the user by a three-axis accelerometer sensor placed on a smartwatch
Priority	High

TABLE 3.11: REQUIREMENT RSFN-04: ACCELEROMETER

ID	*RSNF-05*
Title	Smart device
Description	The system uses Texas Instruments eZ430-Chronos to obtain the accelerometer data from the user
Priority	High

TABLE 3.12: REQUIREMENT RSNF-05: SMART DEVICE

ID	*RSNF-06*
Title	Port connection
Description	RF Access Point collects accelerometer data on a computer sent from the smartwatch
Priority	High

TABLE 3.13: REQUIREMENT RSFN-06: PORT CONNECTION

ID	*RSNF-07*
Title	System ports
Description	The program uses the port COM3 on windows and /dev/ttyACM0 on Linux to access the data collected by RF Access Point
Priority	High

TABLE 3.14: REQUIREMENT RSNF-07: SYSTEM PORTS

ID	*RSNF-08*
Title	Smartwatch drivers
Description	The computer in which the system works must have installed Texas Instruments drivers to control the RF Access Point
Priority	High

TABLE 3.15: REQUIREMENT RSNF-08: SMARTWATCH DRIVERS

ID	*RSNF-09*
Title	Data transmission
Description	The Texas Instruments watch must have ACC mode on to transmit the accelerometer data to the RF Access Point
Priority	High

TABLE 3.16: REQUIREMENT RSNF-10: DATA TRANSMISSION

ID	*RSNF-10*
Title	Learning software
Description	The learning process and user recognition is done using MEKA software
Priority	High

TABLE 3.17: REQUIREMENT RSNF-10: LEARNING SOFTWARE

ID	*RSNF-11*
Title	Training
Description	The system has a different training period for each task: ten minutes for resting and to for walking, running and jumping.
Priority	High

TABLE 3.18: REQUIREMENT RSNF-11: TRAINING

3.4. HARDWARE AND SOFTWARE RESTRICTIONS

Besides the constraints stated during the requirements specifications, there are additional hardware and software restrictions that need to be taken into account as well.

Starting with hardware, the system needs to consider that, although Texas Instruments ez430-Chronos counts with different sensors (accelerometer, temperature, altitude), they cannot transmit data simultaneously. This means that only one sensor can be at a time. The smartwatch is not rechargeable, it uses a button cell to power it, which can only transmit accelerometer data for two days before needing a battery change (Instruments, 2015), page 61. Moreover, the data transmitted from the watch can only be received on a computer using the RF Access Point that comes with it. Accelerometer data can only be transmitted wirelessly to the Access Point and has up to 100 meters reach in free field (Instruments, 2015), page 124, and it only provides values from 0 to 255.

The software constraints imposed on the system are less than the hardware restrictions, although as important. The first is that, in order to be able to use the Access Point, Texas Instruments drivers must be installed first. Without them, the computer cannot access to the port. The second is that the exchange of information between the computer and the Access Point must be done by issuing the correct instructions to the Access Point registers.

3.5. USE CASES

To model the flow of events happening on the system and the main functionalities that it displays, use cases will be used.

Use cases are a software development tool that models the interactions between the different actors and the system, describing the steps that are performed given an action and an alternative to cover other alternatives such as errors.

Each use case is structured in tables with the following fields:

ID: unique identifier for each requirement. They will use the format CU-XX, where XX stands for a two-digit number starting in 01.

Actor: user that interacts with the system

Preconditions: trigger that gives way to the use case.

Main flow: development of the use case.

Alternative flow: sequence of steps to follow in case of an unexpected step or an error.

ID	*CU-00*
Title	Use case example table
Actor	
Preconditions	
Main flow	

TABLE 3.19: USE CASE EXAMPLE TABLE

ID	*CU-01*
Title	Add user
Actor	User
Preconditions	The user wants to be recognized by the system
Main flow	The user accesses to the computer software Selects the option to register The user starts the accelerometer data transmission on the smart device The user waits for the software to store the data and incorporate it to the machine learning algorithm

TABLE 3.20: USE CASE CU-01: ADD USER

ID	*CU-02*
Title	Recognize user
Actor	User
Preconditions	The user wants to be recognized by the system and is already registered on it
Main flow	The user starts the computer software Selects the option to be recognized by the system Starts the smart device accelerometer and waits for one minute The software recognizes the user

TABLE 3.21: USE CASE CU-02: RECOGNIZE USER

The use cases diagram is shown in Figure 3.1.

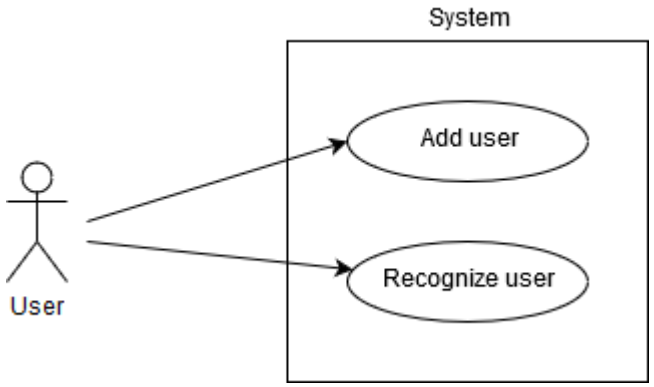

FIGURE 3.1: USE CASES DIAGRAM

3.5.1. TRACEABILITY MATRIX

The following table collects the relationship between the use cases and the functional requirements presented on section 3.3.1.

		Use cases	
		01	**02**
Requirements	**01**		X
	02		X
	03	X	
	04	X	X
	05	X	X
	06	X	

TABLE 3.22: TRACEABILITY MATRIX BETWEEN REQUIREMENTS AND USE CASES

3.6. TEST CATALOGUE

To ensure that the system works correctly and according to the requirements and goals explained, a battery of tests is presented.

In order to check the installation of the system, a laptop was used to reproduce the process and to check that all functions worked correctly. The laptop and the computer in which the project was developed differed on their configuration, so the laptop is used as the minimum configuration needed to be able to run the system.

They will be arranged on tables which contains the following information:

ID: unique identifier for each test. They will use the format TS-XX, where XX stands for a two-digit number starting in 01.

Title: brief description of the test.

Description: detailed explanation of the test.

Output: expected result from the system.

Result: indicates if the result of the test was successful and any additional information.

ID	*TS-00*
Title	Example table
Description	
Output	
Result	

TABLE 3.23: TEST EXAMPLE TABLE

ID	*TS-01*
Title	Install software
Description	The project is installed on a computer, including the libraries and drivers mentioned on the requirements
Output	The software installation is finished without errors
Result	

TABLE 3.24: TEST TS-01: INSTALL SOFTWARE

ID	*TS-02*
Title	Data reception
Description	A user opens the Control Center software provided by Texas Instruments to check that accelerometer data is received on the computer
Output	Data from each axis of the accelerometer is represented graphically on the Control Center
Result	

TABLE 3.25: TEST TS-02: DATA RECEPTION

ID	*TS-03*
Title	Add user
Description	Once the software is installed, a user accesses the software to be included on the system selecting the option “Add user” and performing the indicated activities
Output	The software indicates that the user has been successfully added
Result	

TABLE 3.26: TEST TS-03: ADD USER

ID	*TS-04*
Title	Cancel add user
Description	A user accesses to the software and selects the option “Add user” to be added to the system and selects the option “Cancel” to return to the main screen
Output	The software cancels the addition of the user and returns to the main screen
Result	

TABLE 3.27: TEST TS-04: CANCEL ADD USER

ID	*TS-05*
Title	Recognizing resting user
Description	Once the user is registered on the system, he accesses to the software, selects the option “Recognize user” and rests while the system collects data to identify the user
Output	The software is able to identify the resting user
Result	

TABLE 3.28: TEST TS-05: RECOGNIZING RESTING USER

ID	*TS-06*
Title	Recognizing walking user
Description	Once the user is registered on the system, he accesses to the software, selects the option “Recognize user” and walks while the system collects data to identify the user
Output Result	The software is able to identify the walking user

TABLE 3.29: TEST TS-06: RECOGNIZING WALKING USER

ID	*TS-07*
Title	Recognizing running user
Description	Once the user is registered on the system, he accesses to the software, selects the option “Recognize user” and runs while the system collects data to identify the user
Output Result	The software is able to identify the running user

TABLE 3.30: TEST TS-07: RECOGNIZING RUNNING USER

ID	*TS-08*
Title	Recognizing jumping user
Description	Once the user is registered on the system, he accesses to the software, selects the option “Recognize user” and jumps while the system collects data to identify the user
Output Result	The software is able to identify the jumping user

TABLE 3.31: TEST TS-08: RECOGNIZING JUMPING USER

ID	*TS-09*
Title	Cancel user recognition
Description	The user selects the option “Recognize user” and then selects the “Cancel” option while the system is collecting user data
Output Result	The software cancels user identification and returns to the main screen

TABLE 3.32: TEST TS-09: CANCEL USER RECOGNITION

ID	*TS-10*
Title	Error on unknown user recognition
Description	An unregistered user selects the option “Recognize user”
Output Result	The software communicates that the user is unknown for the system and to register before selecting the option “Recognize user”

TABLE 3.33: TEST TS-10: ERROR ON UNKNOWN USER RECOGNITION

3.6.1. TRACEABILITY MATRIX

After stating the requirements, each of them need to be mapped to at least one test, that way all the functionalities offered by the system are checked to ensure everything works properly.

Table 3.34 collects the relationship between the tests and the functional requirements presented on section 3.3.1. All tests check one or more requirements except from test TS-01, which is not mapped with any requirement. This is because test TS-01 checks that system installation can be completed without problems or exceptions, and there is no such requirement

for that. Even though it cannot be mapped, it is included in the test catalogue as it is essential to check that all components can be correctly installed, and the output for the remaining tests depend on the success of it.

Most of the remaining tests are mapped to several requirements. Recalling section 3.3.1, Requirement RSF-01 refers to user recognition, RSF-02 to task recognition, RSF-03 to data acquisition, RSF-04 to monitorization, RSF-05 to data identification and RSF-06 to user incorporation.

Test TS-02 oversees verifying that data is correctly received at the computer using the software from Texas Instruments, which it is why is related to the two data transmission requirements.

Test TS-03 checks that a user is correctly incorporated to the system, which implies the transmission of data, an update to the machine learning model and user incorporation, thus related with requirements 03, 04, 05 and 06. Similarly, test TS-04 checks that the user can cancel the incorporation of a new element to the system correctly, which is why it is related to the same requirements except from 05, which states the update of the model that would not take place in this situation.

Test TS-05, TS-06, TS-07 and TS-08 are almost the same, as they check that a user is recognize if he is resting, walking, running or jumping respectively. As all these tests refer to the different activities that the user can perform while using the system, all check the same requirements even if the activity changes, those being 01, 02, 03, 04 and 05, as this process needs data transmission (requirements RSF-03 and RSF-04), user and task recognition (requirements RSF-01 and RSF-02) and the use of the machine learning model to obtain the output from the system.

AS test TS-09 checks that a user can cancel the recognition process, involves the same requirements.

Finally, test TS-10 checks that, given an unregistered user trying to be identified by the machine learning model, the system can warn of such situation to the user. It is only related to requirement 5 as it would be the only one involved given that the data obtain does not match any of the integrated classes.

		Tests									
		01	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**
Requirements	**01**					X	X	X	X	X	
	02					X	X	X	X	X	
	03		X	X	X	X	X	X	X	X	
	04		X	X	X	X	X	X	X	X	
	05			X		X	X	X	X	X	X
	06			X	X						

TABLE 3.34:TRACEABLITY MATRIX BETWEEN REQUIREMENTS AND TESTS

4. DESIGN

Over this section the system design and architecture to be implemented is explained. It covers a detailed explanation of the different parts that compose the system. User interface mock-ups follow this section with a schematic design of the product that will be presented to the users. Component diagrams, class diagram and sequence diagram come after the mock-ups, to illustrate how the different components of the system interact with each other and the user with the sequence diagrams, and the different data structures and methods to be implemented so that the functionality talked through the analysis is meet.

4.1. SYSTEM ARCHITECTURE

After researching the different options and tools available to develop the proposed user recognition system, the components of the project are settled, giving place to the layout shown on Figure 4.1.

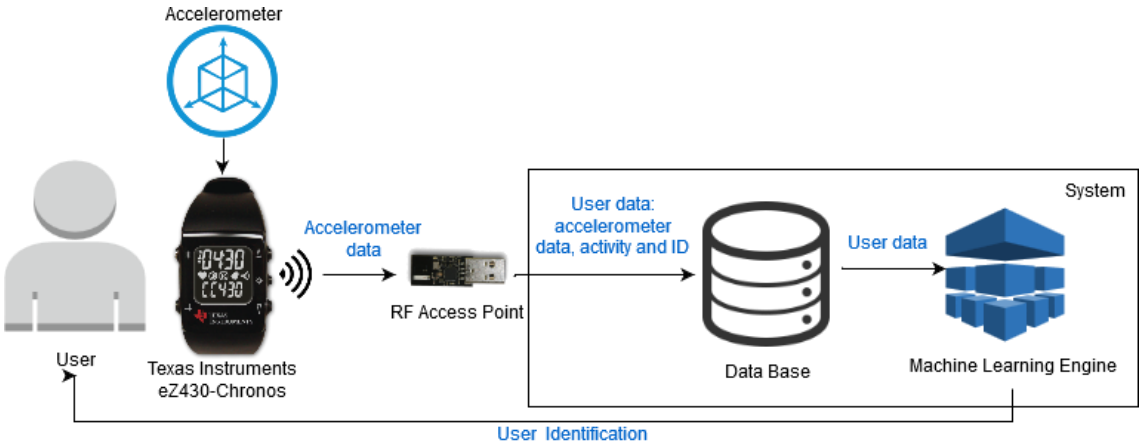

FIGURE 4.1: SYSTEM ARCHITECTURE

The smart device, in this case the smartwatch by Texas Instruments ez430-Chronos, contains a three-axis accelerometer which will send data, when activated, to an access point connected to the computer. Wireless transmission is done over radio frequencies between the smartwatch and the RF Access point connected to the computer. The data transmitted will then be recorded using a Python script.

The script, located on the computer, oversees opening the port and thus the connection to the clock and initiates the listening activity, waiting for the clock to start transmitting data, using rlabs' Python library to manage this process. There are two possible outputs for the received data. On the training period, data is stored to include it on the chosen machine learning model. On the recognition period, data is passed as a stream to the algorithm to recognize the user.

MEKA, the machine learning engine chosen for the project, has two possible outputs too, depending on the use case. If a user wants to be incorporated to the system, then the model

produce by MEKA needs to be updated with the new data. If the model obtained from MEKA is updatable, then the recognition can be done in parallel to the update of the system, as new examples are presented to an updatable algorithm, the most relevant ones are included in the model. If the model is not updatable, then it needs to be generated again to include the information about the new user. The recognition of a new user is done in two steps: first, on a training period, data is collected from different activities and used by MEKA to generate the model again, containing the necessary updates to recognize the new user; then, the user can begin to perform its usual activities and the system will be able to identify him, enabling continuous authentication.

The developed software employs the script and the MEKA model obtained after experimentation to offer the functionality in a simplified way. It counts with a simple GUI that allows the user to choose between adding a new user or recognizing an existing one. Each of the options will call a different routine to comply with the restrictions explained.

The goal of the software is to visually show to the user the accuracy of the system, allowing them to test it by adding more people to the system and then identifying them.

As this software is to be used locally to be able to perform continuous evaluation on a device and it is documented so that it can be portable, using it on Windows or Linux by modifying the parameters indicated on the requirements, it is not recommended to add more than ten users to the system as the accuracy of the classifiers can be affected by a large number of classes (Geoff, Albert, & Bernard , MOA: Massive Online Analysis, 2010).

4.2. USER INTERFACE MOCK-UPS

To be able to perform the tasks talked through the analysis, the user will be presented with a simple GUI that will help them to accomplish the functions and give useful information during the process.

The structure that the GUI follows is the one presented on Figure 4.2, where the main sections are portrayed.

The top bar, *Task name*, presents the user with the process that is being performed at the moment. *User information area* is meant to include useful information for the task that is being performed, indicating the different options available for the user, if the system needs the user to perform any action, as it would happen during the addition of a new user to the system, where resting, walking, running and jumping data needs to be recorder to train the system.

Finally, *Buttons area* includes the buttons with different options, being to start the recognition process, to add a new user or to cancel the current process to return to the main selection screen.

Over the Implementation and evaluation section, the final screens are shown and explained, describing the actions that the user can perform and the tasks that are performed once an option is selected.

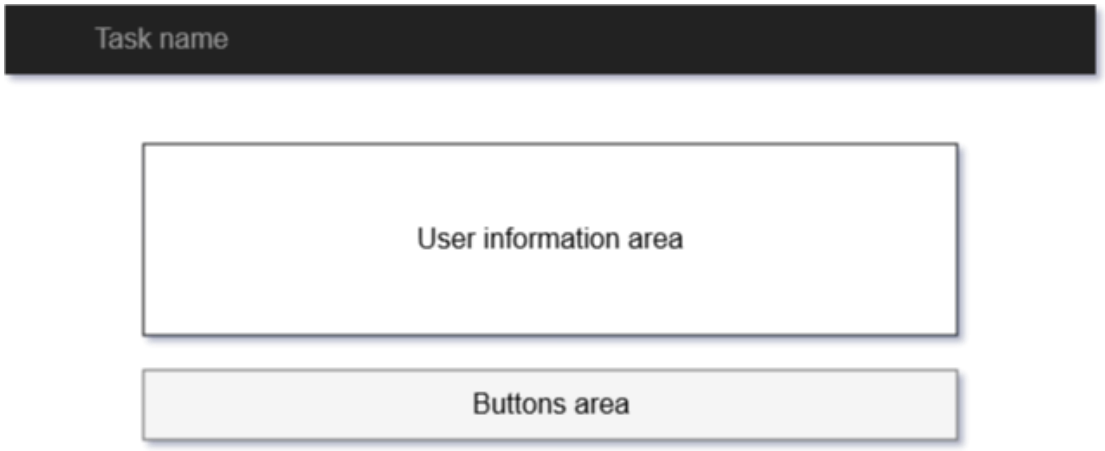

FIGURE 4.2: USER INTERFACE MOCK-UP

4.3. COMPONENT DIAGRAMS

Component diagrams show an organized vision of how a system is organize and the relationships between the different pieces that are part of it. Each of the components might have one or more classes and internally, they can be an abstraction of the interaction between more components.

The system proposed is formed of three main components: the GUI, in charge of the interaction between the user and the system; the data receiver, which gets the data from the smartwatch and parses it so that it can be later used; and the machine learning engine, which uses the parsed stream line either to train a model or to identify a user using that model.

Figure 4.3 includes the relationships between these three components. Afterwards, on Figure 4.4 and Figure 4.5, the pieces of each component are depicted, showing how they work internally and how they treat the information. Marked in purple, the components that are used by the system but not developed exclusively for it are shown in the diagrams.

The system diagram shows how the user software is related to data receiver and the machine learning engine. It sends the selection made by the user to the machine learning engine and, independently of the selection, it gives place to the data receiver. The two possible user selection would be to either add a user or to recognize a person. Both actions need the data

receiver to work in the same way, but the flow of events the machine learning engine needs to perform is different depending on them.

Data receiver depicts its internal structure. There are two components: data reception and data parsing. Data reception employs chronoslib Python library to receive the data sent from the accelerometer. This raw input is then treated by the data parsing component, transforming it in the stream line required by the machine learning engine to be able to process the information.

Machine learning engine contains the artificial intelligence component that will store the model used to recognize the activities and the users. It receives the stream line provided by the data receiver and the user selection from the user software. The machine learning model contains as well two different components: train model, and identify user and activity. They are used depending on the user selection option. If the user wants to be recognized, then its selection will carry them to the user and activity identification, providing as the output the result obtained after processing the stream line. On the other hand, when a user is to be added to the system the train model component is used, which will update the model to include the new information and then pass it to the user and activity identification component so that the latest version is ready to be used.

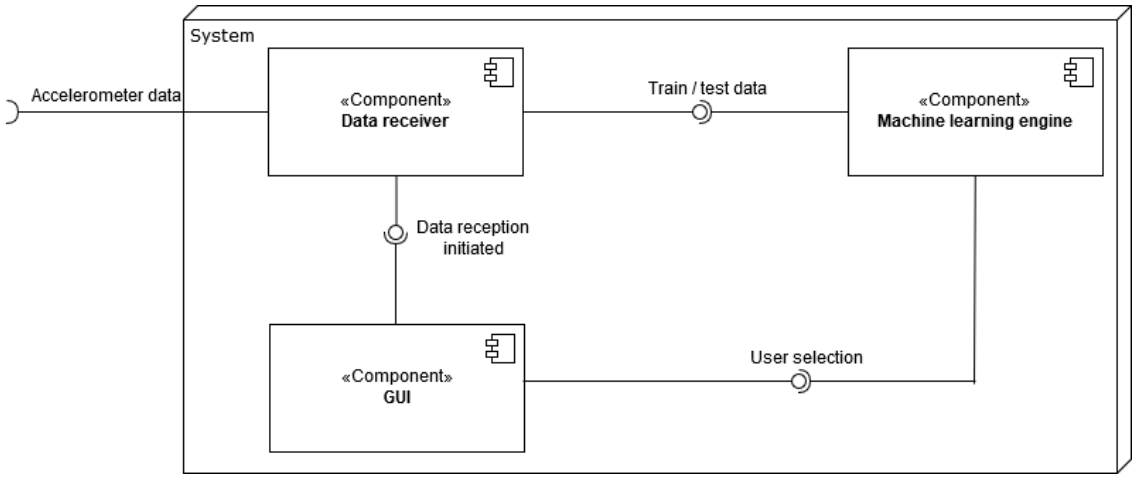

FIGURE 4.3: COMPONENT DIAGRAM (I): SYSTEM

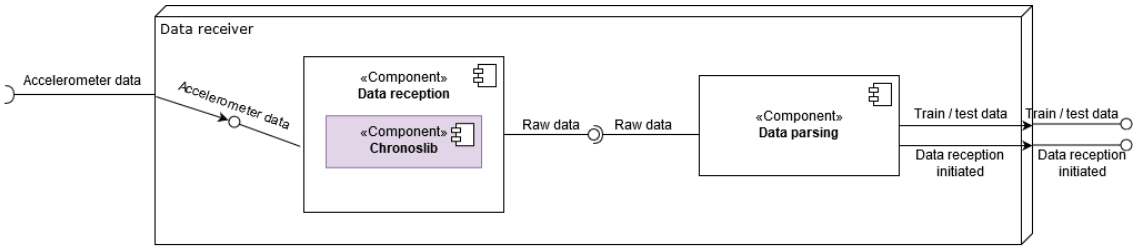

FIGURE 4.4: COMPONENT DIAGRAM (II): DATA RECEIVER

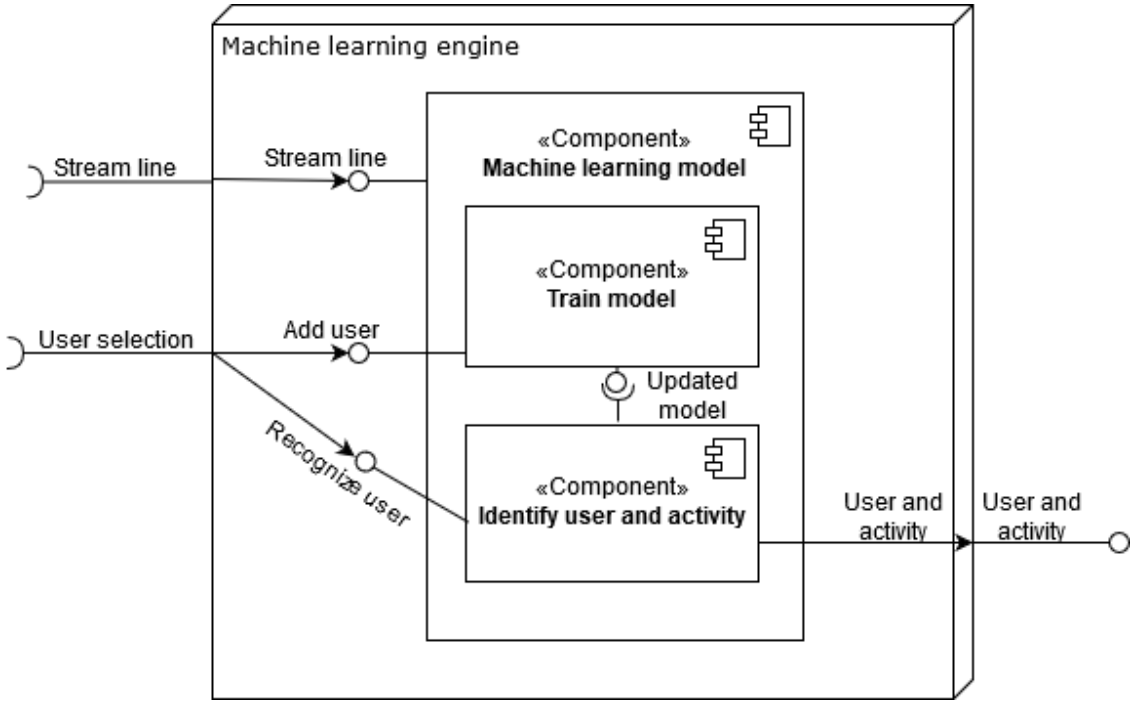

FIGURE 4.5: COMPONENT DIAGRAM (III): MACHINE LEARNING ENGINE

4.4. CLASS DIAGRAM

A class diagram is a schematic view of the system to be implemented by modelling its different classes, objects and functions as well as the relationships between them.

The project counts with four classes: User, which contains the relevant information from the user and will allow to match the machine learning algorithm output with the name of the user; Activity, storing information related to the different activities that can be performed and recognized by the machine learning mode; ACCData, which stores the stream information received from the accelerometer on the smartwatch; and Model, which stores the name of the algorithm, the parameters to configure it, and the users and activities that can be recognized. Each class also includes functions to perform the necessary operations, in addition to the getters and setters' functions -default methods included to set the different attributes of a class or to obtain their current value. The most important functions are trainModel() and getUserActivity from Model, and sendACCData from ACCData. The first one is in charge of training the model if a new user or activity is going to be added, the second function returns the user and the activity being performed given a stream line from the accelerometer and the last one sends the accelerometer data to the model, so that it can be used either to train the model or to get the classes for said data. Figure 4.6 shows the class diagram with this information.

The components stated on the previous section are strongly related with the different classes and structures that define the system. ACCData possesses both attributes and functions from the component Data receiver. The set methods for ACCData are called from the chronoslib

component, which establishes the x, y and z parameters after receiving the data from the user. Data parsing is performed at sendACCData(), where it is modified to meet the structure required by the Model class. The Model class corresponds to the component Machine learning engine, as it stores all the relevant information needed by the Machine learning model as well as the necessary classes to train (trainModel()) and identify (getUserActivity()).

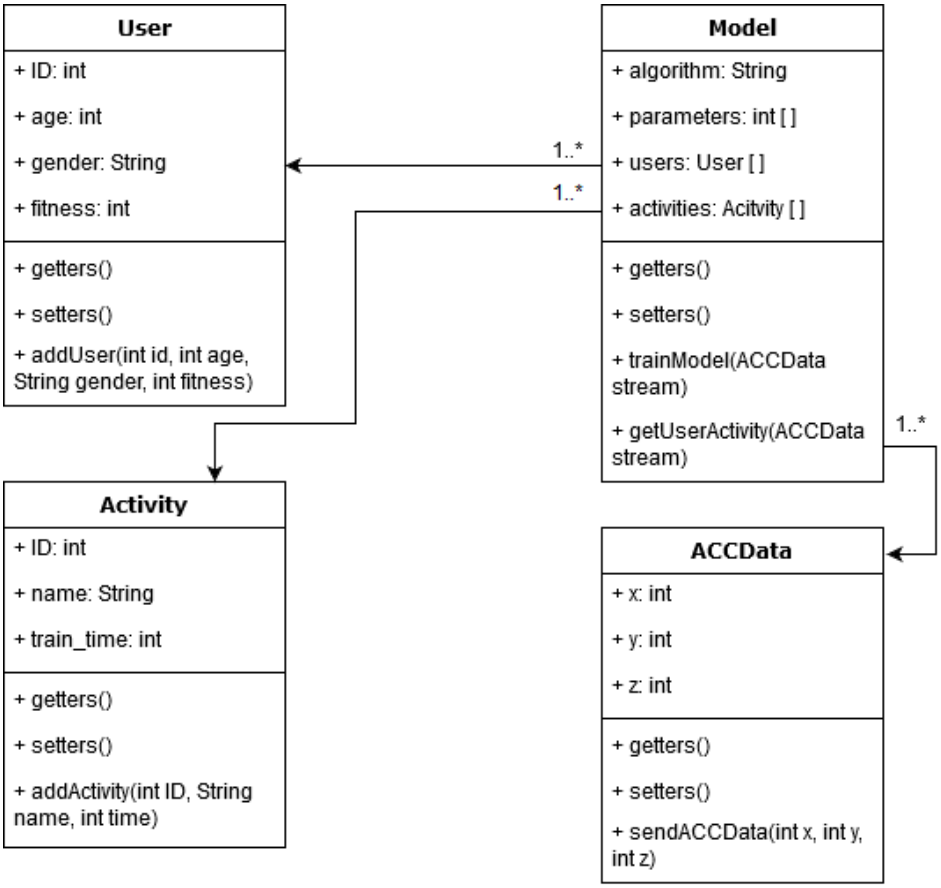

FIGURE 4.6: CLASS DIAGRAM

4.5. SEQUENCE DIAGRAMS

A sequence diagram is used to portray the relationships and interactions among the objects and the order in which the different events happen. They also show the connections between the use cases and the class diagram, as they document how an action performed by a user is carried out internally on the system.

Over Figure 4.7 and Figure 4.8, the internal structure of the use cases explained on Table 3.20 and Table 3.21 respectively are depicted. The first sequence diagram shows how a user is included in the system and its data fixed into the machine learning model so that they can be recognized afterwards. The second sequence diagram shows how the user data is sent into the machine learning model and the answer given by the Model class to the user.

To add a user a new object of the class User needs to be initialized, including it then in the Model user structure to take into account all the users registered in the system. Once the user has been successfully registered in the system, the accelerometer will start to record data to send it to the model in order to train it. On this task, the class ACCData initializes its object to store the data obtained from the three axes and then sends it to the Model class. With a self-call employing the trainModel() function, the model will then train itself to update itself and be able to recognize the user in the future.

User recognition is a much simpler sequence diagram, as there is a lower number of data structures and functions involved on its process. Once the user sets the smartwatch to transmit the accelerometer data, ACCData class will receive this information and set an object to store it. This object will then be sent over to the Model class in order to process its data by a self-call to the getUserActivity() function, which will pass the data through the machine learning model, resulting on the output indicating the user to which the data belongs and the activity that is being performed.

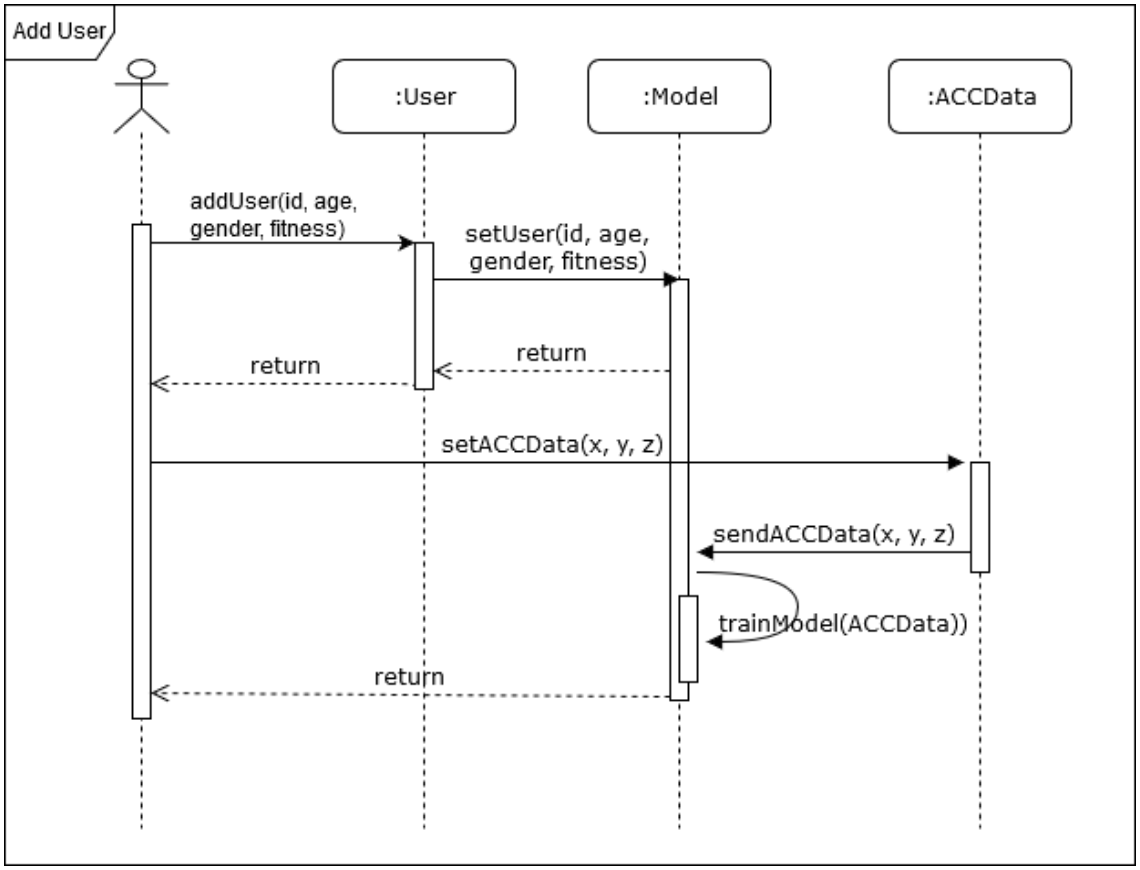

FIGURE 4.7: SEQUENCE DIAGRAM (I): ADD USER

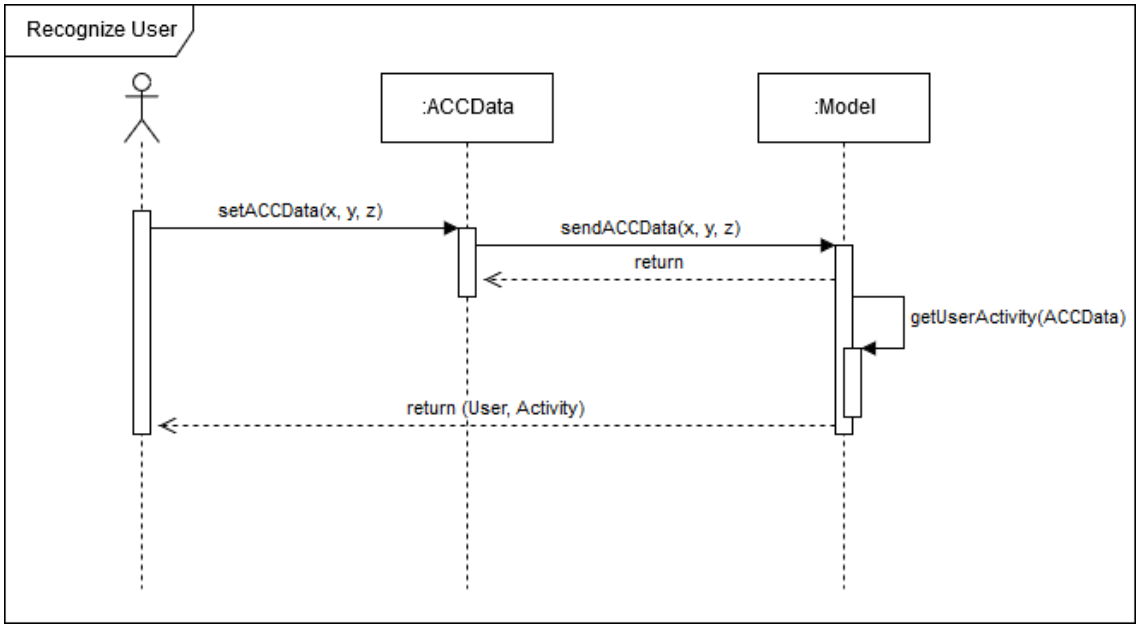

FIGURE 4.8: SEQUENCE DIAGRAM (II): RECOGNIZE USER

5. IMPLEMENTATION AND EVALUATION

This section includes all the information related to the implementation and evaluation of the system, considering the Analysis and Design performed.

The implementation related sections will include the most relevant aspects presented while coding the system and the final user interface.

Over the following sections, the evaluation is talked through, starting with the experimentation process explanation, how the data was acquired and which activities the testing users where asked to perform as well as their profiles, seeing how each of them might affect to the way they carry out any of the activities.

Then, MEKA experimentation is presented to select which machine learning algorithm is the most suitable one for the problem, finishing with the results for the tests proposed over the Analysis and the evaluation form given to the testing users.

5.1. SYSTEM DESIGN ADAPTATION

Once Design established what needed to be done and how it would be implemented by showing the relationships among the different pieces composing the system, the coding process began.

As the machine learning engine chosen to generate the models for this project was either MEKA or MOA and they are coded using Java, several libraries to include the project and access their classes and functions were considered.

Java integration on Python can be done using a library to link both elements. JPype is often the recommended library to include Java jar files and access its classes and functions, and the existing guides showing how to include WEKA on Python use it.

Both MOA and MEKA are extensions to WEKA and include more libraries in their project, in addition to WEKA for MOA, and WEKA and MOA for MEKA, so the final jar has several dependencies that need to be resolved on runtime.

To simplify the final software, MEKA was included, with all the libraries that are needed from Java, and Python *os* library was selected to use in the project. As an integrated library, it allows Python to run instructions on the command line so it would be compatible with both Linux and Windows, although the paths for files must be adapted.

5.2. USER INTERFACE

Using the base proposed over the Design section, the final user interface is presented on the following figures. It has been developed using Tkinter, an integrated Python library to develop GUI. As it includes all the functionality needed and presented on the mock-ups. Each button appearing on the screens will be matched with a function to complete the task chosen by the user.

The first screen, shown in Figure 5.1 displays the two actions that can be done: adding or recognizing a user. Each action calls a different function to carry out the task. If the chosen option is to add then new data needs to be recorded and included into the system, giving place to the screen in Figure 5.2 which displays an informative message to the user communicating that data is being collected and a cancel button that takes the user back to the first screen. When the training period is over, a message indicating so is displayed with a return button -a screen similar to Figure 5.4 -to go back to the main screen. If the chosen action is to recognize an already registered user, then the screen shown is the one in Figure 5.3. This screen reports the user that data is being obtained and offers a cancel button to return to the main screen. After data is collected, a message is displayed, as shown in Figure 5.4, informing the identification of the user and the probability of the output.

A diagram is presented on Figure 5.5 to show the relationship between the different screens, indicating with the arrows that part from one button the state on the system that is reached. All cancel buttons take the user back to the main screen (Figure 5.1) as well as the return button. The two actions on the main screen take the user to two different flow of action, as mentioned on Table 3.20 and Table 3.21.

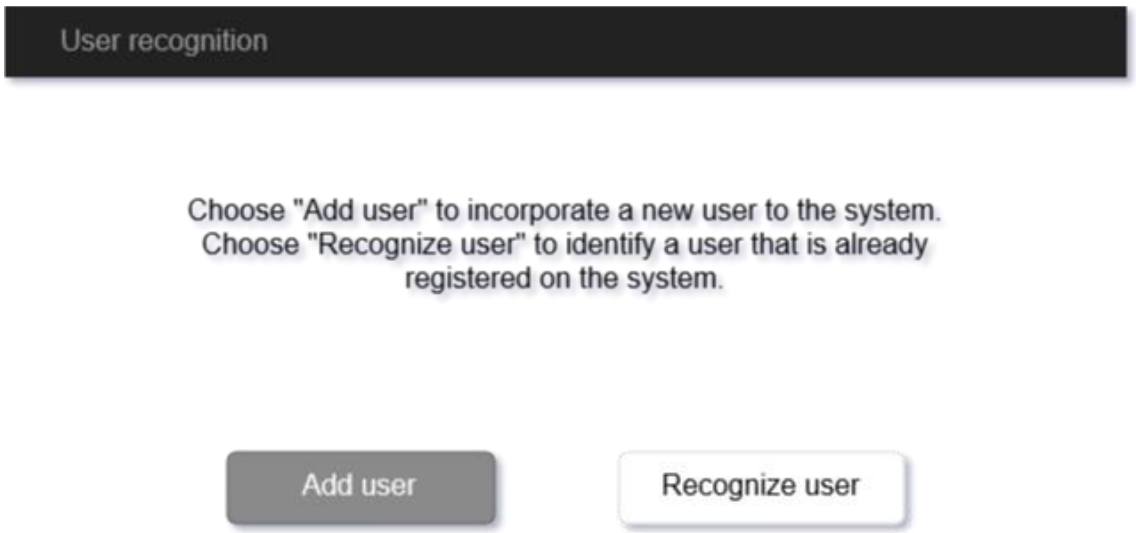

FIGURE 5.1: MAIN SCREEN, ACTION SELECTION

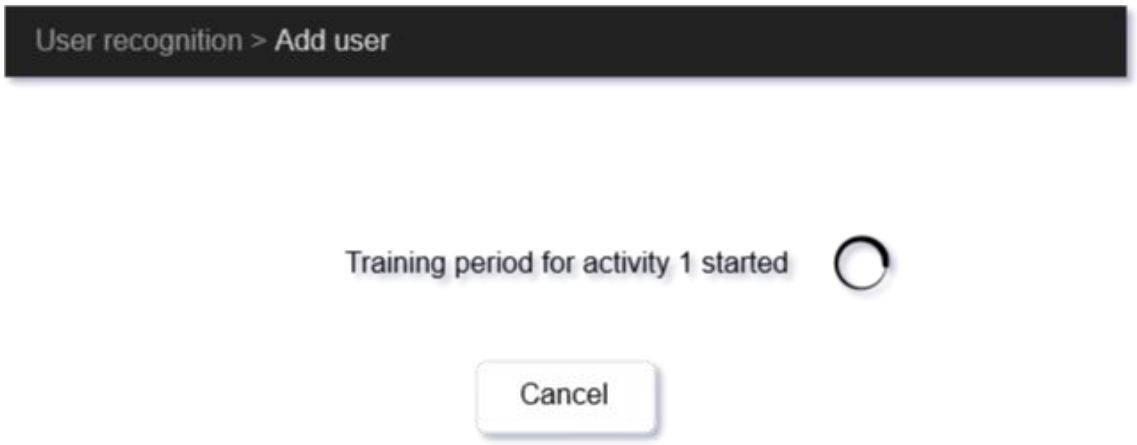

FIGURE 5.2: ADD USER SCREEN

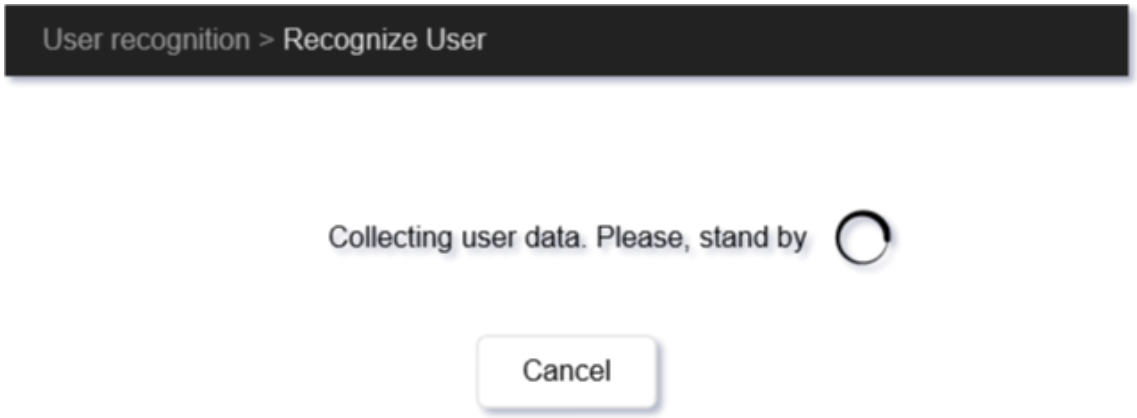

FIGURE 5.3: RECOGNIZE USER SCREEN (I): DATA COLLECTION

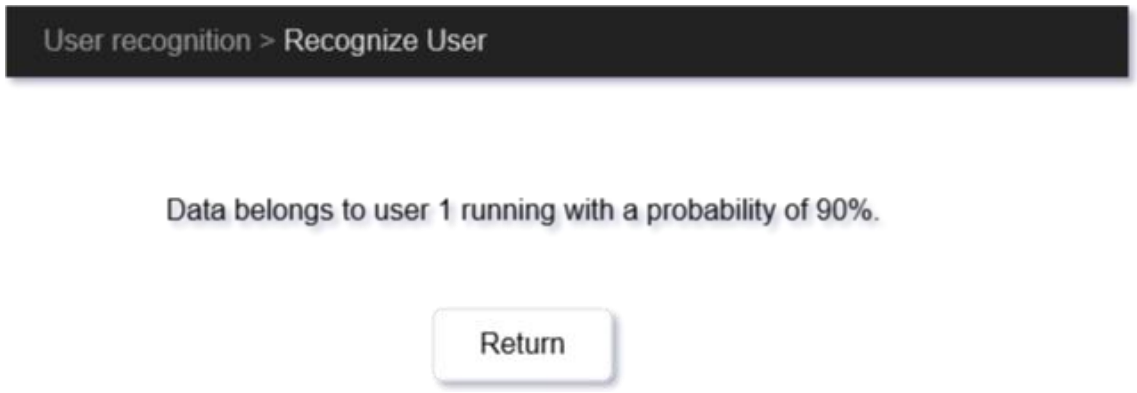

FIGURE 5.4: RECOGNIZE USER SCREEN (II): SYSTEM OUTPUT

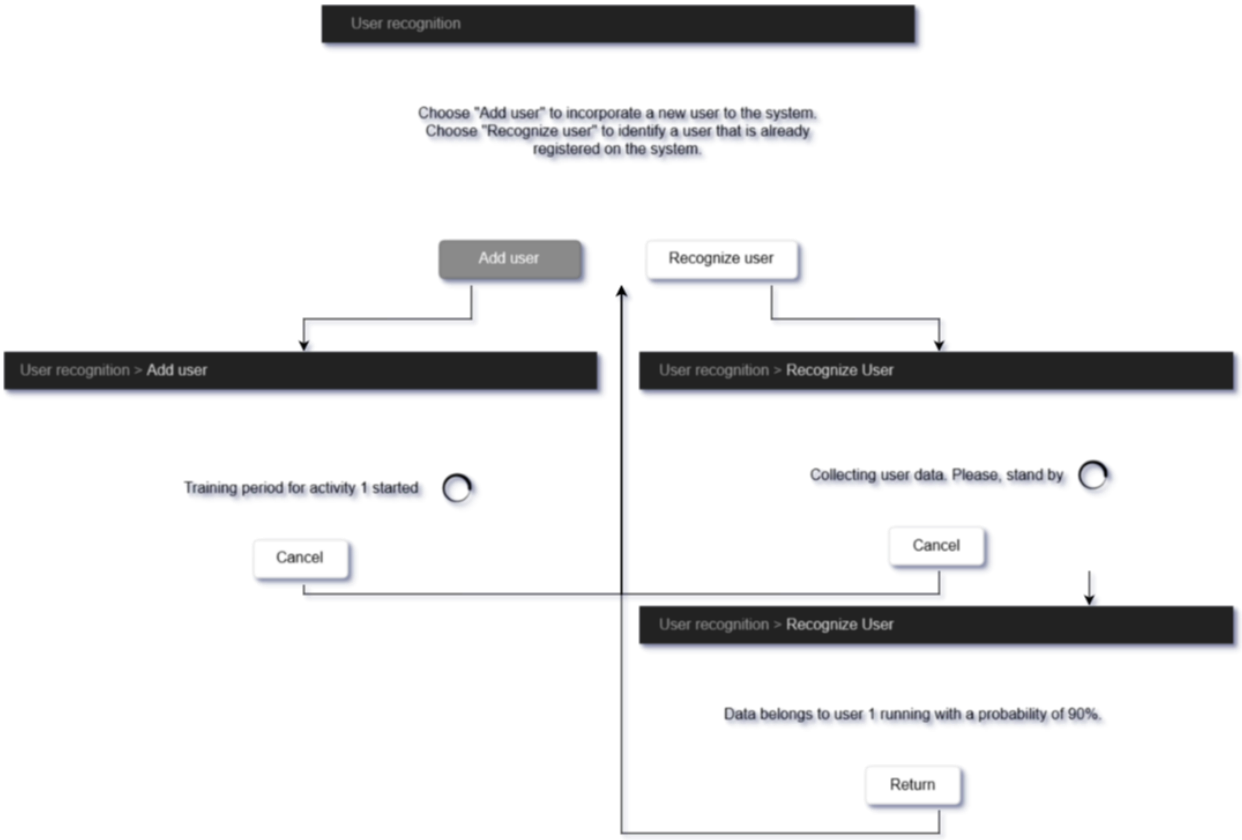

FIGURE 5.5: SCREEN FLOW

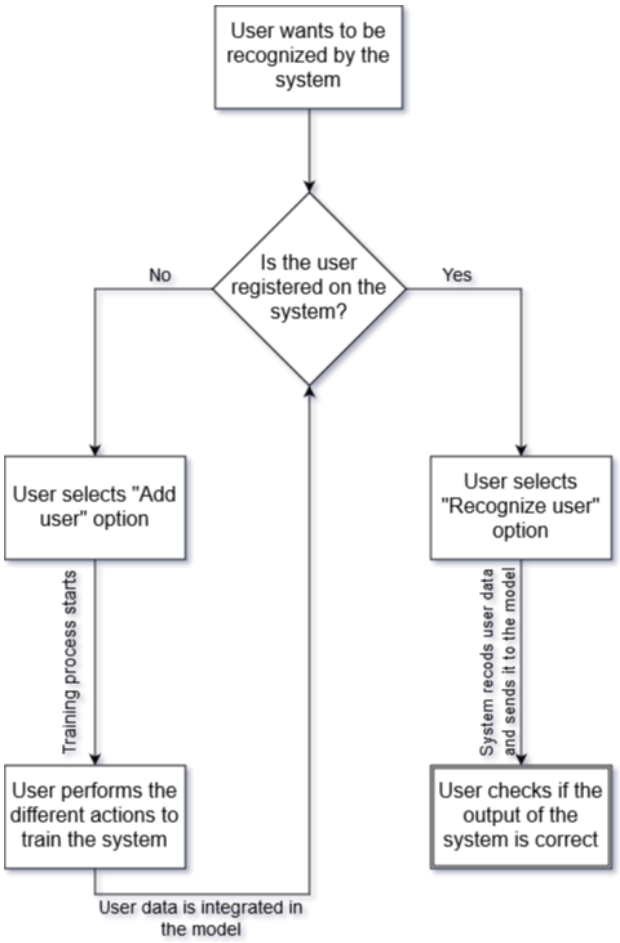

FIGURE 5.6: ACTION FLOW

5.3. DATA ACQUISITION

To train the system four main activities were chosen based on how frequently a person can perform either of them on a day to day basis: rest, walk, run and jump. Seven volunteers were asked to perform each of the activities during a different amount of time to collect enough data that was representative of the user.

These activities were performed on the same environment by all users. A laptop was used to collect the data transmitted by the smartwatch, where the software of the project had already been installed. To avoid the interference that can be caused by walls the users were gathered on Sabatini's courtyard, located on the university campus and shown on Figure 5.7. The courtyard counts with an open space were users could perform the activities without interruptions.

To count with enough data to train and test the models and the complete system afterwards, each task was performed three times and recorded on an *arff* file, the file format used by MEKA, where each line contained the following data:

- X axis data from the accelerometer, a number that can take values from 0 to 255.
- Y axis data from the accelerometer, a number that can take values from 0 to 255.
- Z axis data from the accelerometer, a number that can take values from 0 to 255.
- Activity identifier, a number from 0 to 3, where 0 corresponds with resting, 1 with walking, 2 with running and 3 with jumping.
- User identifier, a number from 0 to 6, different for each user.

The header of the file contains a line with the *@relation* parameter, which usually indicates useful data of the data set, as it is requested by MEKA, telling the number of classes and their position on the data line, in this case being two and on the end of the line: *@relation 'Accelerometer: -C -2'*, the "-" symbol before the number indicates the program that the classes are located at the end of the file.

FIGURE 5.7: SABATINI'S COURTYARD

As each activity needed to be performed three times, data was arranged on three different files as well, each containing the information obtained from each user for each activity. This resulted on three files with an average of 128,795 lines with information of all tasks from all users. To

avoid problems with the model derived from passing ordered data, all the lines were randomized using a filter provided by MEKA.

Over the next section, examples will be provided for the accelerometer information obtained by each user for all four activities, as well as the differences and particularities from them as they were using the smartwatch.

5.3.1. Rest

This activity could prove to be the one with the most uniform data. To avoid this, information was collected over ten minutes. Volunteers were asked to place the smartwatch on the wrist where they usually wear a clock. They were all right-handed and all used the clock on the left hand, this resulted on a minimum variation on the movement. Four of the users where using the computer while data was being collected, so most of the variation of the movement was obtained due to keystrokes, otherwise that hand was still on the table or on the user's lap. Two users were browsing on the internet with their smartphones or talking to each other, and the data obtained from the remaining user was acquired while the volunteer was taking a nap.

A representation of the data obtained from user one is on Figure 5.8. Each spot represents an axis of the accelerometer, the red one corresponds to x axis, the green one to y axis and the blue one to z axis. The graphic represents data over time, thus the y axis contains time data and x axis contains the value taken by an accelerometer axis on that moment. All the graphics obtained to represent the information follow this structure.

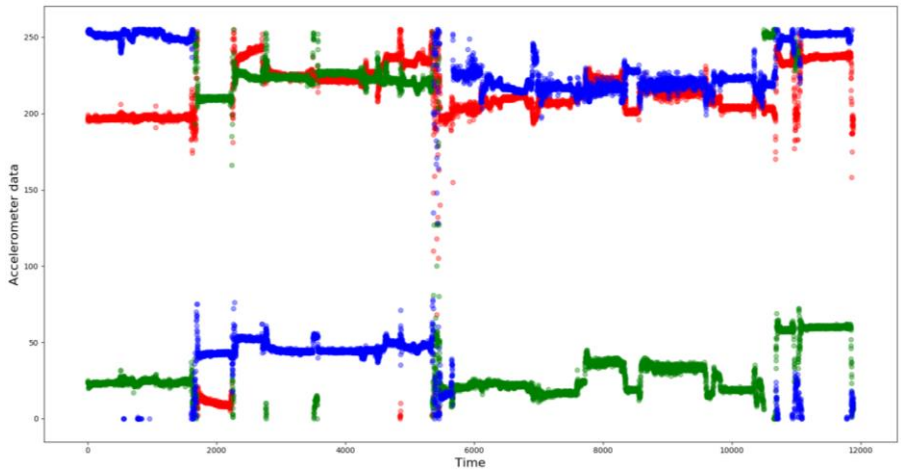

Figure 5.8: Rest data from user 1

5.3.2. WALK

To avoid data loss, all users were briefed before making the test about the restrictions of the smartwatch, mainly the distance limit for data transmission.

After checking up to which distance the users could perform the test without losing data, they were asked to walk for two minutes as they would do on the street, so some users walked with their hands in their pockets, others with crossed arms and some while browsing on their smartphone.

The results obtained from the walking test are the ones that show the greatest difference between the users. On Figure 5.9 the data from user three is represented and the data from user four is on Figure 5.10 The way user three walk is completely different from the way user four walks, as we can see from the position of the axis, which are opposite. On user three, z data takes mostly low values whereas x and y take mostly high values, usually the x-axis presenting the highest. User four results are completely different for every axis, as x and y values are always low, the x taking the lowest values, and z data comes and goes from higher to lower values. The most interesting thing about the difference between these two users is that they appeared to walk in the same way, as their arms were relaxed and out of their pockets, and their movements seemed to be very similar, but from what can be observed on the graphs, it is not the case, which shows that it is possible to distinguish between two users even if their movement seems the same at first sight.

FIGURE 5.9: WALK DATA FROM USER 3

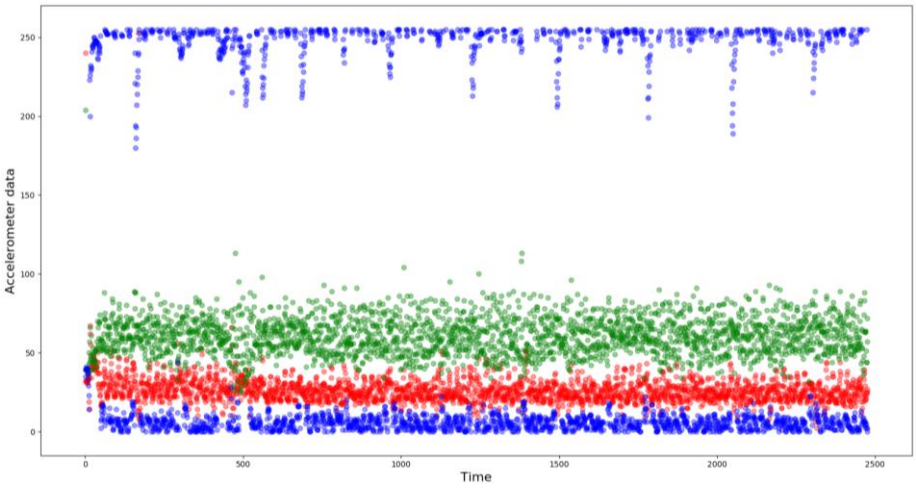

FIGURE 5.10: WALK DATA FROM USER 4

5.3.3. RUN

The running test was done in the same way as the walking test, with the only difference being the activity performed. As all the volunteers knew the limits of the space, they run over the same space for two minutes each.

The results obtained on this test are more homogenous and the differences not as obvious as the example presented on the walking test. This fact might be caused by the space restriction to perform the activity, as it was limited, there was not enough room for the users to carry on with the task as they would do on a wider space. Figure 5.11 presents the data obtained from user five during this task.

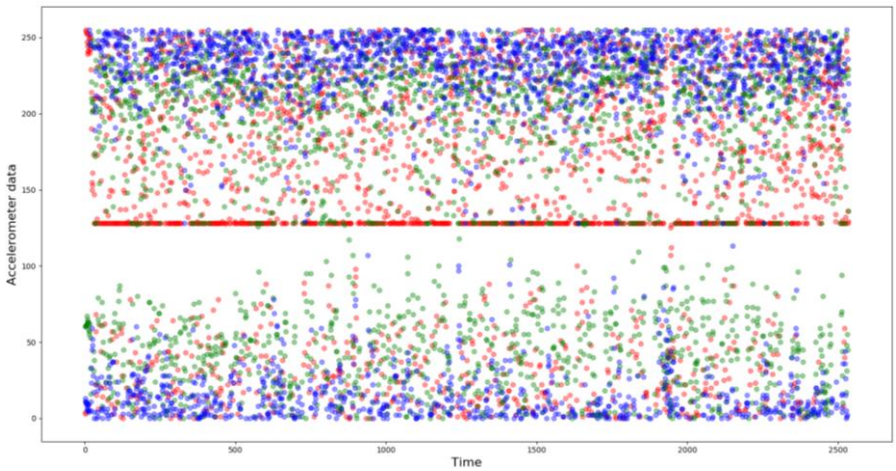

FIGURE 5.11: RUN DATA FROM USER 5

5.3.4. JUMP

Even though considering jumping as a usual task can be argued, the purpose of asking the volunteers to perform it was to emulate a similar behaviour to climbing the stairs. The time given for the test was also two minutes, and users were asked to jump on one spot without a skipping rope to achieve the maximum difference on performing the action among them. If any of them became tired they were asked to stop and rest before continuing.

The graph on Figure 5.12 shows the jumping data obtained from user six. Besides de resting periods present on some users recorded data, this task presents the same problem as the running test. Results are more homogeneous and difficult to distinguish between them. The oscillation of all axis between low and high values that can be observed on Figure 5.12 happens on almost all users, even though it happens on different axis and with different minimum values. From the graphic representation, data cannot be distinguished as easily at first sight.

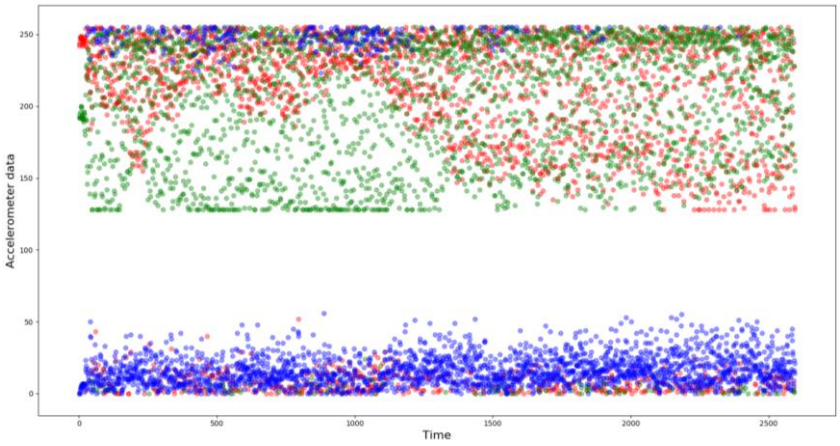

FIGURE 5.12: JUMP DATA FROM USER 6

5.4. TEST USERS

To check the functionality of the system and collect data to train the system, volunteers where looked for. They were briefed on the project goals and what would be needed from them, to perform the activities mentioned on the previous section.

A user profile was elaborated to check if their behaviour affected the way they could perform a task, so they were asked to fill a form and the following data, presented on Table 5.1, was stored:

- ID: a unique number from 0 to 6 to identify each user.
- Age, most users were between 20 and 22.
- Gender: where W stands for woman and M for men.
- Fitness: a number from 1 to 5 indicating the exercise that the volunteer performs, where each number corresponds to:
  - 1: no exercise.
  - 2: exercise only done sporadically.
  - 3: exercises once or twice a month.
  - 4: exercises two times per week.
  - 5: exercises more than twice per week.

ID	Age	Gender	Fitness
0	22	W	1
1	20	W	1
2	57	W	2
3	22	M	5
4	22	W	1
5	21	M	3
6	20	M	2

TABLE 5.1: USERS PROFILE

Apart from the case presented with two users performing the same activity in a seemingly equal way, there are other differences to be expected among the users based on their profile.

Users 0 and 6 have are very similar in terms of age and fitness, the difference between them being on their gender. Their resting and walking data are akin but their jumping data diverges. User 0's jumping data presents continuous ups and downs on the z axis, while user 6 usually stays stable, either maintaining high or low values, and with a few outliers coming from the resting periods. User 0 also presented more difficulty performing the test, while user 6 managed to complete the task without many breaks.

On the other hand, user 0 and 2 share the same gender and a similar fitness, but the age gap between the two of them made the results turn out differently. User 2 got easily tired while jumping and running, resulting on several stops that where portrayed on the results, and walked

at a slower speed than user 0, the results being much more uniform and with less outliers. User 0 walked and jumped faster, although the number of stops made by user 0 while jumping was nearly the same as user 2, and didn't need any stop while running, while user 2 switched to walking when she became too tired to walk.

Finally, to check the difference based on the fitness, we have user 3 and user 6, who share a similar age and their gender. The main achievement made by user 3 was the lack of breaks taken during either of the activities, managing to complete them only resting between one round and the next one. User 6 took pauses between one activity and the next one, and used the walking test to recover for the following tasks.

Another thing worth to mention is that no user stretched before performing the exercises, which caused several of them to suffer from a pulled muscle and the consecutive pause to stretch and recover from the pull. Some of them continued with the exercises and stretched after finishing the activity, but this pause or outliers coming from the stretching exercises can also be seen on some of the data obtained from the users.

The order in which the tests were performed was the same for all users, so that results could be compared among them, the series being walking, jumping and running with breaks between them if the volunteer needed them. The resting test was done on one sitting, as it was the easiest and less demanding test for all users. The three series of the activities where done one after the other, and the results for the last task show the fatigue for some of the volunteers, mainly the ones with lower fitness, as they needed more breaks while jumping and most of them run at a slower speed.

5.5. MOA AND MEKA EXPERIMENTATION

Once all the necessary data was collected from the users, the machine learning model to be implemented needed to be decided. Both MOA and MEKA provide a wide range of options to choose from between classifiers and clusters on MOA and classifiers on MEKA.

From the State of the art, the algorithms most commonly used on projects using biometric data to identify users is Dynamic time warping. There is one paper which used Random forests, and as the problem was also a classification task, it was used as a guide to choose which models evaluate and test more thoroughly.

On MOA, the focus was to test the clustering models, as MEKA does not provide them. The problem raised when the model needed to be generated and evaluated. As mentioned before, the files provided to the software to generate the models are considerably long, which lead to long periods of training and testing. With clusters, this period was too long, surpassing seven hours. This motivated the decision not to use clustering, as the system needs to be fast on recognizing a user and updating a model to include the new information, and this was not possible with this type of algorithms.

With MEKA, all classifiers for multi-target classification were tested. To evaluate the results obtained the following metrics where used (Kaminka, et al., 2016):

- **Accuracy per label**: calculates the ratio of the union and intersection of the predicted and actual label sets, averaged over all examples:

$$apl = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{|Y_i \cap \widehat{Y}_i|}{|Y_i \cup \widehat{Y}_i|}$$

  Where Y_i and $\widehat{Y}_i$ are the sets of actual labels and predicted labels of an instance, respectively.

- **Exact match**: examines if the predicted and actual label subsets are equal or not:

$$em = \frac{1}{|D|}\sum_{i=1}^{|D|} I(Y_i = \widehat{Y}_i)$$

  Where I(true) = 1 and I(false) = 0.

- **Hamming loss**: takes the proportion of misclassified labels (labels predicted incorrectly and correct labels that are not predicted) averaged over all examples:

$$hl = \frac{1}{|L| * |D|}\sum_{i=1}^{|D|}|Y_i \Delta \widehat{Y}_i|$$

  Where Δ is the symmetric difference.

The election of these metrics was based on the MEKA paper (Geoff, Jesse, Peter, & Berdard, MEKA: A Multi-label/Multi-target Extension to Weka, 2016), where they were mentioned among the most commonly used and precise to evaluate a model.

Firstly, all algorithms were tested with their base configuration, avoiding changing any parameters to get a uniform view of the results obtained and to be able to compare the different algorithms. Then, the models providing the best results are selected and enter a second round of experimentation changing the parameters used to improve the results.

Over Table 5.2 the results of the first part of the evaluation process are recorded. The structure of the table is the following: on the first column, the Chaining algorithm used to link the classifier is indicated, the second column indicates the classifier used for each label (the same for both labels), the third column indicates the evaluation method and the three last columns show the results of the evaluation for the metrics mentioned (accuracy per label, where the first value corresponds to the activity class and the second to user, exact match and hamming loss).

Chain method	Algorithm	Evaluation	Accuracy	Exact match	Hamming loss
Bcc	J48	Cross-validation	[0.908 0.766]	0.744	0.163
		Train/test split	[0.906 0.762]	0.74	0.166
	Random Forest	Cross-validation	[0.901 0.755]	0.733	0.172
		Train/test split	-	-	-
	Random Tree	Cross-validation	[0.880 0.729]	0.701	0.196
		Train/test split	[0.879 0.728]	0.699	0.197
	Hoeffding Tree	Cross-validation	[0.878 0.572]	0.544	0.275
		Train/test split	[0.838 0.535]	0.504	0.314
CC	J48	Cross-validation	[0.908 0.766]	0.744	0.163
		Train/test split	[0.906 0.762]	0.74	0.166
	Random Forest	Cross-validation	-	-	-
		Train/test split	-	-	-
	Random Tree	Cross-validation	[0.880 0.729]	0.701	0.196
		Train/test split	[0.879 0.728]	0.699	0.197
	Hoeffding Tree	Cross-validation	[0.878 0.572]	0.544	0.275
		Train/test split	[0.838 0.535]	0.504	0.314
CCp	J48	Cross-validation	[0.908 0.766]	0.744	0.163
		Train/test split	[0.906 0.762]	0.74	0.166
	Random Forest	Cross-validation	-	-	-
		Train/test split	-	-	-
	Random tree	Cross-validation	[0.880 0.729]	0.701	0.196
		Train/test split	[0.879 0.728]	0.699	0.197
	Hoeffding Tree	Cross-validation	[0.878 0.572]	0.544	0.275
		Train/test split	[0.838 0.535]	0.504	0.314
CR	J48	Cross-validation	[0.908 0.764]	0.733	0.164
		Train/test split	[0.906 0.762]	0.73	0.166
	Random Forest	Cross-Validation	-	-	-
		Train/test split	-	-	-
	Random Tree	Cross-validation	[0.880 0.726]	0.689	0.197
		Train/test split	[0.879 0.729]	0.69	0.196
	Hoeffding Tree	Cross-validation	[0.878 0.575]	0.532	0.274
		Train/test split	[0.838 0.520]	0.468	0.321
NSR	J48	Cross-validation	[0.898 0.762]	0.739	0.17
		Train/test split	[0.896 0.759]	0.735	0.172
	Random Forest	Cross-validation	-	-	-
		Train/test split	[0.895 0.756]	0.731	0.174
	Random Tree	Cross-validation	[0.878 0.727]	0.699	0.198
		Train/test split	[0.877 0.728]	0.699	0.197
	Hoeffding Tree	Cross-validation	[0.832 0.570]	0.536	0.299
		Train/test split	[0.830 0.574]	0.539	0.298
SCC	J48	Cross-validation	-	-	-
		Train/test split	[0.896 0.759]	0.735	0.172
	Random Forest	Cross-validation	-	-	-
		Train/test split	-	-	-
	Random Tree	Cross-validation	[0.878 0.727]	0.699	0.198
		Train/test split	[0.877 0.728]	0.699	0.197
	Hoeffding Tree	Cross-validation	[0.832 0.570]	0.536	0.299
		Train/test split	[0.830 0.574]	0.539	0.298

TABLE 5.2: MEKA EXPERIMENTATION RESULTS

It is clear from the results obtained and presented on the table that the best solution comes from Bcc and J48 with Cross-validation. J48 is an implementation on MEKA of C4.5 algorithm, which creates a decision tree and is an extension of ID3. The process to obtain a classification tree is the following:

1. Check base cases.
2. For each attribute *a:*
    a. Find the normalized information gain from dividing *a*.
3. Being *a_best* the attribute with the highest normalized information gain:
    a. Create a decision node that divides *a_best*.
4. Repeat on the subsets obtained after dividing *a_best* and add them as children nodes.

The base cases mentioned above are few and are collected on the following points:

- All examples from the set belong to the same class. The resulting tree consists on a child node indicating to select said class.
- The attributes do not provide any information gain. A decision node is created on top of the tree using the expected class value.
- An example of an unknown class is introduced. A decision node is created on top of the tree with the expected class value.

Even though it is clear that J48 can obtain good results with its default configuration for the project, Hoeffding trees will also be included on the second round. Hoeffding trees can update themselves using new examples introduced as inputs to the system to be classified, and so the learning process never ends and the tree will always contain the latest and most relevant information from the proposed data. Tweaking the algorithm parameters might lead to better results and, as the data provided by the user comes in streams and the way a task is performed might change due external causes as an injury, an updated system without the need to periodically regenerate the model each time a new user is added or something on the way they move changes.

The Hoeffding algorithms provides an incremental decision tree that can learn from data stream, making it suitable for massive online learning or big data, based on the continuity of the example generation distribution. Hoeffding trees are based on the fact that any example is suitable to select an optimal splitting attribute. This is based on mathematical concept of the Hoeffding bound, which provides an upper bound for the probability of the deviation from an expected value from the sum of random variables.

The basic algorithm process is the following, starting with a single leaf to represent the root (Geoff, Albert, & Bernard , MOA: Massive Online Analysis, 2010):

1. For all the training examples:
    a. Classify an example into a leaf *l* using HT.
    b. Update *l* data.
    c. Increment the number of examples at *l*, n_1.
    d. If $n_l mod\ n_{min} = 0$ and the examples at *l* do not belong to the same class:
        i. Obtain $\bar{G}_l(X_l)$ for each attribute.
        ii. If X_a is the attribute with the highest $\bar{G}_l$ and $X_b$is the second highest $\bar{G}_l$ attribute: compute the Hoeffding bound $\epsilon = \sqrt{\frac{R^2 \ln(1/5)}{2n_l}}$.
        iii. If $X_a \neq X_0$ and $(\bar{G}_l(X_a) - \bar{G}_l(X_b) >)\epsilon$ or $\epsilon < \tau$:
            1. Replace with an internal node division on X_a *l*, and for all branches of the division:
                a. Add a new leaf with keyed in statistics.

This gives place to the second half of the evaluation process to select the most suitable machine learning algorithm for the system.

In addition to the information already stated about the results obtained, there is one fact that brings attention, and that is the difference in the accuracy between the two labels, activity and user. Activity always achieves a higher accuracy than users, and this might be related to the fact that, from what we can see on the different graphs presented on the Data acquisition section, the values vary greatly from task to task, but the difference between two users performing the same task is as pronounced.

This means that the system is able to obtain the activity being performed based on the data that it receives but, when it comes to users, it is not as capable to predict the user with the same confidence as it is able with the activity.

Over Table 5.3, the experimentation done over the J48 algorithm with cross validation evaluation are stated. To perform this evaluation, different parameters were changed to see how they affected to the results, maintaining the default values for them and changing only one each test and combining the values which gave the best results on a final evaluation. The parameters used on J48 are:

- **minNumObj**: minimum number of objects per leaf.
- **confidenceFactor**: confidence factor used for pruning -the process in which non-predictive parts of an algorithm are discarded. Lower values result on more pruning.

The default values for the algorithm are 2 for minNumObj and 0.25 for confidenceFactor.

Parameters	Accuracy	Exact match	Hamming loss
minNumObj = 2	[0.908 0.766]	0.744	0.163
minNumObj = 10	[0.909 0.769]	0.748	0.161
minNumObj = 20	[0.909 0.767]	0.746	0.162
minNumObj = 15	[0.909 0.768]	0.747	0.162
confidenceFactor = 0.5	[0.904 0.759]	0.736	0.168
confidenceFactor = 0.1	[0.909 0.769]	0.749	0.161
confidenceFactor = 0.05	[0.909 0.769]	0.748	0.161
minNumObj = 10, confidenceFactor = 0.1	[0.909 0.769]	0.748	0.161

TABLE 5.3: BCC J48 EXPERIMENTATION RESULTS

The results show how little the accuracy varies when the different parameters are modified. After tweaking the parameters and exploring up to which value they could improve, the best result is obtained with 10 objects per leaf and a confidence factor of 0.1. Neither of the metrics vary greatly after this modification, nonetheless, as the results show a minimum improvement, the final model will contain those values for the mentioned parameters.

Once the parameters for J48 are set, Hoeffding trees are explored and the results obtained after changing the algorithm parameters are shown over Table 5.4. As the best results for this algorithm were also obtained using Bcc and cross-validation, Hoeffding trees experimentation use the same process as the one proposed before for J48, where only one parameter varies from the default values. In this case, the ones that are going to be modified are:

- **batchSize**: number of instances to process if batch prediction is being performed.
- **gracePeriod**: number of examples to be processed before a leaf can attempt another division.
- **leafPredictionStrategy**.
- **minimumFractionOfWeightInfoGain**: minimum weight fraction needed for two branches for info gain splitting.
- **naiveBayesPredictionThreshold**: the number of instances a leaf needs to process before allowing naïve Bayes (adaptive) to predict.

The default values for these parameters are 100 for batchSize, 200 for gracePeriod, Naïve Bayes Adaptive for leafPredictionStrategy, 0.01 for minimumFractionOfWeightInfoGain and 0 for naiveBayesPredictionThreshold.

Parameters	Accuracy	Exact match	Hamming loss
Default	[0.878 0.572]	0.544	0.275
batchSize = 200	[0.878 0.572]	0.544	0.275
batchSize = 500	[0.878 0.572]	0.544	0.275
gracePeriod = 100	[0.879 0.578]	0.552	0.272
gracePeriod = 500	[0.878 0.569]	0.542	0.277
leafPredictionStrategy = Majority class	[0.849 0.394]	0.37	0.378
leafPredictionStrategy = Naïve Bayes	[0.873 0.571]	0.538	0.278
minimumFractionOfWeightInfoGain = 0.1	[0.875 0.584]	0.556	0.27
minimumFractionOfWeightInfoGain = 0.5	[0.772 0.332]	0.276	0.448
minimumFractionOfWeightInfoGain = 0.2	[0.870 0.606]	0.577	0.262
minimumFractionOfWeightInfoGain = 0.15	[0.871 0.597]	0.569	0.266
naiveBayesPredictionThreshold = 10	[0.876 0.551]	0.523	0.286
naiveBayesPredictionThreshold = 5	[0.876 0.563]	0.535	0.28
BatchSize = 100, gracePeriod = 100, leafPredictionStrategy = Naïve Bayes adaptive, MinimumFractionOfWeightInfoGain = 0.15, NaiveBayesPredictionThreshold = 0	[0.871 0.590]	0.562	0.27

TABLE 5.4: BCC HOEFFDING TREE EXPERIMENTATION RESULTS

This time, the results show more conflict, as several parameters showed that, while the accuracy for one of the labels worsen the other label accuracy improved, as it is the case between the default results and with MinimumFractionOfWeightInfoGain = 0.2. On these cases, the value that was chosen for the parameter was the one that affected less to the labels, improving as much as possible the second label without much interference on the first. With some other parameters, the default value was maintained because either the results did not suffer any change or they were worse. This ended with choosing a batch size and grace period of 100, maintaining the default option for naïve Bayes prediction threshold and leaf prediction strategy, 0 and naïve Bayes adaptive respectively, and a minimum fraction of weight info gain of 0.15.

The results for either of the algorithms do not vary much from the ones obtained using the default values, but its modification carried to another observation while obtaining the models: J48 needs over twenty minutes to generate and evaluate a model while Hoeffding trees only need five minutes for completing the same process with the same evaluation method. This raises the question whether the different accuracy between the algorithms is worth the extra time and if users would be willing to wait more for the algorithm to update or include new users.

The accuracy for the first label, activity, is low compared with the difference on the accuracy for the second label, user, being an average of 0.2 higher with J48 than with Hoeffding trees. If the results were the opposite, the difference for the activity could be neglected, as it is more important to recognize the user than to recognize the activity that is being performed, but as the main objective is to be able to tell apart several users, J48 is more appropriate for the problem even if it takes longer to obtain the machine learning model.

5.6. EVALUATION RESULTS

After performing each of the tests proposed on the Test catalogue section, the results obtained for each of them are shown in Table 5.5. All tests where successfully overcomed.

Test	*Result*
TS-01	Success
TS-02	Success
TS-03	Success
TS-04	Success
TS-05	Success
TS-06	Success
TS-07	Success
TS-08	Success
TS-09	Success
TS-10	Success

TABLE 5.5: EVALUATION RESULTS

5.7. EVALUATION FORM

To obtain feedback from the volunteers that provided data to train the system and tested it afterwards, a form was created for them to fill with their opinion on the testing process and on the system. It was used to get a first view on if a user identification system would be accepted by the public.

The form was created using Google Forms so that all volunteers could access it and provide honest answers to the proposed questions. As not all users knew enough English to understand and answer the different queries, it is written in Spanish. Users were asked to answer to all the questions on the form.

It consists of two section. On the first section the questions are oriented to the testing process, checking the opinions of the users on the proposed plan to train the system. The queries asked on it are the following:

- Do you think that the activities proposed are enough? (To rest, walk, run and jump). The answer to this question is limited to “Yes” or “No”.
- Would you add any activity? If the answer is yes, which activity would you add and why? The users are allowed to write a long paragraph as an answer.
- Do you thing that the time to acquire data was enough? (2 minutes for walking, running and jumping, and 10 minutes for resting? The answer to the question is limited to “Yes” or “No”.

- Would you modify the data acquisition time for any of the activities? If the answer to this question is yes, then answer with the following format: activity – X minutes, where X is the time that you would consider necessary for the activity.

The second section is focused on the use of the system and contains the following questions:

- Did the use of the smartwatch affect to the way in which you perform any of the activities? The answer is limited to the options “Yes” or “No”.
- Would you prefer the use of an alternative device other than a smartwatch to acquire accelerometer data? (For example, a smartphone or a smart band). If the answer is yes, which device? The users could write a small paragraph as an answer,
- Do you think that user identification by accelerometer data is a valid alternative to other forms of biometric identification? The answer is limited to the options “Yes”, “No” or “Perhaps”.
- Would you use a system that verified users based on accelerometer data? The answer is limited to the options “Yes”, “No” or “Perhaps”.

After all seven volunteers completed the form, the obtained results were analysed.

Almost all users thought that the activities proposed where enough to create the system, although one disagreed, as shown on Figure 5.13. The next question showed that four of the users thought that other activities should be added to the system, and so they proposed riding a bicycle, swimming and dividing resting into two different activities: a person movement while they are sleeping and a person movement while they are not performing any physical activity. The main drawback mentioned was the range of the smartwatch, which would not allow activities such as riding a bicycle, and that it was not waterproof, making impossible an activity such as swimming.

The third question from the first section showed that almost all users agreed that the time to acquire data was enough, although the next question showed that some of them would change the time for some of the activities: reducing rest to five minutes, increasing walking and running to five and four minutes respectively, or running to ten minutes, and reducing jumping to one minute. Volunteers were contacted afterwards to check why they wanted to increase the time for running and walking and they defended that a person way of walking or running is affected by the time they are performing the activity, as they change the way they move the more tired they become by, for example, reducing the speed.

From the second section, all volunteers thought that using a smartwatch did not affect the way they performed any of the activities and on Figure 5.15 can be observed that some of them would rather use their mobile phones or a smart band instead of the smartwatch.

The third question showed that not all users were sure if identifying people by their movement was a method as reliable of user recognition as other biometric methods, as shown on the pie chart on Figure 5.16 and from Figure 5.17 we can see that almost all users would be willing to use a system that used accelerometer data to verify their identity.

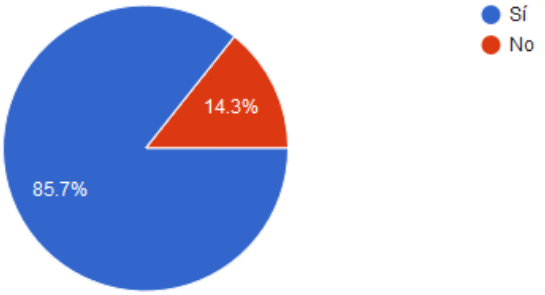

FIGURE 5.13: FORM RESULTS FOR "*DO YOU THINK THAT THE ACTIVITIES PROPOSED ARE ENOUGH?*"

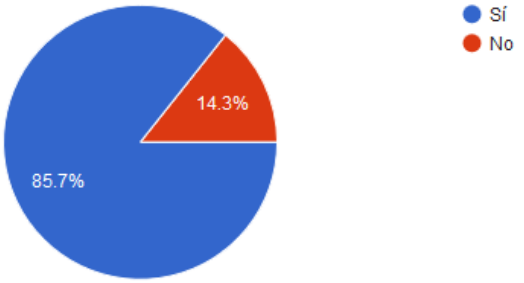

FIGURE 5.14: FORM RESULTS FOR "*DO YOU THING THAT THE TIME TO ACQUIRE DATA WAS ENOUGH?*"

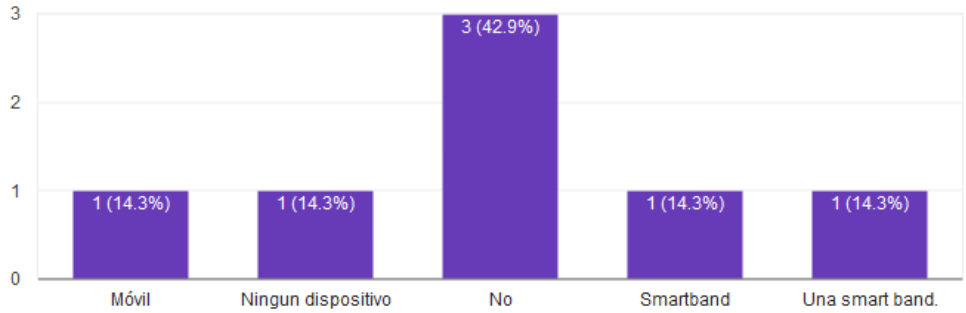

FIGURE 5.15: FORM RESULTS FOR "*WOULD YOU PREFER THE USE OF AN ALTERNATIVE DEVICE OTHER THAN A SMARTWATCH TO ACQUIRE ACCELEROMETER DATA?*"

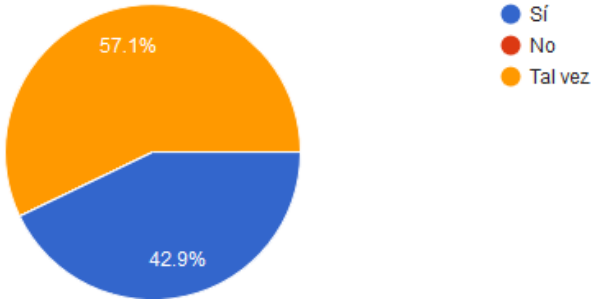

FIGURE 5.16: FORM RESULTS FOR "*DO YOU THINK THAT USER IDENTIFICATION BY ACCELEROMETER DATA IS A VALID ALTERNATIVE TO OTHER FORMS OF BIOMETRIC IDENTIFICATION?*"

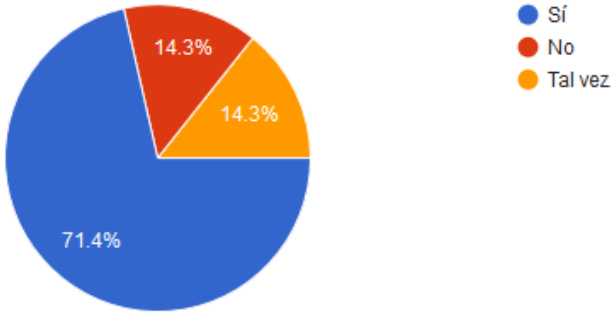

FIGURE 5.17: FORM RESULTS FOR "*WOULD YOU USE A SYSTEM THAT VERIFIED USERS BASED ON ACCELEROMETER DATA?*"

6. MANAGEMENT

This section includes all the information related with the planning of the project, as well as the budget for it, broken down into different concepts and the generated expenses.

To explain the organization project, the first section explains the different tasks that needed to be done and the time that was bound for each of them.

The second section centres on the budged of the project, what costs are expected and the benefits to be obtain from its fulfilment.

Finally, the last section is dedicated to the commercialization of the project, how it would be included in the market.

6.1. PLANNING

The plannification of this project was governed by the delivery dates established by the university. To meet this schedule, the project has been divided into different tasks and each has been asigned a time frame to be completed.

On Figure 6.1 a Gantt diagram is presented with the different tasks, their duration, starting and ending date, and a visual representation of the time that each of them consume.

The first task, documentation, takes place throught all the project, being a labour that needs to be completed throught all the project, stating the necessary information for each part of the project as their correspondent tasks are completed.

The second task, state of the art study, encompasses the period of time dedicated to the study of already existing projects and technology present on the research field of this thesis. It takes one month, from December 1st to December 31st.

The third task, analysis, includes the study of the technologies that are going to be used for the development of this project, comparing them and selecting those who can be better adapted to achieve the proposed goals. It starts when the state of the art study finishes, on January 1st, and ends on February 15th, 46 days after.

The fourth task, design, is destined to the planning of the different components and structures that will form the final system. It starts on Februare 16th and ends on April 4th.

Implementation and experimentation are two tasks that go hand in hand. One is destined to code the final system and the other to test if all the requirements and goals are met. Implementation starts on April 5th and ends on May 20th and experimentation takes 18 days, starting on May 21st and finishing on June 7th.

	Task Name	Duration	Start	ETA
1	Complete project execution	190 days	1/12/16	7/6/17
2	Documentation	190 days	1/12/16	7/6/16
3	State of the Art study	31 days	1/12/16	31/12/16
4	Analysis	46 days	1/1/17	15/2/17
5	Design	48 days	16/2/17	4/4/17
6	Implementation	47 days	5/4/17	20/5/17
7	Experimentation	18 days	21/5/17	7/6/17

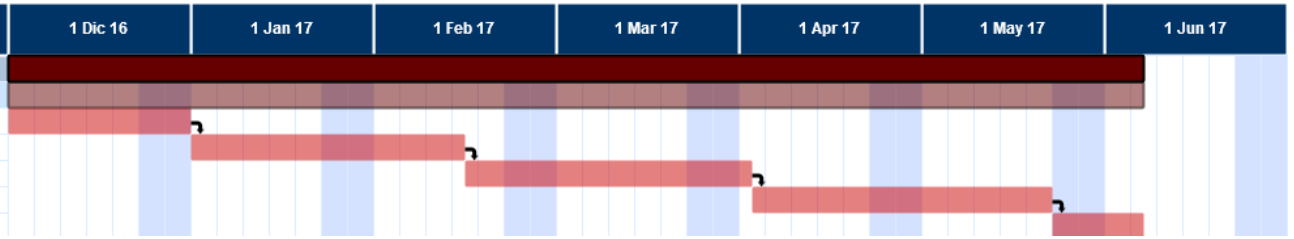

FIGURE 6.1: GANTT DIAGRAM: PROJECT PLANNIFICATION

6.2. BUDGET

The cost for the project is divided into several concepts: direct costs, which encompass personal and equipment costs, software and hardware used to develop the project, and indirect costs, which are fixed on a 20% over directs costs.

On Table 6.1, the first division of the direct costs is presented. It includes the people that would be involved on the project, the hours that they would be working on it and the cost per hour given their position.

Position	Cost per hour	Hours	Cost (€)
Analyst	20	100	2,000
Developer	20	200	4,000
Project manager	30	70	2,100
Tester	15	10	150
		Total:	8,250

TABLE 6.1: DIRECT COST IN PERSONAL

On Table 6.2, the second division of direct costs collects the expenses derived from the material that is needed. On them, the amortization needs to be calculated, using the following formula:

$$Imputable\ cost = \frac{Months\ the\ material\ has\ been\ used}{Life\ span\ of\ the\ material} * Unitary\ cost\ of\ the\ material$$

The project has been developed over seven months.

Concept	Unitary cost (€)	Imputable cost (€)
Desktop computer (8GB RAM, Intel Core i5)	700	102
Laptop computer (4GB RAM, Intel Core I5)	400	117
Texas Instruments eZ430-Chronos	59.38	17
	Total:	236

TABLE 6.2: DIRECT COST ON EQUIPMENT

The total cost of the project is computed after adding the indirect cost, applying a 20% rate over the direct cost that would be destined to electricity, Internet and any other consumable concepts.

The resulting final cost is fixed on 10,215€, as shown in Table 6.3.

Total costs budget	Total cost budget (€)
Personal costs	8,250
Equipment costs	263
Indirect costs (20%)	1,702
Total:	10,215

TABLE 6.3: TOTAL BUDGET FOR THE PROJECT

To obtain the total budget for the proposed system, two more factors need to be considered. The first, risk, is the estimation destined to incidental or unforeseen expenses. The second, benefit, states the amount of money to be made from the project.

Fixing the risk at a 10%, the cost of the project increase to 11,236€. If the benefit is fixed to a 10% as well, then the final budget for the project is 12,357€.

6.3. COMMERCIALIZATION

As mentioned during the Introduction and the State of the art, this thesis presents a proof of concept aimed to establish the basis for future research or systems that want or need to use what is proposed. Thus, the commercialization of the projects comes with the elaboration of future projects on top of this one.

The main goal of the system is to produce a new way of continuous authentication for users, enabling them to unlock a system without the need of performing any additional action. Once unlocked, the users would be able to continue using the system until they decide to lock it again or until the data received from the accelerometer states that the user is not included or is not recognized by the machine learning model.

Even though there are plenty of ways to unlock a phone or a computer, few of them enable this continuous authentication in a non-invasive way, usually people are asked to perform certain tasks or to reintroduce the codes or passwords to continue using the system.

Identification based on accelerometer data can be easily included by mobile phones companies by introducing an accelerometer, if they have not already, and including the machine learning algorithm. This provides users with new ways of securing their data, strengthening their devices against attacks.

7. LEGAL

Over this section, the different legal aspects of the system are talked through and discussed, in order for the project to comply with the current legislation.

The main law that affects and can jeopardize the future of the project is the *Ley Orgánica de Protección de Datos*[1] (Estado, 2016). The General Data Protection Regulation (Union, 2016) also needs to be considered as it is the correspondent European law for *Ley Orgánica de Protección de Datos*.

These laws state that personal data is information related to the user that allows its identification, giving the fundamental right to control personal data and the ability to obtain, use or decide over it.

The main goal of the proposed system is to identify people based on their movement data. As such, the user is entitled to know what will be done with that information and how it will be used. The system does not collect any other information related to the identity of the user apart from the accelerometer data, but as it could imply the identification by a third party, the system is consistent with what is stated on the law.

To avoid user identification by third parties and given that the proposed system is to be used locally, without the interference of cloud systems or third party storing software, all data recorded and used on the system is stored on the device that is being used. The user name associated with their activity will be provided by said user, an identifier so that the system is able to tell the user that the data that is being received belongs to that name.

Given an external intrusion, only the accelerometer data would be obtained without knowing the person it belongs to, as it would only be matched to the label indicated by the user who will be encouraged not to use any personal data such as name and surname, identification number, … that would lead to its identification. Nonetheless, as it offers a service, this law needs to be considered.

[1] In English: Spanish jurisdiction on the protection of data, data protection law.

8. CONCLUSIONS AND FUTURE WORK

The idea for this thesis came from the concept that, nowadays, more biometric systems are being included on small devices, so the first approach to the project was to achieve user authentication based on several sensors' data, mainly an accelerometer and a heart rate monitor or an accelerometer and electrocardiogram.

Between the state of the art study and the analysis of the system, where the sensors and the device that would contain them had to be chosen, the approach to the project changed, focusing on only one sensor instead of two. As electrocardiograms are not quite accessible and heart rate monitors do not allow continuous authentication, and since the heart rate sensor needs to be constantly in contact with the skin to be able to obtain and identify the user, unless they are place on a smartwatch or a smart band; the attention was placed on the accelerometer, an easily accessible sensor that is embedded on most portable devices.

The goal of this project was then divided on two main parts: the first was to check if user authentication based on the way they move is possible, the second was to develop a system that integrated this authentication mode in order to test it and to offer potential users a way to check by themselves how the system works, providing them with a GUI.

Both tasks were developed at the same time, as the GUI implements the authentication system and it is also used to test the results of said system. After pondering over the different algorithms that might be used and how each of them would contribute to the task. Once a basic set of tests where run through the system, the algorithm selection narrowed to two options: J48 and Hoeffding trees, both being supervised classification machine learning algorithms. J48 showed the highest rates of accuracy but Hoeffding trees are able to process data streams and learn from them as data is received, without explicitly generating the model again. After a second batch of tests, it became clear, based on the accuracy and exact match metrics, that J48 exhibited better results than Hoeffding trees, which resulted in J48 being chosen as the model generator even though it is not capable of online analysis. This algorithm was introduced into the GUI system in order to generate the model and update it as users are registered into the system.

The group of volunteers that were asked to test the system were briefed about it, how it worked and what it was used for. They were asked to perform a series of activities: rest, walk, run and jump to be able to train the system, and then test it using the collected data from the performed tasks.

As has been stated over several sections of this document, this thesis is a starting point for all researchers interested on user authentication based on motion, more specifically, on user authentication and action recognition, so that the system can differentiate between classes without the need of a specific attribute which states the activity and sets the correct value by the user. As such, there is more than enough room for future work and improvements over the proposed system.

One of the upgrades that can be done is to add more users than the recommended limit to the system, to test the boundaries of the classification algorithm and get the maximum number of users that can be recognized without a significant drop on the accuracy of the system.

With the addition of more users comes the introduction of more activities. One of the complaints received about the system was how restricted it was in terms of space, as the limit of the transmission was very restricted. Swimming and riding a bicycle were some of the proposed activities.

Changing the smartwatch to one with Bluetooth connection would allow the implementation of the system on a smartphone, where it can be tested on a real-life situation where a user wants to access his phone after exercising.

Finally, as stated at the beginning of this section and probably the most interesting addition that can be done to this project, is to add more sensors to the verification system, so that the output can be more robust as it would be based on more data enabling its authentication.

REFERENCES

A. D., A. H., F. B., C. L., & H. H. (2012). *Touch me once and I know it's you! Implicit Authentication based on Touch Screen Patterns.* Austin, Texas, United States of America.

Accelerometers Information. (n.d.). Retrieved June 2017, from Engineering 360º: http://www.globalspec.com/learnmore/sensors_transducers_detectors/acceleration_vibration_sensing/accelerometers

All About Heart Rate (Pulse). (2015, July). Retrieved June 2017, from American Heart Association: http://www.heart.org/HEARTORG/Conditions/More/MyHeartandStrokeNews/All-About-Heart-Rate-Pulse_UCM_438850_Article.jsp#.WUhAROmLmM8

Azizah Abd Manaf, S. S.-Q. (Ed.). (2011). *Informatics Engineering and Information Science.* Springer. doi:10.1007/978-3-642-25462-8

B. F., S. L., & M. S. (2011). Pulse Oximetry. In A. T. Society, *Patient Education Series* (Vol. 184). Retrieved from https://www.thoracic.org/patients/patient-resources/resources/pulse-oximetry.pdf

Bargal, S. A., A. W., Chan, C. R., S. H., S. S., E. R., . . . C. G. (2015). *Image-based Ear Biometric Smartphone App for Patient Identification in Field Settings.*

Bolle, R. M. (1998). *Biometrics, Personal Identification in Network Society* (Vol. 1). (A. K. Jain, Ed.) Kluwer Academic Publishers Norwell.

Camgal, R. (22 de October de 2013). *Las huellas dactilares.* Recuperado el June de 2017, de Monografias: http://www.monografias.com/trabajos94/las-huellas-dactilares/las-huellas-dactilares.shtml

Clarke, N. &. (2007). Authenticating mobile phone users using keystroke analysis. *International Journal of Information Security, 6*(1), 1-14. doi:10.1007/s10207-006-0006-6

Derawi, M. O., C. N., & P. B. (2010). *Unobtrusive user-authentication on mobile phones using biometric gait recognition.* Darmstadt, Germany: IEEE. doi: 10.1109/IIHMSP.2010.83

Estado, A. E. (2016). *Protección de Datos de Carácter Personal.* (S. J. García, Ed.) Madrid, España. Retrieved June 2017

G. H., A. B., & B. P. (2010). MOA: Massive Online Analysis. *Journal of Machine Learning Research, 11*, pp. 1601-1604. Retrieved from http://www.jmlr.org/papers/v11/bifet10a.html

G. H., J. R., P. R., & B. P. (2016). MEKA: A Multi-label/Multi-target Extension to Weka. *Journal of Machine Learning Research, 17*(21), pp. 1-5. Retrieved from http://jmlr.org/papers/v17/12-164.html

Instruments, T. (2015). *eZ430-Chronos Developement Tool Users' Guide.* Dallas.

Junshuang Yang, Y. L. (2015). MotionAuth: Motion-based Authentication for Wrist Worn Smart Devices. Little Rock, Arkansas, United States of America.

Kaminka, G. A., M. F., P. B., E. H., V. D., F. D., & F. v. (Eds.). (2016). *ECAI 2016, 22nd European Conference on Artificial Intelligence* (Vol. 1). Amsterdam, Berlin, Washington DC: IOS Press. Retrieved from https://books.google.es/books?id=xfboDAAAQBAJ&printsec=frontcover&hl=es#v=onepage&q&f=false

Mohd-Isa, W. N., J. A., J. A., & C. E. (2011). *Recognizing Individual Sib in the Case of Siblings with Gait Biometric.* Malaysia.

Rasmussen, K. B., M. R., I. M., & G. T. (2014). Authentication Using Pulse-Response Biometrics. San Diego, CA, United States of America. Retrieved from http://dx.doi.org/10.14722/ndss.2014.23208

Triggs, R. (2016, December 13). *How fingerprint scanners work: optical, capacitive, and ultrasonic variants explained.* Retrieved June 2017, from Android Authority: http://www.androidauthority.com/how-fingerprint-scanners-work-670934/

Union, O. J. (2016, 5 4). *Regulations.* Retrieved June 2017, from http://ec.europa.eu/justice/data-protection/reform/files/regulation_oj_en.pdf

Y. Z. (2016). Ph.D. Activity-based Implicit Authentication for Wearable Devices. In IEEE, *Information Processing in Sensor Networks.* doi:10.1109/IPSN.2016.7460684

ANNEX

A. GLOSSARY

Random Forest: combination of prediction trees where each tree depends directly on the values of a random vector.

Support Vector Machine: set of supervised learning algorithms related with classification and regression problems.

Dynamic time warping: time series analysis algorithm that weights the similarity between two temporal sequences that can vary in speed.

Hidden Markov Models: statistic models in which a Markov process with unknown parameters is modelled. The objective is to obtain the unknown parameters from observable attributes.

k-NN: supervised machine learning classifier, based on a training and a prototype set, which estimates the density function for each class.

GUI: Graphical User Interface, user interface oriented to the simplification of the use of an electronic device by the users.

Classifier: machine learning algorithms can sort out input data into a category or class.

Clustering: to group a set of objects on different vectors in a way that the objects which share the same vector are more similar between them.

MULAN: multi-label learning tool developed in Java.

Cross-validation: also known as rotation estimation, cross-validation is an evaluation technique to assess the results in order to determine if a statistical analysis is able to generalize. To evaluate the model, input data is divided on k sections or folds and each iteration, k-1 folds are used as training set and the remaining fold as test set. This process is repeated until all folds have been used as the test set, taking a total of k+1 iterations to complete the evaluation.

J48: implementation of the C4.5 algorithm, generating a pruned or unpruned tree.

CC: Classifier Chains, multi-label parsing algorithm, combines Binary Relevance method and label dependencies with classification.

BCC: Bayesian classifier chains, also called Bayesian classifier trees. Generates a maximum spanning tree found on the label dependencies and employs CC classifiers. Originally, the base classifier was Naive Bayes, giving the name to the classifier.

CCp: Multi Target Classifier Chain with a probabilistic output.

CR: Class relevance, it represents a generalization of Binary Relevance (BR) to be used on multi-target problems.

NSR: Nearest Set Replacement. To replace outliers, instead of subsets, the nearest sets are used.

SCC: Super Class Classifier (Super Node Classifier), related to multi-label classificatory RAkELd, they handle into super classes the output space considering label dependence, where multi-target classifiers are applied.