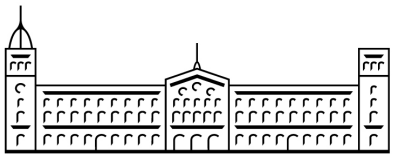


Treball final de grau

GRAU D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques
Universitat de Barcelona

# SLEEPING ACTIVITY RECOGNITION FOR AN INTELLIGENT TELE-MONITORING SYSTEM

Author: Carme Zambrana Seguí

| | |
|---|---|
| Supervisor: | Dra. Petia I. Radeva |
| Completed in: | Departament de Matemàtica Aplicada i Anàlisi |
| Barcelona, | January 28, 2016 |

*Coming up with features is difficult, time-consuming, requires expert knowledge.*
*"Applied machine learning" is basically feature engineering.*

Andrew Ng

# Abstract

People that need assistance, as for instance elderly or disabled people, may be affected by a decline in daily functioning that usually involves the reduction and discontinuity in daily routines, as well as, a worsening in the overall quality of life. Thus, there is the need to intelligent systems able to monitor indoor activities of users to detect emergencies, recognise activities, send notifications, and provide a summary of all the relevant information. In this TFG, a machine learning system is presented, it is aimed at improving the ruled-based system accuracy in detecting whether the user is performing their sleeping activity or not. It has been integrated in a sensor-based telemonitoring and home support system. The data used to build and evaluate the system was obtained from a real-world environment with real end-users, thus ensuring the data reflect the complexities of the real-world.

# Resumen

Las personas que necesitan asistencia, como por ejemplo las personas mayores o con discapacidad, pueden experimentar una disminución de sus actividades diarias, lo que generalmente supone la reducción y/o discontinuidad en sus rutinas, además de un empeoramiento de la calidad de vida. Por consiguiente, existe la necesidad de crear sistemas inteligentes capaces de monitorizar las actividad de los usuarios para detectar emergencias, reconocer las actividades, enviar notificaciones y proporcionar un resumen de toda la información relevante. En este TFG, se presenta un sistema de aprendizaje automático, con el objetivo de mejorar la precisión del sistema basado en reglas. El objetivo del sistema es reconocer si la actividad que está realizando el usuario es la de dormir. Este sistema ha sido integrado en un sistema de soporte en el hogar y de telemonitorizage basado en sensores. Los datos utilizados para construir y evaluar el sistema fueron obtenidos en un entorno real con usuarios finales, esto asegura que los datos reflejen las complejidades del mundo real.

# Resum

Les persones que necessiten assistència, com per exemple les persones grans o amb discapacitat, poden veure afectades les seves capacitats per dur a terme activitats diàries, el que generalment suposa una reducció i/o discontinuïtat en les rutines diàries, a més d'un empitjorament de la qualitat de vida. Per tant, existeix la necessitat de crear sistemes intel·ligents capaços de monitoritzar les activitats dels usuaris per detectar emergències, reconèixer les activitats, enviar notificacions i proporcionar un resum de tota la informació rellevant. En aquest TFG, es presenta un sistema d'aprenentatge automàtic per tal de millorar la precisió del sistema basat en regles. L'objectiu del sistema és reconèixer si l'activitat que està realitzant l'usuari és la de dormir. Aquest sistema ha estat integrat en un sistema de suport a la llar i

de telemonitorització basat en sensors. Les dades utilitzades per a construir i avaluar el sistema van ser obtingudes en un entorn real amb usuaris finals, assegurant que les dades reflecteixen les complexitats del món real.

# Acknowledgements

Foremost, I would like to express my sincere gratitude to my supervisor, Dra. Petia I. Radeva, for her immense knowledge, professional guidance and valuable support. Her advise helped me in the research and writing of this TFG. I could not have imagined having a better mentor.

Besides my supervisor, I would like to thank the Eurecat eHealth department for offering me an internship in their group, which has led me to the opportunity of working on eKauri project. I am particularly grateful to Dra. Eloisa Vargiu and Xavier Rafael for their patient guidance, enthusiastic encouragement and useful critiques of this research work. Their willingness to give their time so generously has been very much appreciated.

I would also like to thank my family: my parents for their support and encouragement throughout my study, Dra. Joana Zambrana and Narcís Puig for being my role model, and Joan Pere Perelló for cheering me up and standing by me through the good and bad times.

Finally, I would also like to extend my thanks to Mariona Condom and Júlia Rodríguez for their daily support, and to my double degree classmates for being there for me during these six years.

# Contents

# Acronyms

**SVM** Suport Vector Machines

**ADLs** Activities of Daily Living

**CRF** Conditional Random Fields

**RF** Random Forest

**$k$-NN** $k$-Nearest Neighbours

**ANN** Artificial Neuronal Networks

**NB** Naive Bayes

**HMM** Hidden Markov Model

**HHMM** Hierarchical Hidden Markov Model

**SB-TMHSS** sensor-based telemonitoring and home support system

**IMS** Intelligent Monitoring System

**IMM** Intelligent Monitoring Modules

**ARM** Activity Recognition Module

**DTs** Decision Trees

**LR** Logistic Regression

**BN** Bayesian Networks

**MCC** Matthews correlation coefficient

**MLS** Machine Learning System

**PIR** passive infrared

**HSMM** Hidden semi-Markov Model

**PP** pre-processing

**ED** emergency detection

**AR** activity recognition

**EN** event notification

**SC** summary computation

**RA** risk advisement

**RBF** radial basis function

**ML** Machine Learning

**PCA** Principal Component Analysis

**SVD** Singular Value Decomposition

**CV** cross-validation

**LOO** Leave-one-out

**ID3** Iterative Dichotomiser 3

**CART** classification and regression tree

**TP** True Positive

**TN** True Negative

**FP** False Positive

**FN** False Negative

**ACC** accuracy

**CVI** Centre de Vida Independent

**IM** intelligent module

# 1 Introduction

## 1.1 Context

Europe is faced with an ageing population, currently 24 per cent of the population is over 60. Furthermore, the population ageing is growing at a rate of 3.26 per cent per year [1]. This phenomenon occurs owing to two principal causes; on the one hand, advances in medicine are helping millions of people live longer and, on the other hand, women have fewer than 2.1 children on average, the level required for full replacement of the population in the long run. The foregoing observations highlight long-term care as a major problem currently, as well as, cognitive impairments and problems associated with ageing. Even, these problems will increase during the foreseeable future.

In elderly care, Activities of Daily Living (ADLs) are used to assess the cognitive and physical capabilities of an elderly person. The ADLs are a useful tool in the study of prognosis and the effects of treatment [2]. Some health problems start with problems in doing daily living activities. On the one hand, the physical signs can arise in two form: the elderly person is not able to complete an activity by themselves or the activity takes more time than usual. On the other hand, forgetting to do some daily activity may indicate diseases that affects memory.

Among other factors, quality of life of people may be influenced by sleep quality [3]. Sleep disorders represent a very common problem, especially in the Western industrialized countries [4]. Epidemiological studies show that the prevalence of sleep disturbances lies between 20% and 30% and increases with age ($1 \pm 3$). Chronic sleep disorders may involve a risk of somatic/psychic diseases.

Various methods, both subjective and objective, of physical activity assessment tools have been developed. Subjective methods, such as diaries, questionnaires and surveys, are inexpensive tools. However, these methods often depend on individual observation and subjective interpretation, which make the assessment results inconsistent [5]. Whereas objective techniques use remote monitoring techniques relying on sensors, such as home-automation, wearable and/or environmental ones [6]. Although this techniques are more expensive than the subjective ones due to the price of the devices used, the results obtained with these techniques are better.

Activity monitoring is an increasingly important research area due to many facts: it can be used as assessment of the ADLs and as a predictor of quality of life; it allows elderly people to be self-reliant longer and it can help with the healthcare, among other. Monitoring users' activities allows therapists, caregivers, and relatives to become aware of user context by acquiring heterogeneous data coming from sensors and other sources. Additionally, activity monitoring provides elaborated and smart knowledge to clinicians, therapists, carers, families, and the patients themselves by inferring user habits and behaviour, as well notifying anomalies in the behaviours of the users.

There has been an increasing social demand for intelligent telemonitoring systems due to the ease with which they help us to manage the problem of the population

ageing and to improve elderly care. This fact makes it necessary to put more emphasis on activity recognition methods. In this TFG a step has been made forward in this direction by using machine learning techniques on sleeping activity recognition.

## 1.2 Related work

A large variety of telemonitoring systems has been proposed in the literature. Sensor-based telemonitoring systems rely on a conjunction of sensors, each one devoted to monitor a specific status, a specific activity or activities related to a specific location. Once all of the data have been recorded it requires a certain degree of intelligence for data processing and identifying if the person requires assistance or an unusual activity has been recognized. These systems combine all kind of sensors and devices with generative and/or discriminative models.

Sensor technology may range from vital signal devices, such as blood pressure monitors, heart rate monitors and devices which can measure body temperature; to sensors which track the person or their activity, like accelerometers, video cameras, presence sensors, door sensors (to detect a door being opened/closed), pressure mat sensors, gas sensors and floating sensors to detect toilet flush.

A lot of discriminative models are used to recognise activities, for instance Conditional Random Fields (CRF) [7], Suport Vector Machines (SVM) [8], [9], [10], Random Forest (RF) [10], $k$-Nearest Neighbours ($k$-NN) [8], and Artificial Neuronal Networks (ANN) [8]. Some approaches use generative models as Naive Bayes (NB) [10], Hidden Markov Model (HMM) [7], [11], [8]. A few systems even use Hybrid generative/discriminative models [8].

This diversity makes it extremely difficult to compare performances and draw conclusive findings from those studies, even so, let us report a number of different papers with different approaches. Due to issues regarding personal privacy, technical installations, and costs of technology the most adopted sensors are anonymous binary sensors. Binary sensors do not have the ability to directly identify people and can only present two possible values as outputs (0 and 1).

An accurate work with activity recognition was done in [7] using temporal probabilistic models: HMM and CRF. The system uses wireless network nodes (RFM DM 1810) and a bluetooth headset combined with speech recognition software to annotate out the starting and the end point of an activity. The system was evaluated by two measures: the timeslice accuracy and class accuracy because the classes are considered to be imbalanced. Although they obtained a high performance (95.6% of timeslice accuracy and 79.4% of class accuracy), these results rely on the accuracy in the annotation process. Moreover, the dataset used was recorded with a 26-year-old man and results evaluating the system with elderly people had not been presented.

The same authors published some years later [11]. They presented a two-layer hierarchical model, Hierarchical Hidden Markov Model (HHMM), to activity recognition where activities consist of a sequence of actions, the top layer represents the

activities and the bottom layer represents the actions. The sensors used were: reed switches to measure whether doors and cupboards was opened or closed; pressure mats to measure sitting on a couch or lying in bed; mercury contacts to detect the movement of objects (e.g. drawers); passive infrared (PIR) to detect motion in a specific area and float sensors to measure the toilet being flushed. The aims of this study was to know the number of action clusters needed for modelling activities; which observation model gives the best performance and compare the performance of HHMM to other common methods used for that purpose like HMM and Hidden semi-Markov Model (HSMM). The results show that the HHMM proposed outperforms the HMM and the HSMM with an increase of 7 percentage points in F1-score

In [8], authors proposed a hybrid approach to recognize ADLs from home environments using a network of binary sensors. They used three types of binary sensors: PIR sensors to detect motion in a specific area; red switches for open/close states of doors and cupboards and float sensors to measure the toilet being flushed. Among the different activities recognized, *sleeping* was one of them. Two hybrid systems were proposed, using an SVM or ANN to estimate the emission probabilities of an HMM. The results show that both hybrid models outperform the other approaches evaluated (SVM, HMM, ANN, Trees, $k$-NN and rules). Among the different schemes evaluated, the SVM/HMM hybrid approach obtains the best performance, a significant 0.7 of F1 score. As authors said, it should be noted that hybridizing their schemes implies a more complex system. Hence, when integrating into a real home monitoring solution, it should be considered whether performance should take priority over efficiency.

In [9] a more complex template SVM was used to automatically recognize 11 different home activities. The proposed technique was integrated in different sliding window strategies (e.g., weighting sensor events, dynamic window lengths, or two levels of window lengths). One of the facets of the work presented in this paper, sets it apart from other related work, is the dataset that is used to evaluate algorithms. They used 6 months of data from 3 different homes, where the subject was living in their natural habitat and conducting their daily routine with no instructions from the researchers. So, the dataset reflects the complexities of unconstrained real-world data.

A recent study [10] evaluated the performance of three classifiers: NB, SVM and RF for recognition of ADLs. The sensor network consists of ten wireless sensor boxes which measure ambient motion, temperature, luminescence, humidity and acceleration of the sensor box itself. The best performance was obtained using RF classifier. The dataset for this study was collected in 10 homes of 10 healthy volunteers over 20 days each. The authors emphasize that an exemplary data obtained with a wearable belt clip have been used.

## 1.3 eKauri

The eHealth team of BDigital, now Eurecat, have been developing a sensor-based telemonitoring and home support system (SB-TMHSS) able to monitor the evolution of the user's daily life activity. The project idea was conceived in BackHome project [1], since then the idea has evolved and now it is considered as a project itself.

The system is intended for elderly people who live alone but need some assistance or monitoring from caregivers, doctors or relatives. Two main requirements have been taken into account: on the one hand, there is a need for assisted living solution to become intelligent and also practical issues such as user acceptance and usability need to be resolved in order to truly assist elderly people [12]; on the other hand, to develop a system to use in real-world scenario, where the subject is living in their natural habitat and conducting their daily routine with no instructions from the researchers, as in [9].

The fact that the main users are elderly people implies some limitations: their purchasing power is limited so the system has to be low-cost; they are not used to using new technologies so that any interaction with this should be as easy as possible; and they do not rely on technology so it must be the least invasive possible, i.e. not have to interact with sensors.

### 1.3.1 Sensor-based telemonitoring and home support system

The SB-TMHSS, shown in Figure 1 comprises four components: home, healthcare center, middleware and Intelligent Monitoring System (IMS).

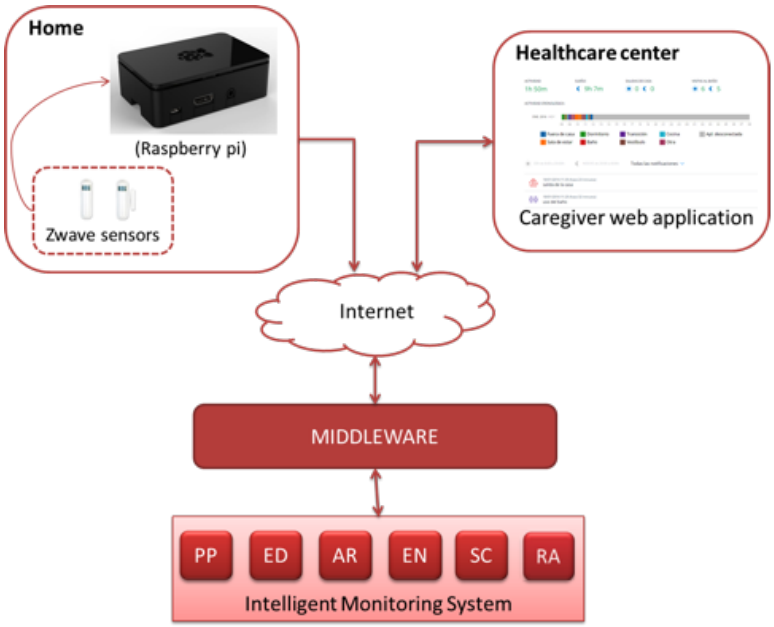



Figure 1: High-level architecture of the SB-TMHSS which include four components: home, healthcare center, middleware and IMS.

[1] www.backhome-fp7.eu

At user's homes a sensor system to detect the activity they perform is installed. All the data are collected by the RasberryPi and stored in the cloud. The middleware, which acts as a Software as a Service, is composed by a secure communication and authentication module; API module to enable the collector transmitting all the data from sensors to make them available to the IMS. The IMS is aimed at continuously analysing and mining the data through 5-dimensions: detection of emergencies, activity recognition, event notifications, summary extraction, and rule triggering. Finally, the healthcare center receives notifications, summaries, statistics, and general information belonging to the users through a web application.

In the following, two of the four components will be explained: home and IMS. All the Intelligent Monitoring Modules (IMM) will be briefly explained emphasising Activity Recognition Module (ARM). Only these components will be detailed because they are directly related to the work presented in this TFG.

### 1.3.2 Home

At home, a set of sensors are installed. In particular, the sensors used are: presence sensors (i.e., TKB TSP01), to identify the room where the user is located (one sensor for each monitored room); a door sensor (i.e., TKB TSM01), to detect when the user enters or exits the premises; electrical power meters and switches, to control leisure activities (e.g., television and pc); and pressure mats (i.e., bed and seat sensors) to measure the time spent in bed (wheelchair). The system is also composed of a network of environmental sensors that measures and monitors environmental variables like temperature, but also potentially dangerous events like gas leak, fire, CO escape and presence of intruders.

The reason for choosing these sensors among others is their value for money, as said the price of the final product should be as low as possible. Moreover, the sensors do not have the ability to directly identify people and they do not need direct interaction to work, so the sensors fulfil all the constraints.

All the adopted sensors are wireless z-wave. They send the retrieved data to a collector (based on Raspberry pi). The Raspberry pi collects all the retrieved data and securely redirects them to the cloud where they will be stored, processed, mined, and analysed. The proposed solution relies on z-wave technology for its efficiency, portability, interoperability, and commercial availability. In fact, contrary to other wireless solutions (e.g., ZigBee), z-wave sensors are able to communicate with any z-wave device. The advantage of a solutions based on Raspberry pi is the fact that it is easy-to-use, cheap, and scalable.

### 1.3.3 Intelligent Monitoring System

The aim of this component is to continuously analyse data and concurrently listen for new data. In order to cope with these objectives, the IMS is composed of the following modules: the pre-processing (PP) module to encode the data for the analysis; the emergency detection (ED) module to notify, for instance, in case of smoke

and gas leakage; the activity recognition (AR) module to identify the location, position, activity- and sleeping-status of the user; the event notification (EN) module to inform when a new event has been detected; the summary computation (SC) module to perform summaries from the data; and the risk advisement (RA) module to notify risks at runtime

**Pre-processing** The goal of PP is to pre-process the data iteratively sending a chunk $c$ to ED according to a sliding window approach. Starting from the overall data streaming, the system sequentially considers a range of time $|t_i - t_{i+1}|$ between a sensor measure $s_i$ at time $t_i$ and the subsequent measure $s_{i+1}$ at time $t_{i+1}$. Thus, the output of PP is a window $c$ from $t_s$ to $t_a$, where $t_s$ is the starting time of a given period (e.g., 8:00 a.m.) and $t_a$ is the actual time. Thus, each chunk is composed of a sequence of sensor measures $s$; where $s$ is a triple $< ID, v, t >$, i.e., the sensor ID, its value and the time in which a change in the sensor status is measured. Figure 2 shows an example of a chunk composed by four sensors measures.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 195 | 24.10 | 2014-02-24 10:21:54 | 195 | 24.10 | 2014-02-24 10:30:04 | 177 | 100 | 2014-02-24 10:31:55 | 195 | 24.10 | 2014-02-24 10:34:54 |

Figure 2: Example of a chunk composed by four sensors measures.

**Emergency Detection** ED module aims to detect and inform about emergency situations for the end-users and sensor-based system critical failures. Regarding the critical situations for the end-users, an emergency is sent when specific values appear on $c$ (e.g.; gas sensor ID, smoke sensor ID). Regarding the system failures, ED is able to detect when the end-user's home is disconnected from the middleware as well as a malfunctioning of a sensor (e.g., low battery). The former is implemented by a keepalive mechanism in the Raspberry pi. If no signals are received from the Raspberry pi after a given threshold, an emergency is sent. The latter is implemented by using a multivariate gaussian distributions of sensor measurements on $c$. If the corresponding total number of measures is greater than a given threshold, an emergency is sent. Each emergency is a pair $< s_i, l_{\varepsilon i} >$ composed of the sensor measure $s_i$ and the corresponding label $l_{\varepsilon i}$ that indicates the corresponding emergency (e.g., fire, smoke). Once the ED finishes the analysis of $c$, the list of emergencies $\varepsilon$ is sent to the middleware, whereas $c$, filtered from the critical situations, is sent to AR.

**Activity Recognition** Module's aim is to recognise the activities performed by the user. To achieve this goal the first steps are: recognize if the user is at home or away and if s/he is alone; the room in which the user is (*no-room* in case s/he is away, *transition* in case s/he moving from a room to another); the activity status (i.e., *active* or *inactive*); and the sleeping status (i.e., *awake* or *asleep*). Thus, the output of AR is a triple $< t_s, t_e, l >$, where $t_s$ and $t_e$ are the time in which the activity has started and finished, respectively, and $l$ is a list of four labels that indicates: the localization (i.e., *home*, *away*, or *visits*), the position (i.e., the *room*, *no-room*, or *transition*), the activity status (i.e., *active* or *inactive*), and the sleeping status (i.e., *awake* or *asleep*). To give an example, let us consider Figure 3 where the same chunk of Figure 2 has been processed by AR. This module will be explained

more accurately in the next subsection.


Figure 3: Example of a chunk after the AR processing.

**Event Notification** Beside activity recognition the system is able to detect important events and notify them. In particular, the EN module detects the following kinds of events: leaving the home, going back home, receiving a visit, remaining alone after a visit, going to the bathroom, coming out of the bathroom, going to sleep, and waking up. Each event is defined by a pair $< t_i, l >$ corresponding to the time $t_i$ in which the event happens together with a label $l$ that indicates the kind of event. These events allow us to study of activity degradation as well as improvement/worsening of the overall quality of life. Following the example in Figure 2, Figure 4 shows an event.


Figure 4: Example of an event notified by EN module.

**Summary Computation** Once all the activities and events have been classified, measures aimed at representing the summary of the user's monitoring during a given period are performed. In particular, two kinds of summary are provided: historical and actual. As to the historical summary, it is interesting to have a list of the activities performed during: the morning (i.e., from 8 a.m. to 8 p.m.); the night (i.e. from 8 p.m. to 8 a.m.); all day; the week (from Monday morning to Sunday night) and the month. In particular, sleeping time; time spent outdoors; time spent indoors; time spent performing indoor activities; time spent performing outdoor activities; number of times spent in each room; and number of times that the user leaves the house are computed. As for the actual summary, the room in which the user is; whether the user is at home, or not; the number of times that s/he leaves the home; sleeping time; activity time; and number of visits per room are monitored.

**Risk Advisement** RA is aimed at advising therapists about one or more risky situations before they happen. The module executes the corresponding rules at runtime according to the sequence of sensor measures coming from the PP as well as the summary provided by the SC. Through the healthcare center, therapists have access to an ad-hoc user interface to define the rules corresponding to risks. Those rules are automatically coded in a suitable language, namely ATML [13], and then translated in DRL by the Rule Builder and stored in the Knowledge-Based of Rules (KBR). RA continuously processes data coming from the other modules of the IMM and acts according to the defined rules in KBR. In particular, it analyses the entire chunk $c$ from PP, the list of activities $a$ from AR, and the complete summary $\Sigma$ from SC. The actions $a_r$ triggered by AR are sent to the middleware which is in

charge of actuate in consequence is sent the corresponding advise to the healthcare center. To implement the AR we relied on Drools, a rules management system that provides a rule execution server, and a web authoring and rules management application. A rule is a quadruple $< i, v, o, a_r >$, where $i$ is the item that has to be verified (e.g., a room, the number of slept hours) according to a given value $v$ (e.g., bedroom, 4 slept hours); $o$ is the logic operator (i.e., and, or, not) and a *null* operator in case there is only one term; and $a$ is the action to be performed (i.e., send a notification, an alarm, or an email).

As a final remark, let us note that all emergencies, activities, notifications, and summaries are stored in a database to be available to all the involved actors.

### 1.3.4 Activity Recognition

Instead of a multiclass classifier with many classes as activities, our approach consists of several classifiers, each one specializing in relevant information (localization, position, activity status and sleeping status) to recognise the activities. This approach is chosen due to the fact that the data come from motion sensors (one per room) and a reed switch in the main door. The next step will be to decide which activity is performed by the user using the information obtained with the classifiers.

**The localization** [14] To recognize if the user is at home or away and if s/he is alone, we implemented a solution based on machine learning techniques. The adopted solution is a hierarchical classifier composed of two levels: the upper is aimed at recognizing if the user is at home or not, whereas the lower is aimed at recognizing if the user is really alone or if s/he received some visits. The input data of the classifier at the lower level are those that have been filtered by the upper level, being recognized as positives. The best results for the upper classifier was obtained with a SVM($\gamma = 1.0,\ C = 0.452$). The classifier obtained a F1 score of 0.97 and an accuracy of 0.968. However, to build the classifier at the lower level, we rely on the novelty detection approach because in this case the data were skewed, we have lower data with visits than without. We use a SVM classifier (one-class SVM with radial basis function (RBF), non linear), the best parametrization was $\nu = 0.01$ and $\gamma = 0.1$, obtaining an accuracy of 0.94. The overall hierarchical approach obtained a F1 score of 0.92 and an accuracy of 0.95.

**The position** The room in which the user is, is also stored, that is to say, the room where the sensor is triggered. The label *no-room* is used to indicate that s/he is away and the label *transition* when s/he is moving from one room to another.

**The activity status** To measure the activity status, we rely on the home automation sensors. By default, the status is *active* when s/he is away (the corresponding positions are saved as $no - room$). On the contrary, when the user is at home, AR recognizes s/he as *inactive* if the sensor measures at time $t_i$ that user is in a given room $r$ and the following sensor measure is given at time $t_{i+1}$ and the user was in the same room, with $t_{i+1} - t_i$ greater than a given threshold $\Theta$. Otherwise, the system classified the user as *active*.

**The sleeping status** The first approach to decide if the user is sleeping or not

was a ruled-based system. The system classifies as a sleeping activity those which the user is in the bedroom; the activity is performed at night, which is defined as the period between 8 pm to 8 am; the user is inactive and the activity's duration is more than half an hour. The work presented in this TFG improves this ruled-based system using machine learning techniques.

## 1.4 Aim of the TFG

The main aim of sleep recognition, in our system, is to detect sleeping activity. We do not pretend to detect when the user falls asleep, owing to the limitations of the adopted sensors. In this project, the sleeping activity is defined as a period which begins when the user goes to sleep and ends when the user wakes up in the morning.

The specific requirements of the sleep recognition are to report the following information: (1) the time when the user went to sleep and woke up, henceforth known as "*go to sleep time*" and "*wake up time*", respectively; (2) the number of sleeping activity hours; and (3) the number of resting hours, which are sleeping activity hours minus the time that the user spent going to the toilet or performing other activities during the night. Moreover, some rules given by the therapists through the healthcare center (see the module RA) may involve the sleeping activity, e.g. the therapists may want to know how many times the user goes to the toilet during their sleeping activity.

As explained above, a ruled-based system was designed as a first approach. Unfortunately, experimenting that approach some issues arose. Firstly, the ruled-based system wrongly classifies some night periods where the user moves in the bed during the night. Secondly, the ruled-based system assumes all users wake up before 8 am, which is a strong assumption that may not be true. Thirdly, the ruled-based system is not accurate because it can not distinguish if the user is in the bedroom watching TV or reading a book, thus classifying them as sleeping.

The aim of this TFG is to build a Machine Learning System (MLS) to detect when the user is performing their sleeping activity. Two sub-goals of the TFG are: the MLS's results improve the results obtained with the ruled-based system and the MLS is able to to report the "*go to sleep time*", the "*wake up time*", the number of sleeping activity hours and the number of resting hours, as the requirements specify. An advantage of using Machine Learning techniques is that the system learns from the dataset, whereas the rules of the ruled-based system are made a priori.

In order to achieve these objectives, a supervised binary classifier is proposed to classify the periods between two bedroom motions in two classes, *sleep* and *awake*. Let us note that *awake* corresponds to the period in which the user goes to another room; performs activities in the bedroom; or stays in the bedroom with the light switched on. Otherwise, the activity is *sleep*.

The rest of the TFG is organized as follows. The methodology used for preprocessing the data, the machine learning techniques and the evaluation methods are explained in chapter 2. Chapter 3 collects the experimental setup, including: the configuration of the premises where the data have been collected; the cleaning pro-

cess; how we obtained the ground truth, and the feature extraction and selection. The results are presented in chapter 4. Chapter 5 ends the TFG with conclusions and future work.

# 2 Methodology

Machine Learning (ML) is a subfield of computer science that evolved from the study of pattern recognition and computational learning theory in artificial intelligence. The ML's aim is to create programs that can generalize behaviours from unstructured information supplied as examples. ML is usually divided in two main types, predictive or supervised learning approach and descriptive or unsupervised learning approach. A third type exists, called reinforcement learning, which is commonly used in many other disciplines, such as game theory and multi-agent systems.

Many factors affect the success of ML on a given task. On the one hand, one of the keys is to decide with approach adapts to the task and to choose the best algorithm, between of all existing algorithms, according to the specific problem. Nevertheless, other questions are as important as this one. First and foremost, a good representation and quality of the data are crucial. A poor performance may come from data with noise or a lot of irrelevant and redundant information present. Moreover, the results should be presented with the right metrics, especially with skewed data, due to the fact that some metrics may hide a bad performance of the minority class.

This chapter is divided in these three main factors: data pre-processing, supervised learning and evaluation techniques. All three sections are intended to give an overview, focusing on the techniques studied and used to do this work. Data pre-processing will explain the procedures of normalisation, feature extraction and future selection, which are used to obtain an accurate data. In the next section, titled Supervised Learning, this type of learning will be explained, as well as, some of its specific models. This section also contains an overview of model selection and model fitting. Evaluation metrics ends this chapter describing the main metrics used to evaluate models of binary classification.

Before starting, two main concepts have to be explained due to the fact that they are used in all ML contexts, they are underfitting and overfitting. Both are related to variance and bias, the former occurs when the model has low variance but high bias, which means that the model does not fit the data well enough, and the latter when the model has low bias but high variance, which means the model fits the data too well.

## 2.1 Data pre-processing

According to [15], data pre-processing may be divided in data cleaning, normalisation, transformation, feature extraction and feature selection. As regards the cleaning process, general methods will not be presented, but in section 3.2 the specific filtering pipeline used in this project will be explained. This section is divided in Normalisation, Feature Extraction and Feature Selection. Normalisation explains this concept and the difference between it and the two related concepts: standardisation and scaling. In addition, it provides a method for each one. Finally, expose in which case should be used each one. Feature Extraction and Feature Selection have

the same purposes, which are to set forth their aims, enumerate the main methods and, briefly explain the methods used in this TFG.

### 2.1.1 Normalisation

Normalisation is a scaling transformation, the same as standardisation. In some occasions these three terms (normalise, scale and standardise) are confused or used wrongly. This shows a need to be explicit about exactly what is meant by each one. Although all three are linear transformations, the purpose of the transformation is different. Whereas scaling it meant to change the range, standardisation converts all the data to a common scale with an average of zero and standard deviation of one. In this context, normalising is understood as a scaling method with fixed boundaries, 0 and 1.

These three procedures can be applied to raw data, usually called variables, or to features. In this section, the term variable will be used, but all methods presented below may also be performed with features.

The min-max normalization, equation (1) rescales each variable of the dataset to lie between a given minimum and maximum value. In equation (1) $min'$ and $max'$ represent the given minimum and maximum, $\nu = (\nu_1, ..., \nu_n)$ represents the values of the variable in the original dataset and $\nu' = (\nu'_1, ..., \nu'_n)$ represent the new data normalized.

$$\nu'_i = \frac{\nu_i - min(\nu)}{max(\nu) - min(\nu)}(max' - min') + min' \tag{1}$$

$$= \frac{\nu_i - min(\nu)}{max(\nu) - min(\nu)} \tag{2}$$

If the minimum and the maximum of the new range are 0 and 1 respectively, the equation (1) is reduced to the equation (2). Using this equation the data may be normalized, allowing the variables to have different means and standard deviations but equal ranges.

The most commonly used method to standardise the data are Z-score, equation (3). In equation (3) $\nu$ is the old value, $\nu'$ the new one, $\mu$ is the mean and $\sigma$ is the standard deviation, both of the old data, shows Z-score. It is the most commonly used method to standardise the data.

$$\nu' = \frac{\nu - \mu}{\sigma} \tag{3}$$

Which method to use depends on each case. On the one hand, if a variable has a variance that is orders of magnitude larger than others, it might dominate the objective function and make the estimator unable to learn from other variables correctly. This problem can be solved scaling the data to the same range. On the other hand, standardization of a dataset is a common requirement of many machine

learning estimators: they might behave badly if the individual variable does not, more or less, look like standard normally distributed data. For instance, such as the RBF kernel of SVM, as will be explained in section 2.2.2, or in Principal Component Analysis (PCA) as will be explained in the next section.

### 2.1.2 Features Extraction

The design of appropriate data representation is an important part of ML. Creating the appropriate set of features is extremely important since the feature set is the only source of information for any learning algorithm. Moreover, as is noted in [16], better performance is often achieved using features derived from the original input.

The features extraction procedures may be divided into two sub-groups, depending on the manner in which the features are obtained. Firstly, there are a number of generic feature extraction methods, such as clustering, PCA, Linear Discriminant Analysis (LDA), Fourier and Hadamard transforms, wavelet transforms or convolutions of kernels. Secondly, the features may be built using the domain knowledge to create specifics features for the application. Below, PCA is briefly reported, this method is also used as a dimensionality reduction technique in order to visualize data with large dimensionality.

PCA is based on an orthogonal linear transformation which can be viewed as a change of basis, the vectors which form the new basis are known as the principal components. The transformation is defined in such a way that the first principal components has the largest possible variance, and each other component has the highest variance possible under the constraint that it is orthogonal to the preceding components.

A well-known method used to calculate the PCA from a data matrix is the Singular Value Decomposition (SVD), due to the fact that efficient algorithms exist to calculate it. A standardization of the data are required to use this method [17], it can be done with the Z-score method explained. The main idea of SVD is to factorize the data matrix in three different matrices, as shown in equation (4), where $M$ is a $m \times n$ matrix with the features, $U$ is a $m \times m$ unitary matrix with the left singular values, as its columns, $\Sigma$ is a $m \times n$ rectangular diagonal matrix with the singular values and $V$ is a $n \times n$ unitary matrix with the right singular values, as its columns. Note that the left-singular and the right-singular vectors of $M$ are eigenvectors of $MM^\top$ and $M^\top M$, respectively. The non-zero singular values of $M$ (found in the diagonal entries of $\Sigma$) are the square roots of the non-zero eigenvalues of both $M^\top M$ and $MM^\top$.

$$M = U\Sigma V^\top \tag{4}$$

The projection of the values in the new space can be done with the simplest matrix multiplication $MV$ which is equivalent, using the equation (4), to $U\Sigma$ whose computational cost is lower, due to the fact that $\Sigma$ is rectangular diagonal. The new values are obtained as a linear combination of the originals ones, in that way

PCA may be used to construct a new feature as a combination of data with a large variance.

### 2.1.3 Features Selection

Acording to [16], the objective of feature selection is three-fold: improving the performance of the predictors, providing faster and more cost-effective predictors and providing a better understanding of the underlying process that generated the data.

Feature selection is the process of identifying and removing as much irrelevant and redundant features as possible. The method used in this process may be divided in two general groups: ranking and subset selection. The first one relies on the predictive power of the feature by itself, while the second is based on the idea of features which are useless by themselves but can be useful together. As far as which group of methods is better, depends on the authors. On the one hand, some authors, [18], [19], defend ranking methods because of its computational and statical scalability. On the other hand, [20], [21], have shown the importance of selecting features as a set, rather than selecting the best features individually.

As regards ranking selection, its aim is to select the features with higher scores which are computed by a scoring function. There are many ranking selection methods as ways to calculate the score. One of these methods are a briefly reported below.

The Pearson's correlation is a measure of the linear correlation between two variables and it is defined as in equation (5), where $cov(x, y)$ means covariance and $var(x)$ means variance, both defined as in equation (6), where the bar notation stands for the mean. Its value ranges from $+1$ to $-1$, both inclusive, where positive values mean positive correlation, 0 means no correlation, and negative values mean negative correlation.

$$corr(x, y) = \frac{cov(x, y)}{\sqrt{var(x)var(y)}} \tag{5}$$

$$= \frac{\sum_{i=1}^{m}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{m}(x_i - \overline{x})^2 \sum_{i=1}^{m}(y_i - \overline{y})^2}} \tag{6}$$

Correlation may be used as a scoring function based on the correlation between a feature and the class, and considering that a good feature is one that is highly correlated with the class. Moreover, the correlation measure can be used to identify redundant features, such as the ones with high correlation between them.

With respect to subset selection, the methods may be divided in three groups: wrappers, filters and embedded methods. Wrappers utilise the machine learning algorithms as a black box to score subsets of variables according to their predictive power. These methods are simple, but often criticized because they are seen as *brute force* methods. However, greedy search strategies seem to be particularly

computationally advantageous and robust against overfitting. Filters select subsets of variables as a pre-processing step, independently of the chosen predictor. The main difference between filter and the other two methods is that it provides generic selection, not tuned for/by given learning machine. Algorithms like Markov blanket algorithms [22], or lineal $\ell_1$-norm SVM [23], are used as filters. Embedded methods perform variable selection in the process of training and are usually specific to given learning machines. A well-known embedded method is the Decision Trees (DTs), which will be explained in section 2.2.4, due to the fact that it is used in RF method. Although filters are faster, recently proposed efficient embedded methods are competitive in that respect.

In most real-world situations, it is not known what the best set of features is, nor the number ($n$) of features in such a set, as is noted in [24]. Currently, there are no means to obtain the value of $n$, which depends partially on the objective of interest. Even assuming that $n$ is known, it is extremely difficult to obtain the best set of $n$ features since not all $n$ of these features may be present in the data comprising the available set of features.

## 2.2 Supervised Learning

A few interesting supervised algorithms and the method to choose one over the other will be discussed in this section. Let us introduce first the main concept of supervised learning.

The goal, in supervised learning approaches, is to learn from a set, which is called training set and consist of input-output pairs $D = \{(x_i, y_i)\}_{i=1}^{N}$, where $N$ is the number of training examples, to create a mapping from inputs $x$ to outputs $y$. Each training input $x_i$ is a $n$-dimensional vector of numbers, where $n$ is the number of features. In other words, the problem can be formalized as function approximation, assuming $y = f(x)$ for some unknown function $f$, and the goal of learning is to estimate the function $f$ given a labelled training set, and then to make predictions using $\widehat{y} = \widehat{f}(x)$, where the hat symbol denotes an estimation. The main goal is to make predictions on new inputs, meaning ones that have not been used to approximate the function. The problems which can be solved with a supervised learning approach may be classified, in terms of the form of the output, into classification, when $y_i$ is a categorical or nominal variable from some finite set, $y_i \in \{1, ..., C\}$ or regression problems, when $y_i$ is a real-valued scalar.

As explained earlier, the goal in classification is to learn a mapping from inputs $x$ to outputs $y$, where $y \in \{1, ..., C\}$, in which $C$ is the number of classes. Classification problems may be classified, depending on the value of $C$, into binary classification problems, when $C = 2$, or multiclass classification problems, when $C > 2$. In the former, it is often assumed that $y \in \{0, 1\}$. The class labels may not be mutually exclusive, this case is best viewed as predicting multiple related binary class labels.

Due to the fact that our outputs are *awake* or *sleep*, our problem is focused as a binary classification problem. Some of the methods commonly used for binary classification are: DTs, RF, Bayesian Networks (BN), SVM, Neural Networks, $k$-NN

and Logistic Regression (LR). Among these methods only three of them: SVM, RF and $k$-NN will be used, therefore, these three will be deeply explained in this section.

Much of machine learning is concerned with devising different models, and algorithms to fit different problems, but each classifier is best in only a select domain, according to the *No free lunch theorems* [25]. These theorems demonstrate that if an algorithm performs well on a certain class of problems then it necessarily pays for that with degraded performance on the set of all remaining problems. So the success of the classifier is based upon the number of observations, the dimensionality of the feature vector, the noise in the data and many other factors.

### 2.2.1 Model Selection and Model Fitting

It is as important to choose a good classifier (model selection) as to choose a good parametrization from a given set (model fitting). As a consequence of the *No free lunch theorem* [25], the best method for a particular problem should be chosen empirically. This empirical search should be performed carefully, otherwise the chosen classifier may not generalize well. Fortunately, methods such as cross-validation exist.

Below the considerations to perform model selection and fitting will be set forth. Then the reasons why it is necessary to use cross-validation and the method itself will be explained. Finally, two useful plots, validation curve and the learning curve, will be presented. The first one may be used in order to visualize model fitting. The aim of the second one is to visualize the performance of the classifier changing the amount of training samples.

The first step is to split the data in two sets, training and testing, the split is often performed so that 80% of the data go to the training set, and 20% to the testing set. Nevertheless, the testing set should not be used for model fitting or model selection, otherwise the method will obtain an unrealistically optimistic estimate of performance. As said in [26], this is one of the *golden rules* of machine learning research. Instead of using the testing set, the training set may be split in two sets: the part used for training the model, and a second part, called the validation set, used to evaluate the model. According to the performance, the best model will be picked and evaluated on the testing set, to estimate its generality.

However, by partitioning the available data into three sets, the number of samples which can be used for learning is drastically reduced, and the results can depend on a particular random choice for the pair of (train, validation) sets. A solution to this problem is a procedure called cross-validation (CV). In the basic approach, called $k$-fold CV, the training set is split into $k$ smaller sets. For each of the $k$ folds, the model is trained using $k-1$ of the folds as training data and then the resulting model is validated on the remaining part of the data (i.e., it is used as a validation set). The performance measure reported by $k$-fold CV is the average of the values computed in each fold. A special cross-validation method is when $k = N$, recalling that $N$ is the number of training samples. The method is called Leave-one-out (LOO) and each learning set is created by taking all the samples except

one, the validation set being the sample left out. Comparing LOO with $k$-fold CV the former builds $N$ models from $N$ samples instead of $k$ models, as the latter does, so when $N \gg k$, then LOO is computationally more expensive than the $k$-fold CV. As a general rule, empirical evidence suggest to use 5- or 10- fold CV.

Regarding model fitting, to plot the performance of a single parameter on the training score and the validation score may be useful to visualize whether the classifier is overfitting or underfitting for some parameter values. This graph is called validation curve and it should be interpreted as follows. The classifier will be underfitting when the training score and the validation score are both low and overfitting when the training score is high and the validation score is low. Otherwise, the classifier is working very well. A low training score and a high validation score is usually not possible.

A similar useful plot, the learning curve, may be used to find out whether adding more training data will improve the performance. It shows the validation and training score of a classifier for varying the numbers of training samples. There are two possible scenarios: the validation score and the training score converge in which case add more data to the training set will not beneficial; and the training score is much greater than the validation score for the maximum number of training samples, which means that adding more training samples will most likely increase generalization.

In these two plots a scoring function is needed to validate the models, as well as in the cross-validation method. The same scoring function can be used to evaluate the performance of the model on the testing set. A few scoring functions will be presented in the section 2.3

### 2.2.2 Support Vector Machine

SVMs were developed by Cortes and Vapnik [27] for binary classification. The machine conceptually implements the following idea: input vectors are non-linear mapped to a very high-dimension feature space and, in this feature space, a linear decision surface is constructed. The huge dimensionality of the new feature space causes two problems: one conceptual and one technical. Following, the two problems are presented, as well as the idea of the solutions presented in [27], in order to better understand the SVM.

The conceptual problem is how to find a separating hyperplane that will generalize well. It was shown that if the training vectors are separated without errors by an optimal hyperplane the expectation value of the probability of committing an error on a test example is bound by the ratio between the expectation value of the number of support vectors and the number of training vectors, as in equation (7). Here an optimal hyperplane is defined as the linear decision function with maximal margin between the vectors of the two classes. This margin may be determined with a small amount of the training data, the so called support vectors. Figure 5 shows an example of a separable problem in a 2 dimensional space; the support vectors are encircled and the optimal hyperplane is plotted in blue.

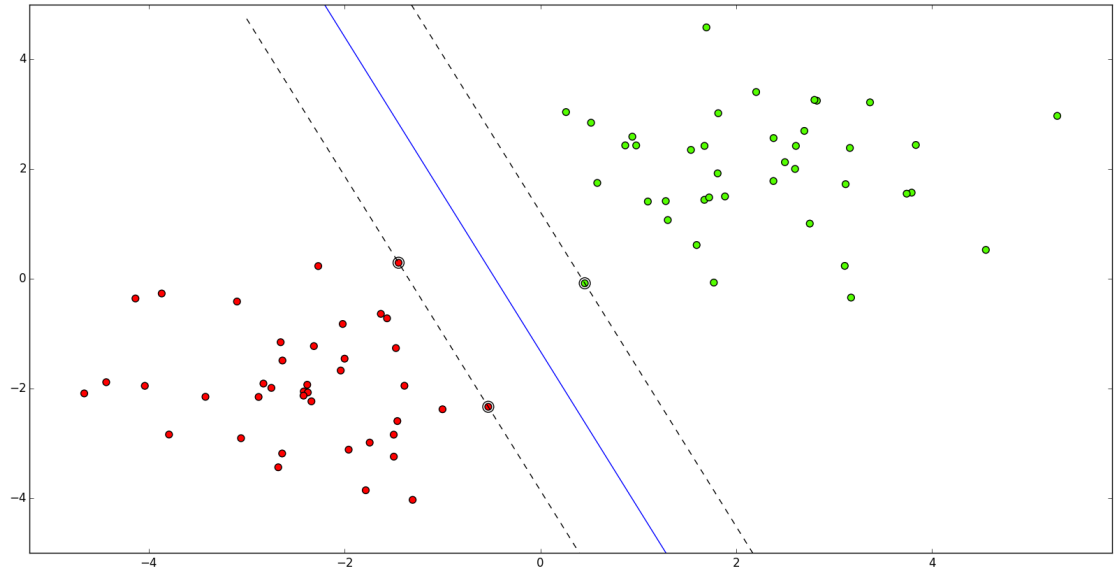



Figure 5: Example of a separable problem in a 2 dimensional space. The support vectors are encircled and the optimal hyperplane is plotted in blue.

As shown in equation (7), this bound does not explicitly contain the dimensionality of the space of separation, so the optimal hyperplane will generalize if it can be constructed from a small number of support vectors relative to the training set size.

$$E[P(Err)] \leq \frac{E[\#\{\text{support vectors}\}]}{\#\{\text{training vectors}\}} \tag{7}$$

As regards the technical problem, how computationally to treat such high-dimensional spaces, it can be solved interchanging the order of operations for constructing a decision function: instead of making a non-linear transformation of the input vectors followed by dot-products with support vectors in feature space, one can first compare two vectors in input space, and then make a non-linear transformation of the value of the result. Thereby, the decision surface is given by the equation (8), where $x_i$ is the image of a support vector in input space and $\alpha_i$ is the weight of a support vector in the feature space.

$$f(x) = \sum_{i=1}^{l} y_i \alpha_i \kappa(x, x_i) \tag{8}$$

Using different dot-products $\kappa(u, v)$, known as kernel, different learning machines with arbitrary types of decision surfaces can be constructed. The kernel can be any function that satisfy Mercer's condition, for example the ones presented in the equations (9), (10), (11) and (12). In some implementations the parameter $\sigma^2$ from $\kappa_2$ is given as its inverse, $\gamma = \frac{1}{\sigma^2}$.

$$\text{Linear } \kappa_0(u,v) = (u \cdot v) \tag{9}$$

$$\text{Polynomial } \kappa_1(u,v) = ((u \cdot v) + \Theta)^d \tag{10}$$

$$\text{Gaussian or RBF } \kappa_2(u,v) = exp\left(-\frac{||u-v||^2}{\sigma^2}\right) \tag{11}$$

$$\text{Sigmoidal } \kappa_3(u,v) = tanh(\kappa(u \cdot v) + \Theta) \tag{12}$$

When the training data cannot be separated without error, the aim of SVM is to find a hyperplane which minimize the number of misclassified samples. This idea can be expressed formally as the minimization of (13), where $w$ is the hyperplane, $C$ is a regularization parameter and $\{\xi_i\}_{i=1}^{l}$ are slack variables. This equation is subject to the constraints (14), (15).

$$\frac{1}{2}||w||^2 + C\sum_{i=1}^{l}\xi_i \tag{13}$$

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i, \qquad i = 1, \cdots, l \tag{14}$$

$$\xi_i \geq 0, \qquad i = 1, \cdots, l \tag{15}$$

The regularization parameter can be used to weigh the misclassification. For large values of C, the margin hyperplane goes smaller, trying to correctly classify as many points as possible. Conversely, with a very small value of C, the margin hyperplane goes larger, even if that hyperplane misclassifies more points.

Several extensions have been developed using the idea of SVM, three of them are enumerated below. The $\nu$-classification allows more control over the number of support vectors by specifying an additional parameter $\nu$, which approximates the fraction of support vectors. The One-class-classification, which is used for outlier/novelty detection, tries to find the support of a distribution. The Multi-class classification, which allows the use of SVM for multi-class classification, uses the one-against-one technique by fitting all binary subclassifiers and finding the correct class by a voting mechanism.

### 2.2.3 K Nearest Neighbour

The k-NN is a simple non-parametric algorithm, which can be used in both, classification and regression. There may be some differences if the algorithm is implemented in order to be used for classification or regression, henceforth reference will be made to the classification algorithm.

The principle behind this algorithm is to predict the class of the new instance using the $k$ closest training examples in the feature space. Therefore, the predicted class is calculated with its $k$ neighbours classes. It can be the most common or weight can be added to the contributions of the neighbours, so that the nearer

neighbours contribute more to the average than the more distant ones. For example, a common weighting scheme consists of giving each neighbour a weight of 1/d, where d is the distance to the neighbour.

The number ($k$) of neighbours, should be a positive integer given as a parameter. In binary classification, when the neighbours contribute to the same weight, the $k$ should be an odd number, avoiding possible ties. The optimal choice of the value $k$ is highly data-dependent: in general a larger $k$ suppresses the effects of noise, but makes the classification boundaries less distinct. Another possible way to pick the neighbours is radius-based, here the parameter is a floating-point which is used as a radius. The neighbours picked are those which are inside the circle, with the center being the new sample and the radius the parameter given. This option is helpful when data are not uniformly sampled.

The $k$-neighbours classification is the more commonly used of the two techniques. When this method is used a distance should be defined to decide which neighbour is picked up. This distance can, in general, be any metric measure, standard Euclidean distance is the most common. The $k$-NN algorithm is often implemented with the Minkowski metric, see equation (16), the advantage is that when $p = 1$ this metric is equivalent to Manhattan distance and when $p = 2$ is equivalent to Euclidean distance, thus changing the parameter $p$ different metrics are obtained.

$$L_p = \left( \sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p} \tag{16}$$

Figure 6 shows a plotted set where the sample in yellow has to be classified. The Nearest Neighbours are enumerated in order of proximity, using Euclidean Distance. In this example, if $k = 3$, the predicted class will be red, but taking into account 5 neighbours the predicted class will be green. As is said if we take $k$ as an even number, for example 2 or 4, the number of neighbour in each class is the same, so the classifier could not predict any class.

Despite its simplicity, fast computation of nearest neighbours is an active area of research in ML. Brute force, KD tree and Ball tree, among other, can be used to search the nearest neighbours. By Brute force we mean the algorithm which calculate the distances between all pairs of points in the dataset. The KD tree is a binary tree structure which recursively partitions the parameter space along the data axes. In general, these structures attempt to reduce the required number of distance calculations, the basic idea is that if point A is very distant from point B, and point B is very close to point C, then we know that points A and C are very distant, without having to explicitly calculate their distance. Ball trees partition data in a series of nesting hyper-spheres, defined by a centroid $C$ and radius $r$, such that each point in the node lies within the hyper-sphere defined by $r$ and $C$. A single distance calculation between a test point and the centroid is sufficient to determine a lower and upper bound on the distance to all points within the node.

The computational time scales as $O(DN^2)$ in Brute Force and $O(D\log(N))$ in Ball Trees, where $N$ is the number of samples and $D$ the dimension. Regarding KD

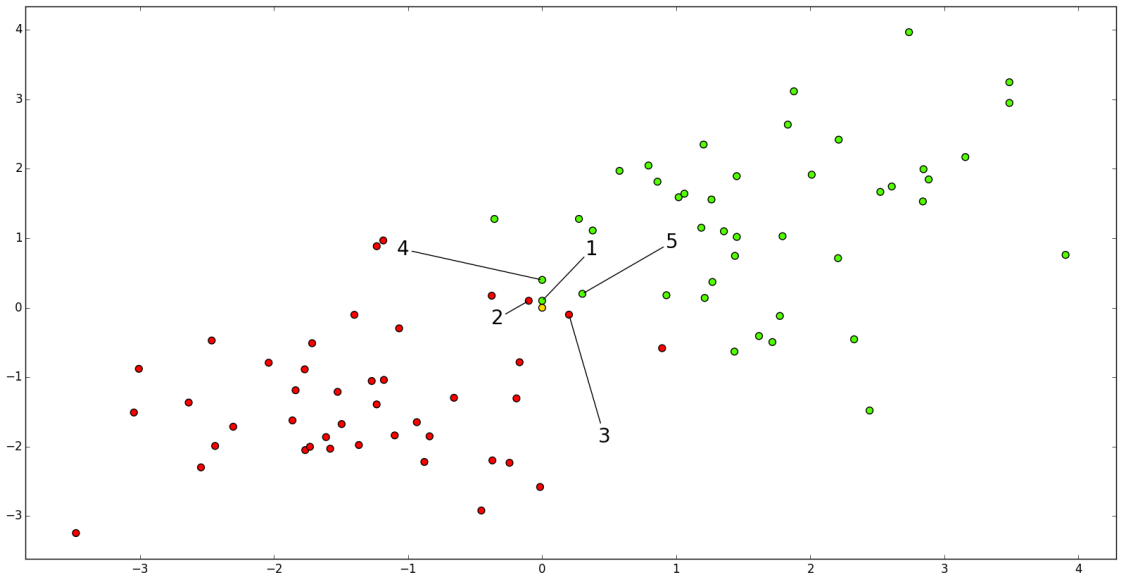



Figure 6: Example to illustrate the $k$-NN.

tree time complexity changes with D, for small D (less than 20 or so) the cost is approximately $O(D\log(N))$ and for larger D, the cost increases to nearly $O(DN)$.

### 2.2.4 Random Forest

RF is a notion of the general technique of random decision forests [28], which are an ensemble learning method for classification or regression. The goal of ensemble methods is to improve the performance of a single estimator combining the predictions of several base estimators. These methods may be divided in two families: the averaging methods, whose driving principle is to build several estimators independently and then to average their predictions; and boosting methods, which build, sequentially, the estimators one of which tries to reduce the bias of the combined estimator. The most well-known methods for each group are RF and AdaBoost, respectively.

According to the definition provided by the authors of [29], a random forest classifier consists of a collection of $T$ tree-structured classifiers $\{h(x,\Theta_k)\}_{k=1}^{T}$, where the $\Theta_k$ are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input $x$. Algorithm 1 shows the RF algorithm in pseudocode, where $\widehat{C}_i^{DT}(x)$ means the class predicted by the $i$-th decision tree and $\widehat{C}^{RF}(x)$ means the final class predicted by the RF. The build decision tree can be implemented by different algorithms, three of them will be explained below.

DTs are a non-parametric supervised learning method, their goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. Different algorithms exist to create the trees, like Iterative Dichotomiser 3 (ID3), C4.5, C5.0 and classification and regression tree (CART), among others. ID3 creates a multiway tree, finding for each node

**Algorithm 1** Pseudocode for random forest algorithm

Let be:
$T$ the number of trees,
$N$ the number of training samples.

forest ← CreateForest(T)
**for** x **in** set to classify **do**
  **for** tree **in** forest **do**
    $\widehat{C}_i^{DT}(x) \leftarrow$ tree prediction
  **end for**
  $\widehat{C}^{RF}(x) \leftarrow$ *mojority vote* $\{\widehat{C}_i^{DT}(x)\}_{i=0}^{T}$
**end for**

**function** CreateForest(T)
  **for** i = 0 to T **do**
    Randomly choose, with replacement, a subset of $n$ samples of the $N$.
    Create a root node
    $t_i \leftarrow$ BuildDecisionTree(node) ▷ Depends on the algorithm
  **end for**
  **return** $\bigcup_{i=0}^{T} t_i$
**end function**

the categorical feature that will yield the largest information gain for categorical targets. Trees are grown to their maximum size, then a pruning step is usually applied to improve the ability of the tree to generalise. C4.5 and C5.0 improve this algorithm, the former removing the restriction that features must be categorical and the latter improving some technicalities. CART is very similar to C4.5, it constructs binary trees using the features and threshold which yields the largest information gain at each node.

Besides the decision tree algorithm, two parameter may be adjusted. Firstly, the number of trees and, secondly, the size of the random subsets of features considered when splitting a node, which will be called $n\_estimators$ and $max\_features$, respectively, in the results chapter. Regarding the number of trees, the larger the better, but also the longer it will take to compute. In addition, the results will stop getting significantly better beyond a critical number of trees. In the case of the size of subsets of features, the lower the greater the reduction of variance, but also the greater the increase in bias. Although empirical good default values are $max\_features = \sqrt{n\_features}$ for classification tasks, where $n\_features$ is the number of features in the data, the best parameter values should always be cross-validated [25].

## 2.3 Evaluation metrics

There are many metrics that can be used to measure the performance of a classifier, depending on the problem and the specific goal. The following is a brief report on metrics which may be used to evaluate a binary classifier, still many of these metrics may be extended to multiclass classification.

In many real-world binary classification problems, the classes are unbalanced. This is an important factor that must be taken into account when choosing a metric to evaluate the classifier. It is recommended to compute the metrics for each class besides the all average, in this way the performance of each class may be visualized separately.

Before defining the different metrics four main concepts has to be defined True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). All four concepts refer to the number of predictions that has been made of a class, thus, TP/FP refer to the number of predictions predicted positives that were correct/incorrect, and TN/FN, which refer to the number of predictions predicted negatives that were correct/incorrect. Here a positive prediction means that the class predicted is the evaluated class and negative prediction means that the class predicted is the other class.

These can be arranged into a $2 \times 2$ contingency table, called confusion matrix, with columns corresponding to predicted values and rows corresponding to real values. Table 1 shows how can be arranged TP, TN,FP and FN. in a confusion matrix. In the case of binary classification, the TN value of the class 0 is the TP value of the class 1 and so on exchanging true values between each other and false values between each other.

| | | Predicted | |
|---|---|---|---|
| | | 0 | 1 |
| Real | 0 | TP | FN |
| | 1 | FP | TN |

Table 1: Confusion matrix of binary classification

The Matthews correlation coefficient (MCC) is a good way of describing the confusion matrix by a single number. It is in essence a correlation coefficient between the real and predicted binary classifications; its result is a value between $-1$ and $+1$. A coefficient of $+1$ represents a perfect prediction, 0 no better than random prediction and $-1$ indicates total disagreement between prediction and real. This measure is related to the chi-square statistic for a $2 \times 2$ contingency table and it is calculated as shown the equation (17).

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{17}$$

A lot of metrics may be obtained combing TP, TN,FP and FN; a few ones will

be explained below. The result of all these metrics range between 0 and 1, where 0 means a bad performance and 1 means a good one.

The simplest metric is the accuracy (ACC), which can be defined as the proportion of true results among the total number of cases examined and calculated as in equation (18). This metric gives an overview of the results, but in problems with imbalanced data, it may be better to use other metrics, because the accuracy results may hide a bad performance of the minority class.

$$ACC = \frac{TP + TN}{TP + FP + FN + TN} \tag{18}$$

Another two well-known metrics are Recall and Precision. Regarding Recall, also known as Sensitivity or True Positive Rate, is the proportion of real positive cases that are correctly predicted. Conversely, Precision, also called Specificity or True Negative Rate, denotes the proportion of predicted positive cases that are correctly real positives. Intuitively, precision is the ability of the classifier not to label as positive a sample that is negative, and recall is the ability of the classifier to find all the positive samples.

$$Recall = \frac{TP}{TP + FN} \tag{19}$$

$$Precision = \frac{TP}{TP + FP} \tag{20}$$

These two measures can be combined as a single measure, as shown in equation (21) and (22). The F1-score (21), also know as F-score or F-measure, is the harmonic mean of precision and recall. It is a particular case of $F_\beta$-score, where the weight of the precision and recall can be modified with the parameter $\beta$, as shown in equation (22).

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{21}$$

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}} \tag{22}$$

# 3 Experimental setup

The aim of this chapter is to explain the process followed in order to construct the dataset which will be used to build and evaluate the models. It is divided in five sections: Data Collection, Filtering Pipeline, Ground Truth, Feature Extraction and Feature Selection. Data Collection summarizes the collecting process, focusing on where the data were obtained, which sensors was used and how the users provided the ground truth. Filtering Pipeline details the process used to clean the data, which is divided in three steps, and reports the number of remaining samples before each step. Ground Truth presents the reason why the ground truth coming from the user can be used and the procedure used to obtain the final ground truth. Feature Extraction sets forth the features proposed and the procedure followed to reach them. Finally, Feature Selection describes the selected features and the reason why they have been selected among the others.

## 3.1 Data Collection

In order to collect a real-world dataset we contacted the Centre de Vida Independent (CVI) [2]. CVI is a Catalan centre whose aim is to improve the quality life of dependent people, both elderly and disabled, making their home life easier. CVI provides to their users the latest products to improve their self-reliant life and suggests to them reforms to remove architectural barriers from their home. They also provide continuous assistance to their users, in this context they saw great potential in our system, loving the idea of a passive telemonitoring system which allowed them to be aware of their users state and to be notified about risky situations.

The system was installed in 10 homes of 10 volunteers CVI users (mean age $72.4 \pm 6.4$ years, one man and nine women). Unfortunately, one of these users left the project because her home was flooded some days after the installation was done. The data collected from this user will not be used so henceforth only the other 9 users will be taken into account.

Only PIR and door sensors were used in these installations, Table 2 shows the sensors used in each installation, one PIR sensor in each room and one door sensor in the main door. Pressure mats were not installed because some issues were detected when testing them in lab. The main issue is related with mat position, which is important to detect pressure correctly, the fact that the sensor is installed on the bed makes it easier for the mat to move during the night or when the user makes the bed.

The sensors send data in hexadecimal but when data are stored in the database it is done with the proper data type. PIR sensors send three kinds of data when the sensor is triggered: a binary value, which is stored as a boolean, to inform that the sensor has been triggered, and two multilevel values, which are stored as floats, one with the temperature in Celsius degrees and the other with the percentage of the luminance. Door sensors send a binary value, which is stored as a boolean,

[2] www.cvi-bcn.org

with true value if the door has been opened and with false value if the door has been closed, in addition it has an integrated PIR sensor which can be switched off. Testing the door sensors in the lab, it has been found that the integrated PIR sensor can interfere with the door sensor if both are triggered at the same time, therefore the integrated PIR sensors have been switched off.

| | User 1 | User 2 | User 3 | User 4 | User 5 | User 6 | User 7 | User 8 | User 9 | User 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Main Door | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| PIR Bedroom | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| PIR Kitchen | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| PIR Bathroom | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| PIR Living room | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| PIR Other room | ✓ | | ✓ | | | ✓ | ✓ | ✓ | ✓ | ✓ |

Table 2: Sensors used in each installation.

The telemonitoring period lasted for five months, from May 15th to October 20th. We requested that CVI users answer a questionnaire every day with some relevant information to create and evaluate the intelligent modules. During the telemonitoring period a problem with the method used to obtain the questionnaire answers was detected. Initially, the method used was an online questionnaire. A month later we realised some users were not answering the questionnaire, the reason was they were not used to using the internet. Therefore, the method was changed and in the next months a CVI employee called them every day asking the questionnaire questions and writing down the answers for us.

Unfortunately, although the changes were to ease the task of the users, they did not answer the questionnaire every day as we requested. Table 3 contains the amount of days that CVI users answered the questionnaire, grouped by user and month. It shows that during the first month some users did not answer the questionnaire any day, thus, we found out there was a problem with the online questionnaire and changed the method, as explained above. The amount of answers from the next months did not increase as expected, the users gave different reasons for not answering the questionnaire, for example, some of these days they were away from home, so these days do not concern us because the user did not perform any activity at home; in contrast some days the user did not answer the questionnaire even when at home and performing normal daily activities, they explained that they did not remember what they had done the day before so they could not answer the questionnaire. Moreover, the CVI employee, who called them, took vacation in August so we do not have any answers from this month. As shown, including all the answers from six months and from all the users we have a total of 180 days.

| | mid-May to mid-June | mid-June and July | August | September | October | Total |
|---|---|---|---|---|---|---|
| User 1 | 0 | 9 | 0 | 8 | 8 | 25 |
| User 2 | 11 | 3 | 0 | 1 | 7 | 22 |
| User 3 | 9 | 3 | 0 | 4 | 6 | 22 |
| User 4 | 10 | 6 | 0 | 7 | 0 | 23 |
| User 5 | 0 | 5 | 0 | 5 | 0 | 10 |
| User 6 | 0 | 2 | 0 | 3 | 1 | 6 |
| User 7 | 10 | 10 | 0 | 2 | 2 | 24 |
| User 8 | 18 | 10 | 0 | 9 | 8 | 45 |
| User 9 | 0 | 2 | 0 | 1 | 0 | 3 |
| Total answers | 58 | 50 | 0 | 40 | 32 | 180 |

Table 3: Number of days that the users answered the questionnaire, grouped by user and month.

## 3.2 Filtering Pipeline

In order to obtain our dataset which will be used to create and evaluate the methods, the data were filtered, dismissing those days with noisy data. The filtering pipeline is composed of three steps, each one uses different criteria (sensor errors, anomalies detected by ED module and multiuser at night) to dismiss data from the final dataset.

During the telemonitoring period the sensors of some installations had issues, so the data from these days is not reliable. The main issues were that one or more motion sensors stopped detecting motion (so they did not send information at all) or they detected motion but they did not send luminance information. The aim of the first step of the filtering pipeline is to dismiss those days when some sensors did not send motion or luminance information at all.

The data which arrives to AR module is filtered by the ED module, as explained in section 1.3.3. As sleeping activity recognition will be integrated in AR module, the second step of the filtering pipeline uses the same criteria of ED module to filter those days with critical situations from the dataset. In this step, days with sensor malfunctioning, due to low battery, or Rasberry pi disconnections are dismissed.

The final step focuses on those days with multiuser at night. The overall system does not compute information when there is more than one person at home because, with the kind of sensors installed, it is not possible to distinguish between the actions performed by the user and by the guest. When another person enters to the home the system stops classifying the activities, so those days with multiuser at night are dismissed from the dataset.

The data from those 180 days in which the user answered the questionnaire was filtered by the pipeline, step by step. First of all dismissing those days with sensors errors, then filtering the remaining days by ED module and, finally, removing from the remaining dataset those days with multiuser at night. At the end of the filtering pipeline, only data from four different users remained, making a total of 22 days. Table 4 shows the remaining days after each step of the pipeline, grouped by user. As shown, the step which dismissed more days is the first one, this step dismisses

days with sensor problems, so the team in charge of the sensors and the installations was informed.

| | Quest | Remaining after dismiss sensors errors | Remaining after dismiss anomalies detected by ED module | Remaining after dismiss multiuser night rows |
|---|---|---|---|---|
| User 1 | 25 | 3 | 0 | 0 |
| User 2 | 22 | 1 | 0 | 0 |
| User 3 | 22 | 17 | 12 | 7 |
| User 4 | 23 | 10 | 7 | 2 |
| User 5 | 10 | 3 | 0 | 0 |
| User 6 | 6 | 2 | 0 | 0 |
| User 7 | 24 | 12 | 8 | 5 |
| User 8 | 45 | 25 | 21 | 7 |
| User 9 | 3 | 1 | 0 | 0 |
| Total | 180 | 74 | 48 | 21 |

Table 4: Remaining days after each step of the pipeline, grouped by user.

## 3.3 Ground Truth

The ground truth was based on the answers of the following questions *What time did you stand up from bed?* and *What time did you go to sleep?*. The users always responded using on the hour or increments of 15 minutes and sometimes these hours do not correspond to a cessation of activity in the house. For these reasons the ground truth has been refined with the knowledge of three experts who readjusted the hour when the user went to sleep and woke up relying on the data coming from the sensors.

The manner in which the users answered the questionnaire has been explained in Section 3.1. Regarding the process followed by the experts, their aim was to respond to the questions *What time did the user stand up from bed?* and *What time did the user go to sleep?* for each day of the dataset. To do their work the experts has the user's answers as a guide and a graph, for each day, with all the motions grouped by room in a temporal axis. As the sensors only send information when they are triggered, the experts' answers coincide with a bedroom motion.

Figure 7 is an example of this graph, axis $y$ shows different rooms, the ones with PIR sensor and axis $x$ is a temporal axis. The motions are plotted as coloured points, each room has a different colour. The graph used by the expert was interactive, they can zoom in the temporal axis and the exact time when the motion happened appears pointing to a specific motion.

To choose the definitive ground truth the answers of the three experts have been compared, if two of them coincided this time was chosen as definitive, in case the three answers of the experts differed they explained why they chose their answer and reached an agreement.

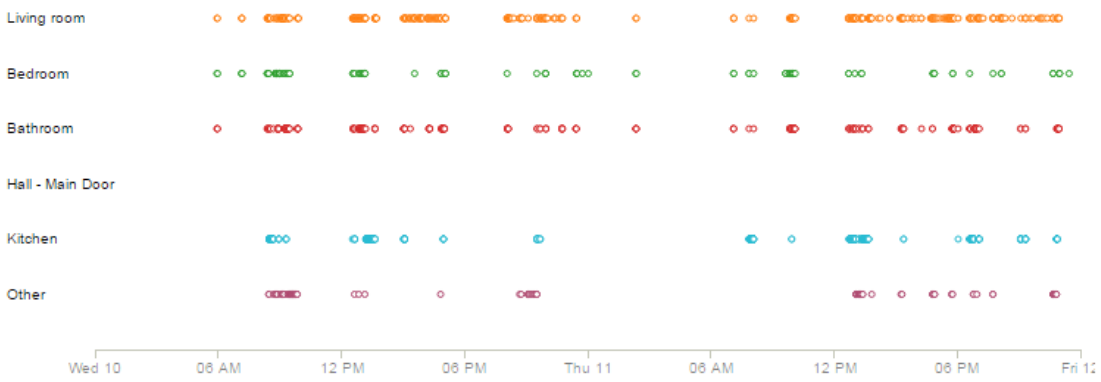



Figure 7: Example of a graph with all the motions grouped by room in a temporal axis.

Figure 8 shows the difference between the questionnaire and the expert's answers. It has a temporal axis (axis $x$) and each coordinate of axis $y$ represents a different day of the dataset, showing, in red, the sleeping activity hours coming from the questionnaire answers and, in blue, the sleeping activity hours according to ground truth. As both sleeping activity hours of the same day are plotted in the same $y$ coordinate if questionnaire and ground truth coincide the colour turns purple. If the "*go to sleep time*" and/or "*wake up time*" do not coincide, there is a text next to the corresponding side with the difference. If the difference is bigger than an hour the text appears in red. In the middle of each bar there is the total time that questionnaire answer differs from the experts' answer.

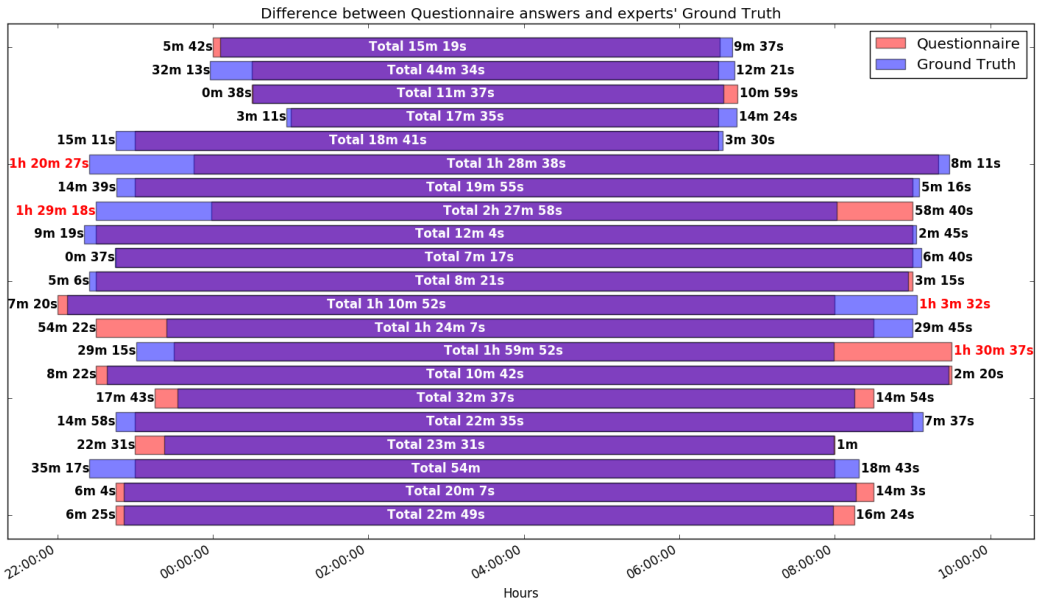



Figure 8: Difference between questionnaire and experts' answers.

As shown, most answers only differ by a few minutes and a few differ between 15 and 30 minutes. About the seven occasions where the difference is bigger than

an hour the experts said that relying on the motions it was not possible that the questionnaire answers was correct. After the experts' conclusions the ground truth used was the answers coming from them.

## 3.4 Feature Extraction

As explained in Section 3.1 the raw information coming from sensors is motion, temperature in Celsius degrees and luminance as a percentage. The temperature does not significantly vary when the user enters a room or when s/he is sleeping, so this information was not used to calculate the features to distinguish if the user is performing their sleeping activity or not. The motion and luminance information has been plotted as in Figure 9, for each night in dataset, in order to visualize how this information can be used to distinguish between the classes *awake* and *sleep* explained in section 1.4.

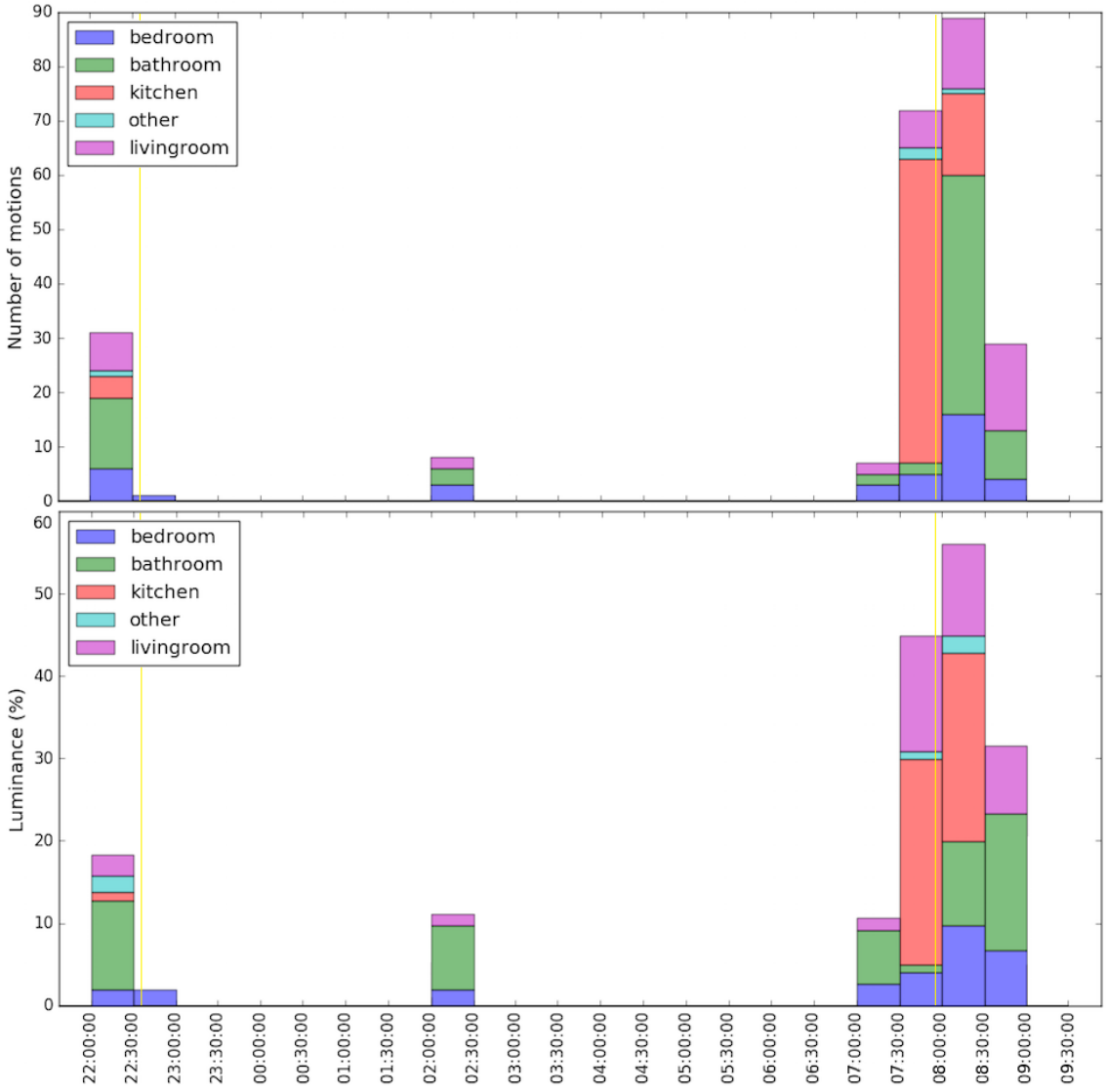



Figure 9: Example of plotted motion and luminance information

Figure 9 can be used as a representative of the all dataset to understand the conclusions extracted. The figure is divided in two subplots, they share the axis x which is a temporal axis, where the time is divided in slices of 30 minutes. The y axis is different for each subplot, the upper one shows the number of motions and the y axis of the bottom subplot shows the amount of luminance information as a percentage. In both subplots, the information has been plotted split by rooms.

As shown all the information between the "*go to sleep time*" and the"*wake up time*" comes, in most cases, from the bedroom and sometimes from specific rooms, such as the bathroom or the living room. In contrast, all the information before the "*go to sleep time*" and the"*wake up time*" is higher than in the bedroom. Almost always, motion and luminance information increases and decreases together but it is important to consider these two measures because in some cases as, for example when the user is in bed reading a book or watching TV, the number of motions can decrease before the luminance does.

The conclusion extracted with these plots is that the main difference between sleeping activity periods and awake period is if the information regarding motion and luminance comes from bedroom sensor or from all the sensors, but in some cases the bedroom information is needed separately. Moreover, both motion and luminance information is important. All this consideration can be summed up in the next four quantities:

- Number of motions in bedroom.
- Number of motions in all the rooms.
- Amount of luminance in bedroom.
- Amount of luminance in all the rooms.

These four quantities can be calculated in different periods of time. In Figure 9, the time has been divided in slices of 30 minutes, but this method may not be appropriate to compute the features.

On the one hand, the sensor only send information when it is triggered so if the time is divided in equal periods, as in Figure 9, the luminance information calculated can be wrong. For example if in one of these periods any motion happened the luminance calculated would be 0% but the truth is that in that period there is no luminance information. On the other hand, the overall system, where our classifier has to be integrated, tries to classify periods between motions. So if the features were calculated using periods between consecutive motions, the two first quantities would be zero.

Mixing the idea of slicing the time in fixed periods and using motions as a reference, the features will be computed for each bed motion and the slice of time will be a fixed amount of minutes before and after this motion. The time when the motion took place can be significant, so it is going to be another feature.

Finally, the next features have been proposed taking into account the four points, extracted with information visualization, and the consideration presented above.

1. The time when motion took place.
2. Quantity of motions in bedroom $[2, 5, 10, 15]$ minutes before the motion.
3. Quantity of motions in bedroom $[2, 5, 10, 15]$ minutes after the motion.
4. Quantity of motions in all the rooms $[2, 5, 10, 15]$ minutes before the motion.
5. Quantity of motions in all the rooms $[2, 5, 10, 15]$ minutes after the motion.
6. Average of luminance in bedroom $[2, 5, 10, 15]$ minutes before the motion.
7. Average of luminance in bedroom $[2, 5, 10, 15]$ minutes after the motion.
8. Average of luminance in all the rooms $[2, 5, 10, 15]$ minutes before the motion.
9. Average of luminance in all the rooms $[2, 5, 10, 15]$ minutes after the motion.

After computing the features a quick graph was plotted to visualize them. Figure 10 is an example of this graph, each graph contains the features of one night, using before or after motion information and calculated with the same slice of time. The graph is divided in two subplots, they share the axis x which is a temporal axis, but only the bed motions have been plotted. The y axis is different for each subplot, the upper one shows the features with motion information and the y axis of the bottom subplot shows the features with luminance information. In both subplots, the information has been plotted taking into account bedroom PIR sensor (red) and all PIR sensor of the house (blue).

Plotting the features a problem with the data coming from User 8 was detected. During the same night some sensor sent luminance information only sometimes. This problem had not been detected during the filtering process because the first step only discards those days without luminance data. In order to avoid introducing noise in data set, this user's data are not going to be used.

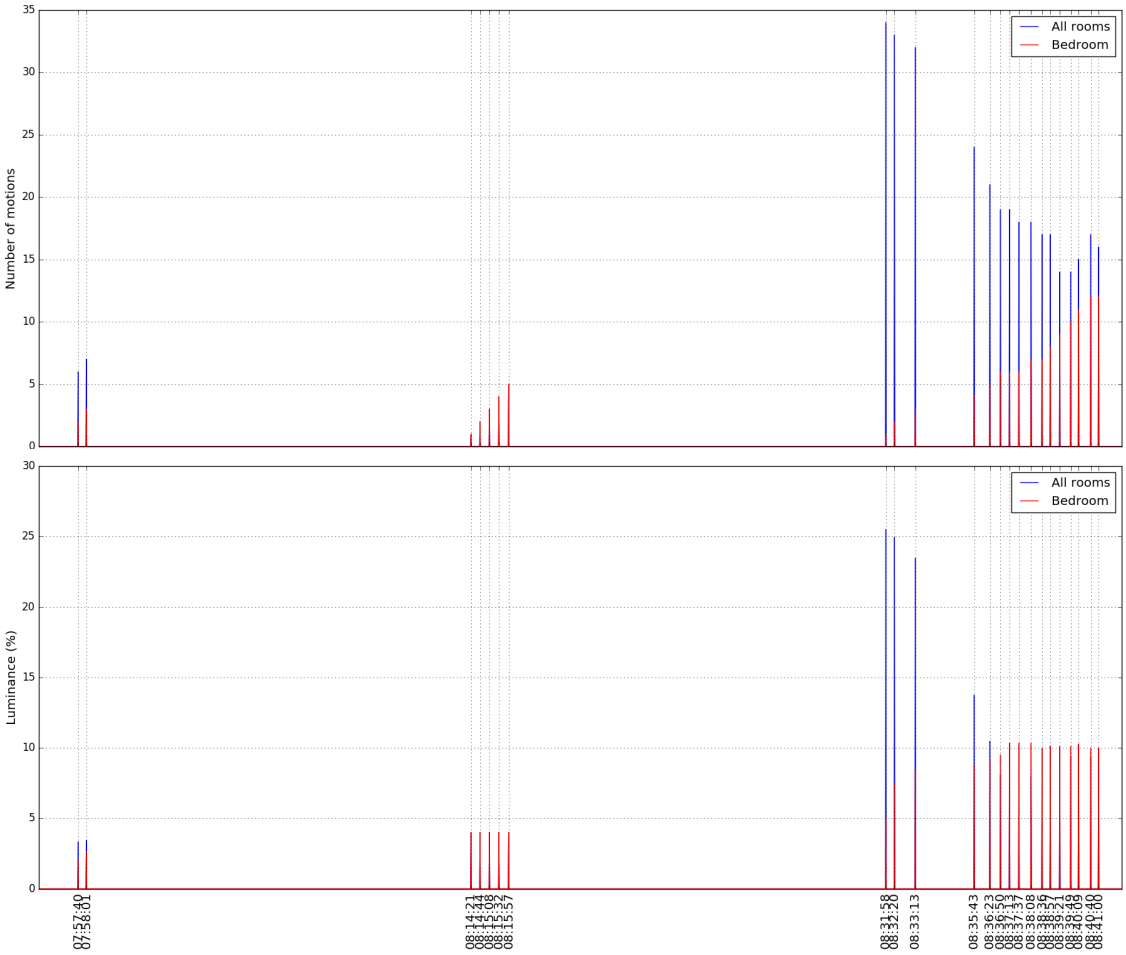



Figure 10: Example of plotted features.

## 3.5 Feature Selection

The first criteria to reduce the number of features to use with the classifier is a practical one. The overall system is analysing the data coming from the sensors continuously, the information sent to the healthcare center can be computed in real time, on-line, or at the end of the day, off-line. The quantity of hours that the user spent performing sleeping activity will be calculated off-line, but the healthcare center shows the activity which is performed by the user, in real time, in order to include the sleeping activity in this visualization the features have been reduced to those which do not contain information after the motion (1,2,4,6 and 8).

In Figure 11, we can see the correlation between the features and the class, as expected the correlation between the same kind of features with close slices of time is high. But as there are not any critical correlation (1, $-1$ or 0), no feature has been dismissed. The features can be ranked by the correlation value between the class and the feature. But as said before the number of features that would be selected using this ranking should be obtained empirically. For lack of time, the feature selection will be done as a feature work, all seventeen features will be used

for this project.

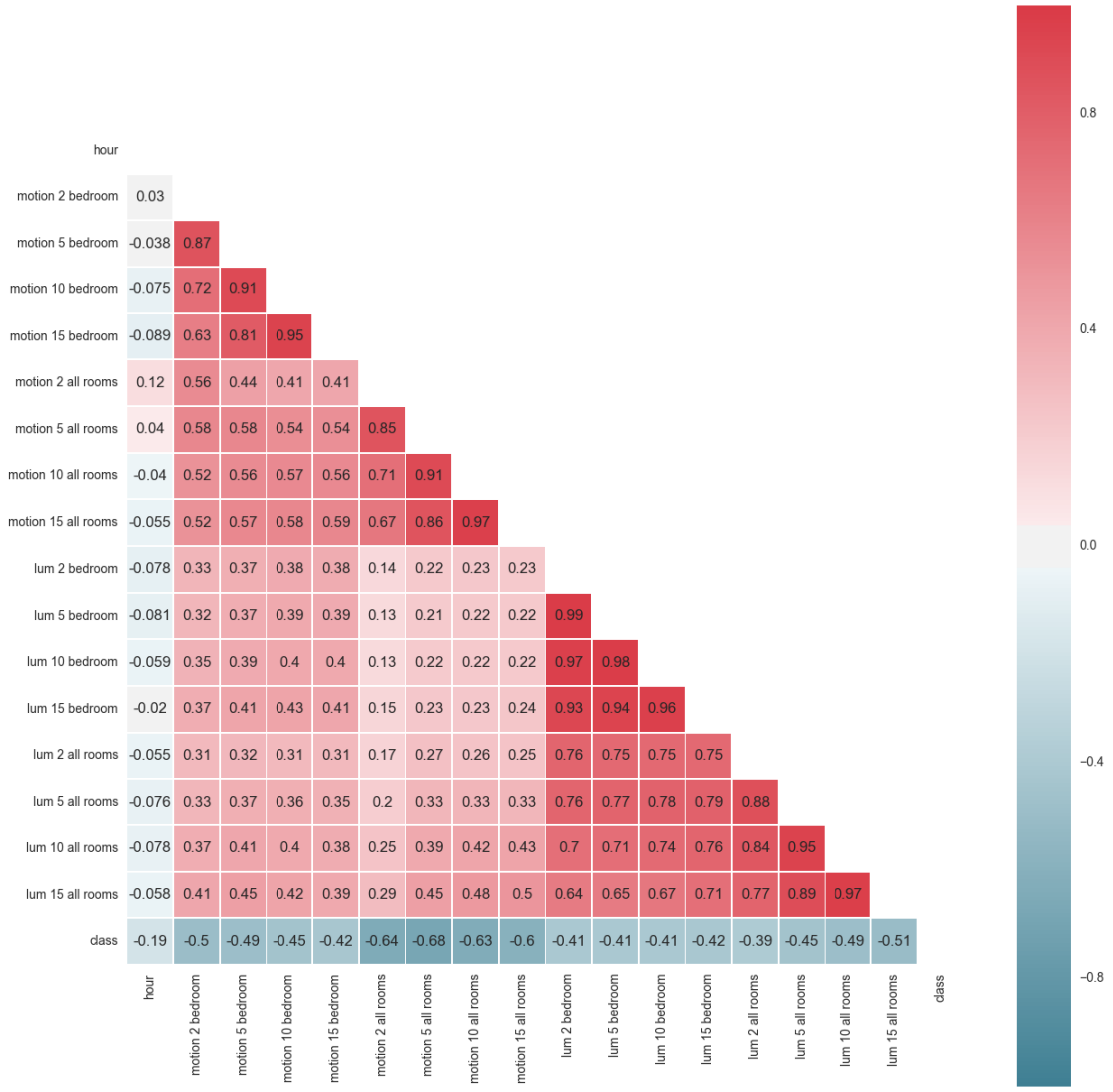



Figure 11: Correlation between the features and the class.

Figure 12 shows PCA of the dataset, a lot of *sleep* samples are separated from the *awake* samples, those with x coordinate near to -4, but the other ones are mixed with some *awake* samples. The mixed samples probably are those near to the "*go to sleep time*" and the "*wake up time*", because during some nights some users moved a lot in the bed after they went to sleep or before they woke up.

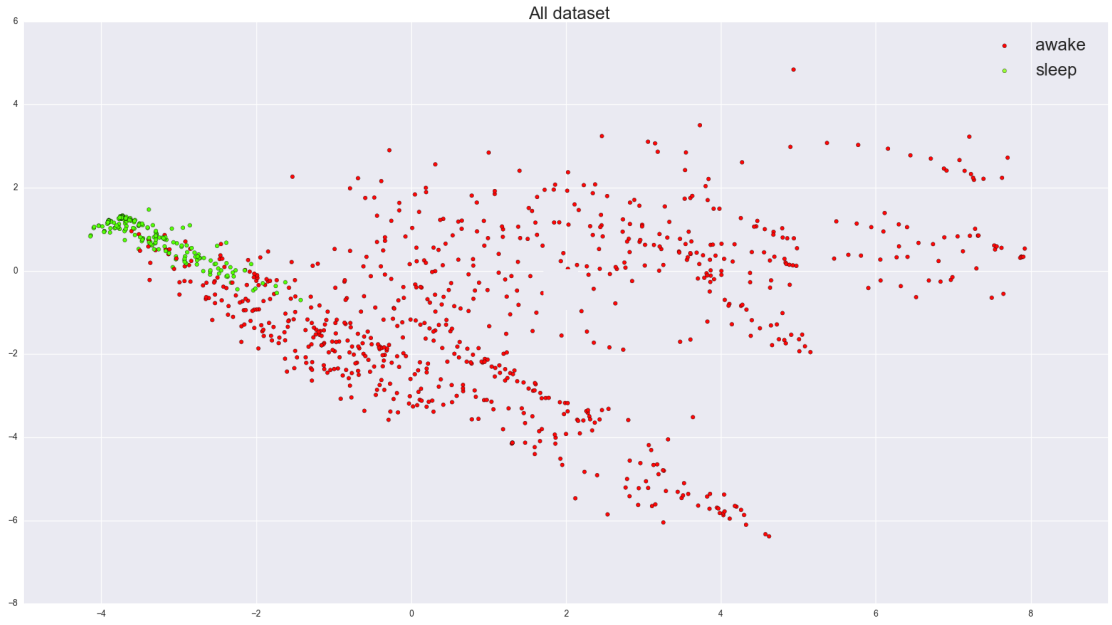



Figure 12: Dataset PCA.

# 4 Results

The purpose of this chapter is to present the results of this project. It is divided in three sections: Partitioning Data, Building and Evaluation. The first one detail the partition of the data in two different sets: the training and the testing set. In Building are presented the results obtained during the model fitting using the techniques explained in section 2.2.1, and the model chosen. Evaluation is divided in two subsections: Classifier and Overall system. In the former, the results obtained using the best classifier on the testing set are presented and the latter starts explaining the method used to computed the informations demanded in the requirements, and then the results obtained with the Ruled-based system and the MLS are compared.

## 4.1 Partitioning Data

The dataset is composed of 14 nights coming from three different users. In order to validate not only the classifier but also the overall system and compare the classifier's results with the ruled-based's results, the reduced-dataset has been split so that all the rows from the same night go to the same set. As the dataset only contains 14 nights in total, it has been split so that the training set will contain 8 nights and a testing set 6 nights. Moreover, both sets have nights from each user. Table 5 shows the number of samples from each class in each set. The fact that the split was made by nights and not by samples and each night has a different number of samples means that the number of samples in both sets are almost the same, as shown in Table 5.

| | | Set | | |
|---|---|---|---|---|
| | | Train | Test | Total |
| Class | Awake | 371 | 393 | 764 |
| | Sleep | 122 | 109 | 231 |
| | Total | 493 | 502 | 995 |

Table 5: Number of samples from each class in each set.

To visualize the two datasets separately Figure 13 shows the PCA of both in different subplots, the training set on the left and the testing set on the right. The red points of both subplots are samples which belong to *awake* class and the green ones belong to *sleep* class.

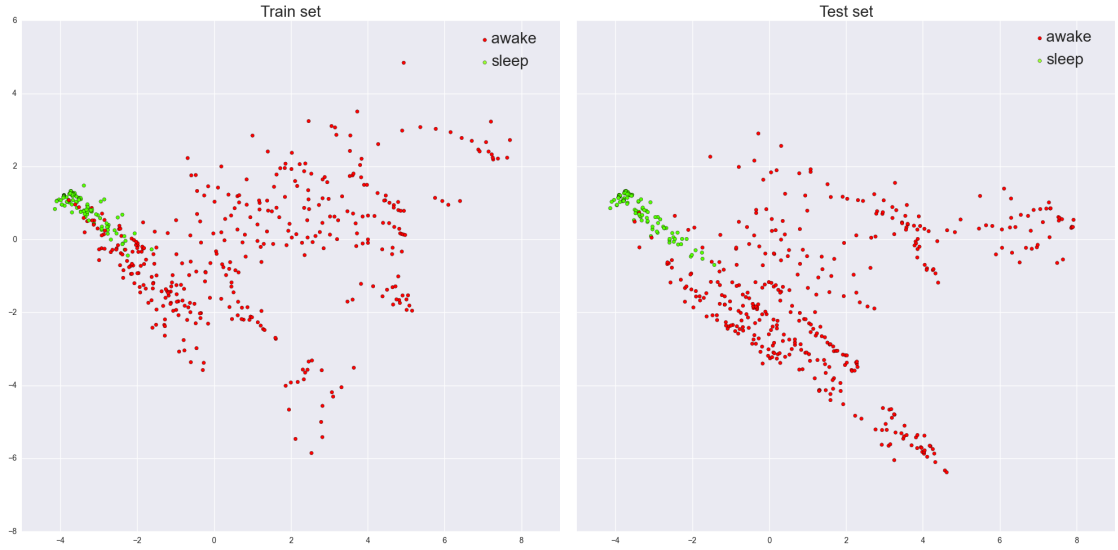



Figure 13: PCA of both dataset, train (on the left) and test (on the right).

## 4.2 Building

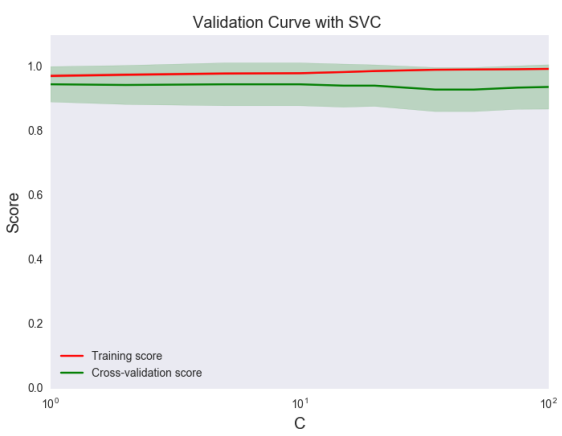
Three different machine learning techniques have been used (SVM, $k$-NN and RF) to build a classifier with the training set. A 10-fold cross-validation method had been used over the training set to choose the best parametrization. Figure 14 shows the validation curves of each classifier varying one of their parameters. As shown, the score would not increase if the parameters used in each classifier were changed.

Table 6 shows the results of the best parametrization for each one, all the classifiers have obtained a great accuracy, the best performance has been obtained relying on the SVM (RBF kernel, $C = 1.0$, $\gamma = 1.0$).

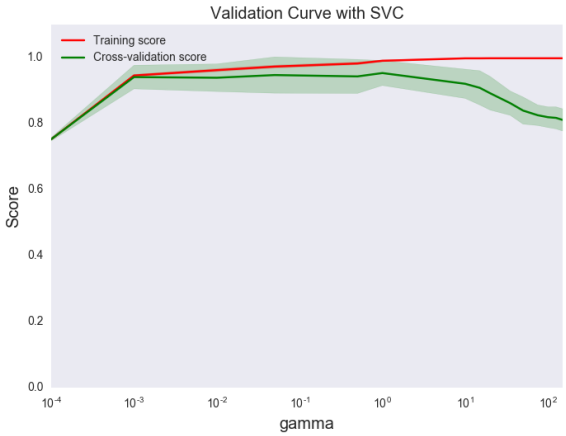
| Classifier | Parameter (s) | Accuracy |
|---|---|---|
| SVM | $kernel = rbf$, $C = 1$ , $\gamma = 0.055$ | $0.947 \pm 0.072$ |
| **SVM** | $\boldsymbol{kernel = rbf, C = 1.0, \gamma = 1.0}$ | $\mathbf{0.953 \pm 0.038}$ |
| KNN | $weights = uniform$ , $n\_neighbors = 13$ | $0.941 \pm 0.048$ |
| KNN | $weights = distance$, $n\_neighbors = 9$ | $0.945 \pm 0.066$ |
| RF | $n\_estimators = 7$, $n\_features = 4$ | $0.939 \pm 0.073$ |
| RF | $n\_estimators = 10$, $n\_features = 1$ | $0.941 \pm 0.050$ |

Table 6: Classification results during the training.

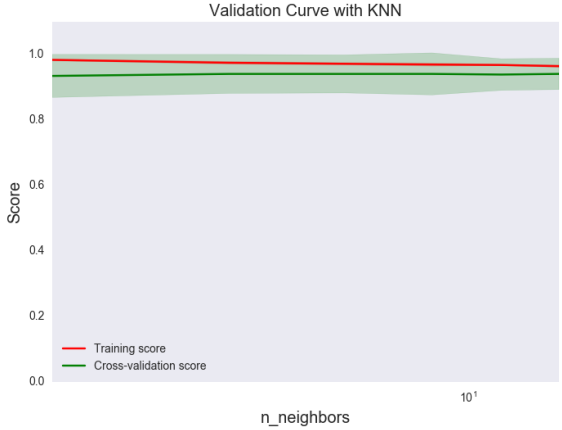
In order to see if it is possible to obtain better performance by increasing the training dataset the learning curve of the SVM (RBF kernel, $C = 1.0$, $\gamma = 1.0$) has been plotted, see Figure 15. As shown, the training and the cross-validation score converge around the top of the training example, so the score will not improve by adding more samples in the training set.



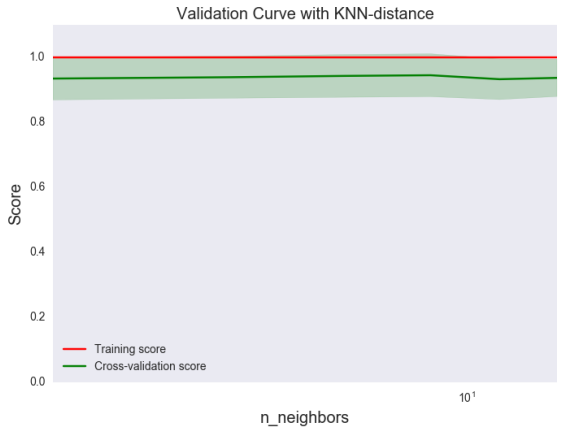

(a) SVM $C$



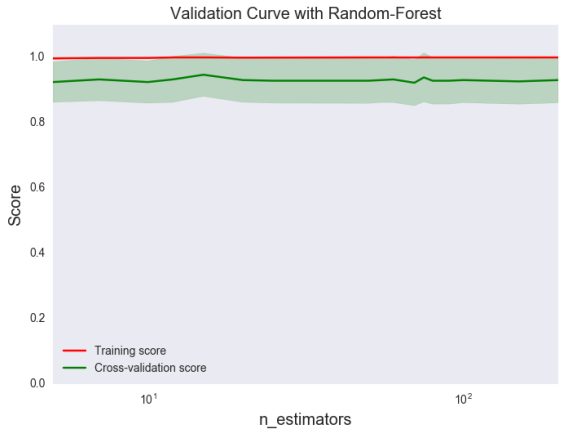

(b) SVM $\gamma$



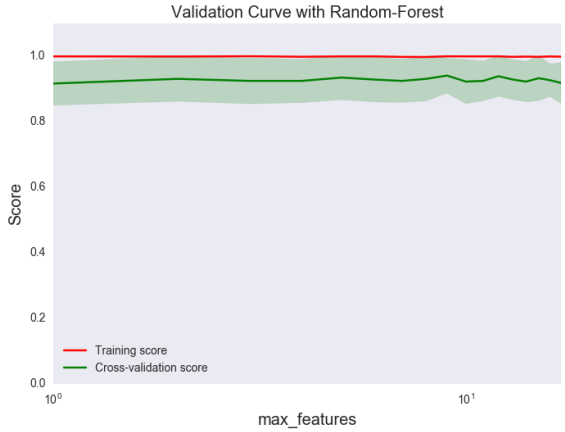

(c) KNN neighbours ($weight = uniform$)




(d) KNN neighbours ($weight = distance$)




(e) RF estimators




(f) RF features

Figure 14: Validation curves of each classifier varying one of their parameters.

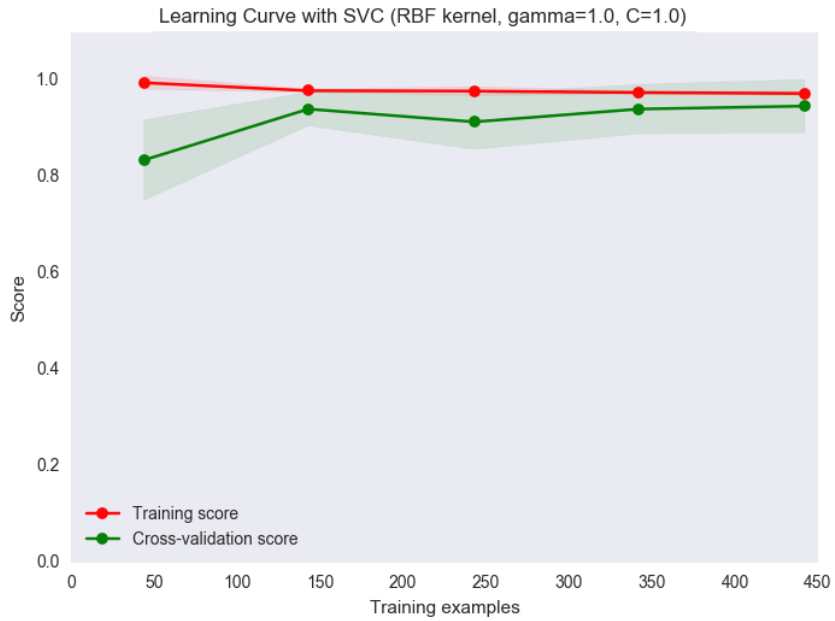



Figure 15: SVM (RBF kernel, $C = 1.0$, $\gamma = 1.0$) learning curve.

## 4.3 Evaluation

### 4.3.1 Classifier

The best classifier has been used to classify the testing set and, on average, it obtained a F1 of 0.91 and an accuracy of 0.964. Table 7 shows the performance split by classes. The confusion matrix showed on Table 8 obtains a MCC of 0.89, only 18 samples were misclassified in *sleep* class when they were in fact *awake*.

| | **Precision** | **Recall** | **F1-score** | **Support** |
|---|---|---|---|---|
| **Awake** | 0.96 | 1.00 | 0.98 | 393 |
| **Sleep** | 1.00 | 0.83 | 0.91 | 109 |
| **Total** | 0.97 | 0.96 | 0.96 | 502 |

Table 7: SVM (RBF kernel, $C = 1.0$, $\gamma = 1.0$) results split by class, using testing set .

| | | Predicted | |
|---|---|---|---|
| | | Awake | Sleep |
| Real | Awake | 393 | 0 |
| | Sleep | 18 | 91 |

Table 8: SVM (RBF kernel, $C = 1.0$, $\gamma = 1.0$) confusion matrix using test set.

Figure 16 shows the test set PCA, using red points for *awake* samples and green ones for *sleep* samples. On the left the testing set with the ground truth labels has

been plotted and on the right with the predicted labels.

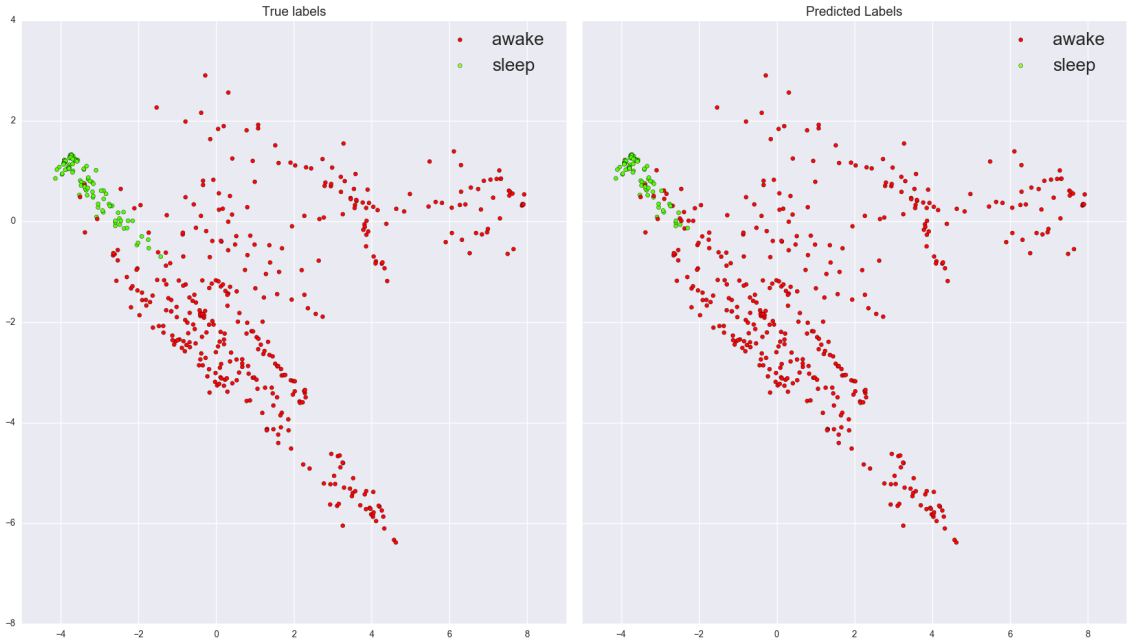



Figure 16: Testing set PCA, with ground truth labels (on the left) and predicted labels (on the right).

This set, as well as the training set, only includes night samples. For this project, the night is defined as the period between 10 pm and 12 pm. The reason for choosing these limits, for the night period, is that all the users went to sleep after 10 pm and woke up before 12 pm during the days of the dataset. In order to know if the classifier is able to classify all the bedroom motions, without restrictions, the bedroom motions between 12 pm and 10 pm of the 6 days in the testing set has been classified. The classifier achieve an accuracy of 100%, which means it classifies all the bedroom motions as *awake*. This is a great achievement because it means that the classifier can be used on all bedroom motions and it is not necessary to distinguish between day an night.

### 4.3.2 Overall system

Once the classifier has predicted the class of each period, the information is post-processed with a simple sequential track in order to obtain the "*go to sleep time*" and the "*wake up time*" and to compute the number of sleeping activity hours and the number of resting hours. The "*go to sleep time*" will be the time when the first *sleep* period started, and the "*wake up time*" will be the time when the last *sleep* period finished. The number of sleeping activity hours will be computed as the difference between these two times. Finally, the duration of the *awake* periods between the "*go to sleep time*" and the "*wake up time*" will be added together and then subtracted from the number of sleeping activity hours, in order to compute the number of resting hours.

Figure 17 shows an example in order to illustrates the post process. The classified periods of one night are plotted, grouping the consecutive period of the same class and printing their duration below each period. The "*go to sleep time*" and the "*wake up time*" are marked in yellow. In the example, the number of sleeping activity hours are 11h 15m, as it shown in Figure 17; and the number of resting hours will be computed as 11h 15m $-(9\text{m} + 2\text{m}) = 11\text{h } 4\text{m}$.

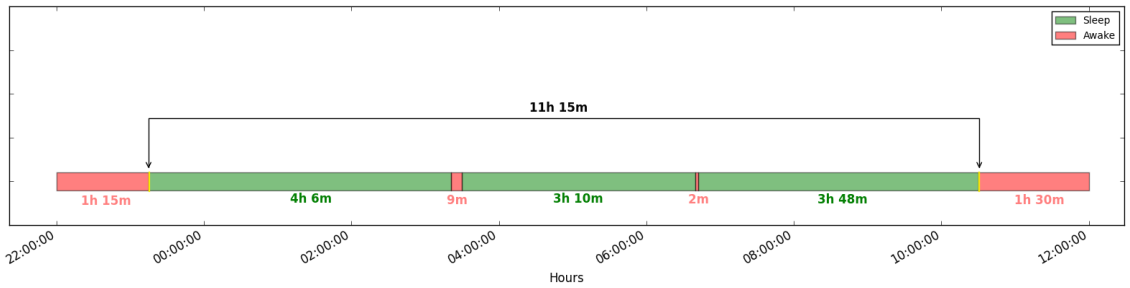



Figure 17: Example of a classified night.

In order to verify if the performance of the MLS is better than the performance of the ruled-based system, the systems have been evaluated comparing the number of sleeping activity hours according to the ground truth and those coming from the results.

Figure 18 shows the comparisons between the ground truth and the results obtained with ruled-based system (on the top) and MLS (on the bottom). All the plots have as temporal axis (axis $x$) and each coordinate in axis $y$ represents nights in the dataset, each subfigure shows, in red, the sleep activity hours according to the ground truth and, in blue, the sleep activity hours obtained by the system. As both sleep activity hours of the same night are plotted in the same $y$ coordinate, if the ground truth and the results coincide the colour turns purple. If the "*go to sleep time*" and/or "*wake up time*" do not coincide, there is a text next to the corresponding side with the difference between the time coming from the ground truth and that coming from the results. If the difference is bigger than an hour the text appears in red. In the middle of each bar there is the total time which results differ from the ground truth.

As shown, Ruled-based results are rather different from the ground truth, on some nights the difference between the ground truth and the results is reasonable (less than 20 minutes), whereas the difference on the other nights vary from about 1 to 2 hours. Regarding results obtained using MLS, 50% of the results coincide completely with the ground truth and the others differ from the ground truth by about 15 minutes.

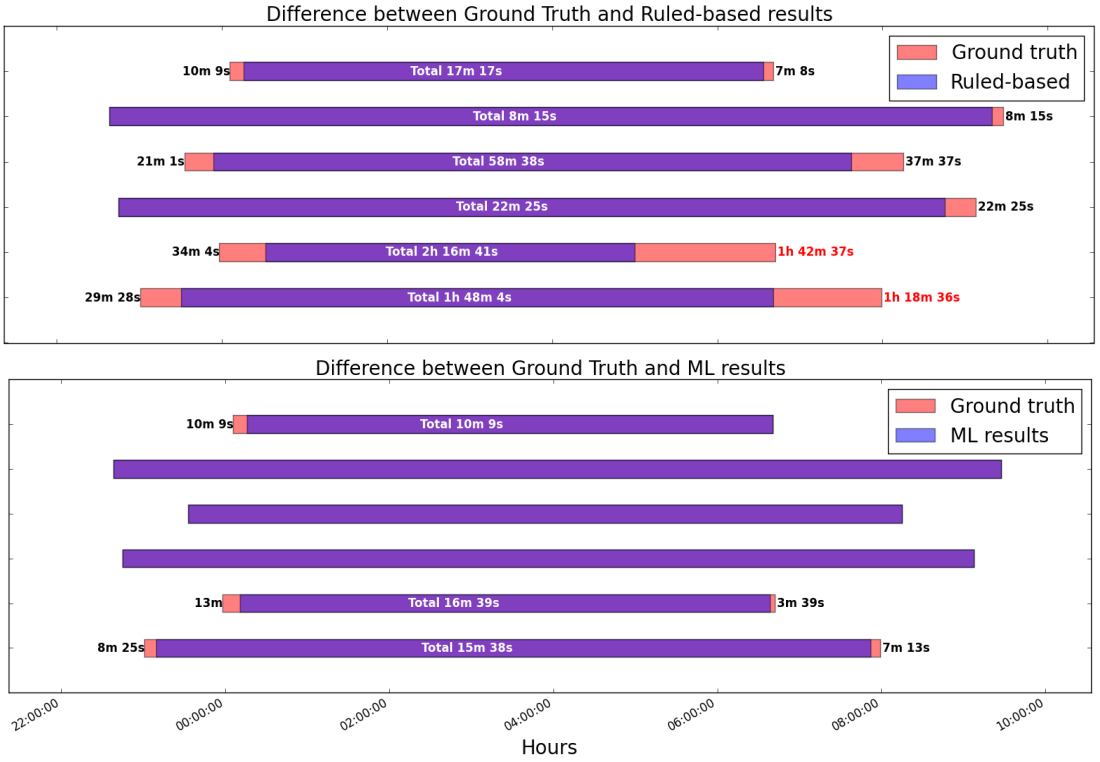



Figure 18: Comparing the ground truth and results obtained with ruled-based system (on the top) and MLS (on the bottom).

Figure 19 summarises the results obtained with both techniques. It shows the accuracy of each system increasing the error allowed. The error is measured as the difference between the results and the ground truth, in increments of 6 minutes. As shown, machine learning results achieves an accuracy of 50% with an error of 0 minutes, which means 50% of the results coincide with ground truth and the 100% of accuracy is reached with an error of 18 minutes. Regarding ruled-based results, an accuracy of 50% is reached with an error of 24 minutes and an accuracy of 100% is obtained with an error of 2 hours and 18 minutes, with two hours of difference respect machine learning results.

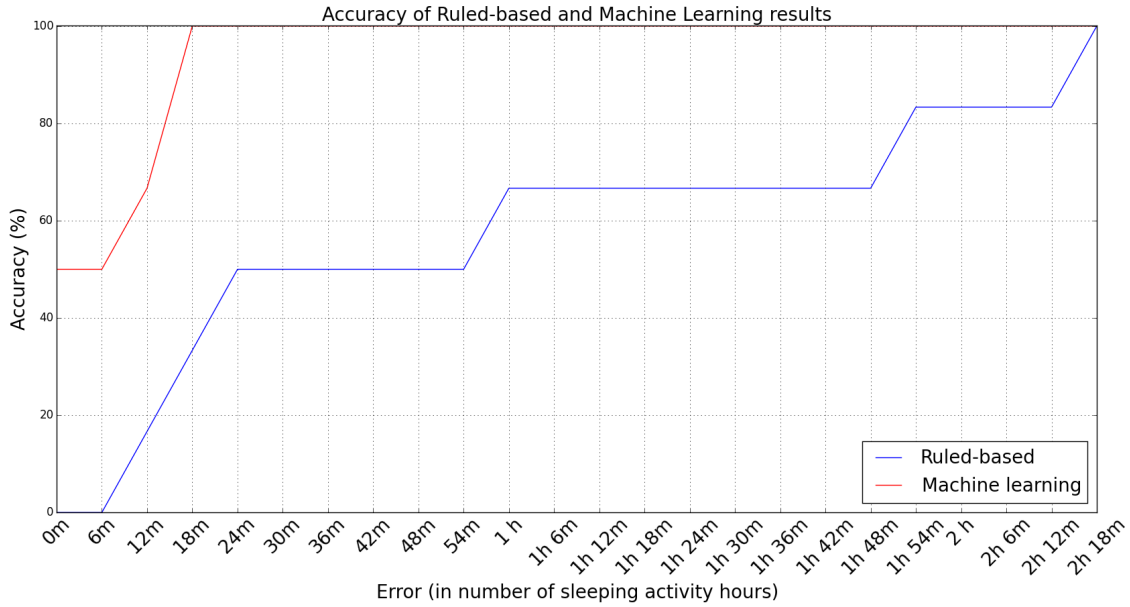



Figure 19: Accuracy of Ruled-based and Machine learning results.

# 5 Conclusions

This chapter concludes the TFG describing the conclusions. It is divided in two sections: Conclusions and Future work and TFG conclusions. The former summarises the project's goals, discussing whether they have been achieved or not, and exposes some future work. The latter explains my personal conclusions drawn whilst carrying out this TFG.

## 5.1 Conclusions and Future work

Community based living, often alone with intermittent care, creates possible scenarios of risks for all individuals. When cognitive changes are likely to have taken place it is crucial to understand what the risks may be and monitor these. To monitor users at home, a sensor-based system has been developed, which is able to gather data and report on the stability and evolution of the user's daily life activity. Unfortunately, performance of sensor-based telemonitoring and home support systems depends, among other issues, on the reliability of the adopted sensors. To solve this problem, a IMS has been developed with different modules. This work presents a MLS, whose aim is to detect sleeping activity.

In order to collect a real-world dataset, the SB-TMHSS was installed in 10 elderly people's homes in Barcelona. The collected data have been cleaned by a pipeline, whose aim is three-fold: dismissing the days with sensor errors, anomalies and multiuser activity. Although the telemonitoring period lasted for five months, after the cleaning activity, only data from 4 users remained, making a total of 22 days. The features have been built using the domain knowledge and, for a lack of time, generic feature selection methods have not been performed. However, in order to predict the sleeping activity status in real time, only the features which do not contain information after the motion have been used. The final dataset contains 14 nights, 8 of them have been used to build the models and the other 6 to evaluate the best classifier obtained with a 10-fold CV. Finally, the information is post-processed with a simple sequential track in order to obtain the "*go to sleep time*" and the "*wake up time*" and to compute the number of sleeping activity hours and the number of resting hours.

The results obtained with the MLS clearly show a great improvement in performance with respect to the ruled-based system. In addition, the MLS presented is able to easily obtain the "*go to sleep time*", the "*wake up time*", the number of sleeping activity hours and the number of resting hours, as the requirements specify; thus fulfilling the main goals of the project. Owing to the satisfactory results, the MLS will be integrated in the SB-TMHSS developed in eKauri project. More specifically, it will replace the ruled-based system in "the sleeping status" of the AR module.

Nevertheless, some problems were encountered during system development. These were solved as best we could during the project. Below, these problems, the solutions taken in the project and other possible solutions will be presented. The latter

as possible future work.

Our sensor-based system differs a lot from other similar systems, due to the fact that our system only uses PIR and doors sensors. The fact that the system only uses these two types of sensors limits the machine learning models which can be used in the intelligent module (IM). For this reason, in the project, discriminative models are used rather than generative models as many authors.

The data used to build the models was obtained from a real-world environment with real end-users, thus ensuring the data reflect the complexities of the real-world. At the same time, using real data leads two main downsides. On the one hand, the number of days which the users answered the questionnaire was limited, compared to the number of days which the telemonitoring period lasted. On the other hand, the information given by the users, through the questionnaire answers, was not as accurate as the information which could be obtained in a lab. Despite these inconveniences, we strongly believe that it is important to train the models with real-world data.

As for the future work, we envisage three main directions that may be gone through. (1) We will study the possibility of adding press mats and electrical switches in order to make the results more accurate, and use generative models. (2) We are currently setting-up a system to accurately obtain the ground truth from the end-users. (3) We are also interested in studying whether we can generalize the proposed approach to adopt it for all the users of the system, or if we have to use a personalized approach for each different user.

## 5.2 TFG conclusions

This project is the first time I have faced a whole ML problem, by myself. I have learnt a lot by reading papers and books but also with the feedback of my workmates and my tutor. In this section, I hope to explain my previous experience with ML since it has been the reason why I chose this project as my TFG, and then to summarise the main aspects that this project has allowed me to learn.

I was fascinated the first time I heard about ML and about the real-world solutions that can solve. I had my first contact with this field during last year in both subjects "Visió Artificial" and "Taller de nous usos de la informàtica". At the end of the first semester, when I finished these subjects, I had the opportunity to start an internship at BDigital. There I joined to the eHealth department and started to put into practice what I learnt in these subjects. The eKauri project was the first big project that I participated in. Firstly, testing and improving some models of "the localization" in AR module [14]. Secondly, building a MLS to the sleeping activity recognition, which will be integrated to "the sleeping status" in AR module. Thus, I decided to present as TFG the work done to develop this MLS, which has expanded my knowledge of ML in many ways. The main aspect will be explained below.

Firstly, this project has been a small taste of the research world, for me. I has entered in the fascinating world of papers and articles. It is true that I had to

read some for the "Visió Artificial" project but in this case the paper was given to me and I had to read it with the purpose of learning about the project and then, explaining it to my classmates. Whereas, the papers, which I read for this project, I had to search for and read in order to know the state-of-art of the project and how to make use of this knowledge for our own project.

Secondly, in my previous experiences with ML, I hadworked with a clean dataset, but in this case I faced to a raw dataset with a lot of noise. Cleaning the dataset took me longer than I expected, really I had not scheduled it. I can firmly claim that it has not been my favourite part of the project but I have learnt that a clean dataset is essential for good performance of the models.

Thirdly, I have read and learnt a lot about feature extraction and selection, but for lack of time I could only apply one of the feature selection methods. Regarding the feature extraction, it is the first time that I have proposed a features by myself. It has been interesting creating features using the domain knowledge, but I still want to implement some generic methods, which have not been possible, once again, for lack of time.

Fourthly, I have been able to make use of some ML models, learnt in the subject mentioned, such as RF or $k$-NN; I have also learnt about a new method, the SVM. Previously, I had only heard the name of SVM, so I had to learn how it works in order to use it in this project.

Finally, I have realised that presenting results in the right way is as import as obtaining a good ones, therefore I have to learn some basic skills about how to present the results, especially visual methods. In addition to cleaning the dataset, it has been one of the new things that I have learnt from zero in this project. I regret not having the time to learn some visualization techniques.

Although there have been many bumps in the project and the lack of time has not allowed me to explore and apply everything that have learnt, I am please with the experience. I am very grateful to have participated in a project like eKauri and that my work will be integrated into the system. As I said before I hope that this experience is just the beginning of my contribution to the research world.

# References


[1] U. N. D. of Economic and P. D. Social Affairs, *World Population Prospects: The 2015 Revision*. United Nations Publications, 2015.

[2] S. Katz, A. B. Ford, R. W. Moskowitz, B. A. Jackson, and M. W. Jaffe, "Studies of illness in the aged: the index of adl: a standardized measure of biological and psychosocial function," *Jama*, vol. 185, no. 12, pp. 914–919, 1963.

[3] I. Lobentanz, S. Asenbaum, K. Vass, C. Sauter, G. Klösch, H. Kollegger, W. Kristoferitsch, and J. Zeitlhofer, "Factors influencing quality of life in multiple sclerosis patients: disability, depressive mood, fatigue and sleep quality," *Acta Neurologica Scandinavica*, vol. 110, no. 1, pp. 6–13, 2004.

[4] J. Zeitlhofer, A. Schmeiser-Rieder, G. Tribl, A. Rosenberger, J. Bolitschek, G. Kapfhammer, B. Saletu, H. Katschnig, B. Holzinger, R. Popovic, *et al.*, "Sleep and quality of life in the austrian population," *Acta Neurologica Scandinavica*, vol. 102, no. 4, pp. 249–257, 2000.

[5] G. A. Meijer, K. R. Westerterp, F. M. Verhoeven, H. Koper, and F. Ten Hoor, "Methods to assess physical activity with special reference to motion sensors and accelerometers," *Biomedical Engineering, IEEE Transactions on*, vol. 38, no. 3, pp. 221–229, 1991.

[6] S. Warren, "Wearable and wireless: Distributed, sensor-based telemonitoring systems for state of health," *Canadian Journal of Animal Science*, vol. 80, pp. 381–392, 2000.

[7] T. Van Kasteren, A. Noulas, G. Englebienne, and B. Kröse, "Accurate activity recognition in a home setting," in *Proceedings of the 10th international conference on Ubiquitous computing*, pp. 1–9, ACM, 2008.

[8] F. J. Ordónez, P. de Toledo, and A. Sanchis, "Activity recognition using hybrid generative/discriminative models on home environments using binary sensors," *Sensors*, vol. 13, no. 5, pp. 5460–5477, 2013.

[9] N. C. Krishnan and D. J. Cook, "Activity recognition on streaming sensor data," *Pervasive and mobile computing*, vol. 10, pp. 138–154, 2014.

[10] T. Nef, P. Urwyler, M. Büchler, I. Tarnanas, R. Stucki, D. Cazzoli, R. Müri, and U. Mosimann, "Evaluation of three state-of-the-art classifiers for recognition of activities of daily living from smart home ambient data," *Sensors*, vol. 15, no. 5, pp. 11725–11740, 2015.

[11] T. L. van Kasteren, G. Englebienne, and B. J. Kröse, "Hierarchical activity recognition using automatically clustered actions," in *Ambient Intelligence*, pp. 82–91, Springer, 2011.

[12] S. Chernbumroong, S. Cang, A. Atkins, and H. Yu, "Elderly activities recognition and classification for applications in assisted living," *Expert Systems with Applications*, vol. 40, no. 5, pp. 1662–1674, 2013.

[13] J. M. Fernández, S. Torrellas, S. Dauwalder, M. Sola, E. Vargiu, and F. Miralles, "Ambient-intelligence trigger markup language: A new approach to ambient intelligence rule definition.," in *DART@ AI* IA*, pp. 1–12, Citeseer, 2013.

[14] X. Rafael-Palou, C. Zambrana, E. Vargiu, and F. Miralles, "Home-based activity monitoring of elderly people through a hierarchical approach," in *Information and Communication Technologies for Ageing Well and e-Health*, pp. 145–161, Springer International Publishing, 2015.

[15] S. Kotsiantis, D. Kanellopoulos, and P. Pintelas, "Data preprocessing for supervised leaning," *International Journal of Computer Science*, vol. 1, no. 2, pp. 111–117, 2006.

[16] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *The Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[17] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[18] M. A. Hall, *Correlation-based feature selection for machine learning*. PhD thesis, The University of Waikato, 1999.

[19] X. Geng, T.-Y. Liu, T. Qin, and H. Li, "Feature selection for ranking," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 407–414, ACM, 2007.

[20] T. M. Cover, "The best two independent measurements are not the two best," *Systems, Man and Cybernetics, IEEE Transactions on*, no. 1, pp. 116–117, 1974.

[21] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial intelligence*, vol. 97, no. 1, pp. 273–324, 1997.

[22] D. Koller and M. Sahami, "Toward optimal feature selection," 1996.

[23] J. Bi, K. Bennett, M. Embrechts, C. Breneman, and M. Song, "Dimensionality reduction via sparse support vector machines," *The Journal of Machine Learning Research*, vol. 3, pp. 1229–1243, 2003.

[24] S. Piramuthu, "Evaluating feature selection methods for learning in data mining applications," *European journal of operational research*, vol. 156, no. 2, pp. 483–494, 2004.

[25] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 1, no. 1, pp. 67–82, 1997.

[26] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[27] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[28] T. K. Ho, "Random decision forests," in *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, vol. 1, pp. 278–282, IEEE, 1995.

[29] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.