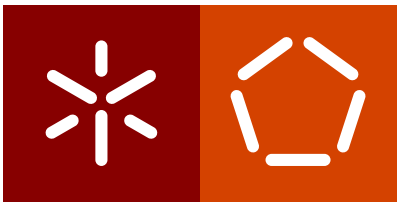
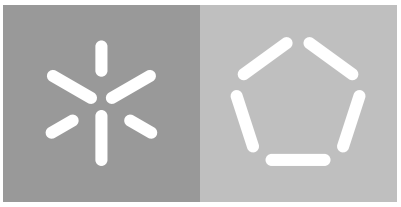


Universidade do Minho
Escola de Engenharia
Departamento de Informática

José Alexandre Da Silva Mirra

Enhance Smart Home capabilities by learning from usage patterns and IoT devices

October 2018



Universidade do Minho
Escola de Engenharia
Departamento de Informática

José Alexandre Da Silva Mirra

Enhance Smart Home capabilities by learning from usage patterns and IoT devices

Master dissertation
Master Degree in Computer Science

Dissertation supervised by
Cesar Analide
Fábio Silva

October 2018

ACKNOWLEDGEMENTS

I would first like to thank my master project advisor Cesar Analide of Departmento de Informática at University of Minho. The door to Professor Analide office were always open whenever I ran into a trouble spot or had a question about my research or writing. He consistently allowed this paper to be my own work, but steered me in the right the direction whenever he thought I needed it.

I would also like to acknowledge Fábio Siva of Centro Algoritmi at University of Minho as the second reader of this master project, and I am gratefully indebted for his very valuable comments on this work.

Finally, I must express my very profound gratitude to my parents and to my girlfriend and friends for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this project. This accomplishment would not have been possible without them. Thank you.

Author

José Mirra

ABSTRACT

Domotics represent a field of consumer electronics related primarily to home automation. Although smart environments have existed for decades and even for longer in the imaginary of people and sci-fi, the new age of Internet of Things (IoT) and the low-cost Do It Yourself (DIY) electronics/maker market has brought smart homes and the understanding of domotics closer to everyone.

In recent years, the interest in the fields of domotics and IoT has increased. This recent academic and industrial interest has contributed to the evolution of the Smart Home concept, being more and more appealing to our civilization. This has prompted new commercial solutions on the market, which are discovering ways to expand and increase their own value by integrating new features, especially from the IoT market.

The proliferation of IoT devices leads to new sources of information. In addition, the relation between the environment occupant and the environment is responsible for adding value to this data. Afterwards, the data can then be used in a meaningful way, by machine learning algorithms to learn usage patterns. Furthermore, this data may or may be unrelated to the data incoming from sensors.

This Masters Project will focus on learning strategies to allow Smart Homes to become intelligent, in a sense that they anticipate needs and actions, thus enabling extended features to the home automation system. The user should be able to give feedback on his "feeling" regarding the decision making and suggestions, possible by our system implementation.

Keywords: Internet of Things, Machine Learning, Data Mining, Domotics, Smart Home, Smart Environments.

RESUMO

Domótica é um campo da eletrónica de consumo relacionado principalmente com automatização de casas. Embora a domótica exista há décadas, e ainda há mais tempo nas nossas imaginações e na ficção científica, a nova era de Internet das Coisas (IdC) e o mercado de aparelhos eletrónicos de baixo custo permitiu a aproximação da domótica com todos nós.

Com o aumento de interesse nas áreas da domótica e IdC, soluções comerciais procuram formas de expandir e aumentar o seu valor no mercado, integrando novas funcionalidades, especialmente oriundas do mercado IdC.

A proliferação dos dispositivos IdC conduz a novas fontes de dados. Através da interacção do utilizador com o ambiente, os dados são gerados, podendo assim serem usados de forma significativa, por algoritmos de aprendizagem máquina, conduzindo à aprendizagem de padrões de utilização . Os dados, podem ou não, estar relacionados com informação oriunda dos sensores.

Este Projeto de Mestrado irá focar em estratégias de aprendizagem para permitir que as Casas Inteligentes se tornem realmente inteligentes, no sentido em que elas antecipam necessidades e ações, permitindo assim, uma ampliação de funcionalidades para o sistema de automação residencial.

O utilizador deve ser capaz de dar feedback sobre o seu "sentimento" em relação à tomada de decisão e às sugestões do sistema, sendo possível através da nossa implementação.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1

INTRODUCTION

In this chapter it is intended to present the context of the dissertation theme, the motivation behind the purpose of this project, the different objectives and for last, the development process. With this in mind, this chapter aims to state the general topic and provide some background.

1.1 CONTEXT

Home automation or Domotics is a word well known in today's society. Domotics has started as simple as remotely controlling lights and appliances, with the introduction of x10 protocol at 1975, the first general purpose domotics network technology. (Rye, 1999)

We are witnessing a new era of Internet of Things (IoT). Generally speaking, IoT refers to the networked interconnection of everyday objects, which are often equipped with ubiquitous intelligence. IoT will increase the ubiquity of the Internet by integrating every object for interaction via embedded systems, which leads to a highly distributed network of devices communicating with human beings as well as other devices. Thanks to rapid advances in underlying technologies, IoT is opening opportunities for a large number of novel applications that promise to improve the quality of our lives. In recent years, IoT has gained much attention from researchers and practitioners from around the world. (Xia et al., 2012)

This has prompted a new phase, seeing more and more complex projects in the domotics field. One example, is the *The Smart Kitchen Project* of Soucek et al. (2000). This requires a advanced network technology.

There are several standards for home automation networking, from cabled technologies to wireless protocols. Wireless sensor and actuator networks have gained high momentum, receiving significant attention from academia and industry. One of the primary application domains of this technology is home automation. (Gomez and Paradells, 2010)

Although the communication protocols are standard, most are not open to the public and are behind a "pay wall". This means that access to information on how they work and what is needed to interact with systems connected on the same network might need the payment of a license.

ONLY, the domotic system from Enancer Electrónica S.A., has a proprietary protocol. Although it is proprietary, anyone can develop for the system if permission is requested to the company and granted. There are no royalties or any other type of fees associated with developing for the system. The protocol has evolved with the needs shown by customers along the years. Backwards compatibility has been a major concern when designing the protocol. Another main concern was to allow for a simple and easy installation, customizable by the owner of the infrastructure; this is relevant as some systems do block the ability to make changes to the system so that changes can only be made by certified (and payed) representatives of the system. The ability to change the behavior, either on the hardware, or by using the web app or the openly available REST API, is also a large part of what makes ONLY distinguish itself. The system is also meant to be simple to install and to use. (S.A, 2016)

There are several protocols for the communication between IoT devices. There are also specific protocols for IoT devices communicating through the Web or to cloud services, for example Message Queuing Telemetry Transport (MQTT) Hunkeler et al. (2008). Although IoT devices do not necessarily need cloud services, there is the common misconception that this is the case. Although some popular IoT devices do use the cloud, it is not a requirement. One of the reasons for the use of cloud services in IoT devices is that it is a way to: a) certify that the device is a valid device and b) to be able to communicate remotely between devices of different types/networks through a common web service and/or API (e.g. If This Then That (IFTTT) Ovadia (2014)).

Finally, we have seen recently in some systems like Alexa West (2016) and Google Home GoogleLLC, the usage of data collection for inferring behaviours and expectations from users, although some of this developments are still in its infancy in terms of their capabilities.

Given this context, we see possibilities for improve and integrate different components from the bond between these 3 areas of interest: Domotics, IoT and Artificial Intelligence (AI).

1.2 MOTIVATION

The possibilities are immense with the arrival of home automation, starting with ease accessibility to power saving. Although, there are still some social barriers about this field. (Balta-Ozkan et al., 2013) The necessity of a non intrusive system is real, as well as security. Machine learning can be useful, Machine learning engines can monitor incoming and outgoing IoT device traffic to create a profile that determines the normal behavior of the IoT ecosystem. From there, detecting threats will boil down to discovering traffic and exchanges that do not fall within the established normal behavior. Alarms can be sent to device owners to warn them about potential risks and suspicious behavior. Machine learning can help bring lightweight endpoint protection to IoT devices (they do not have the processing power and storage capacity to run security solutions and store huge databases of threat and malware signatures to protect them against threats.)

Smart Homes are being more coveted nowadays, a recent research of Li et al. (2016) has shown a gain interest on Smart Homes but a decrease in home automation over time, this is simple because that Smart Home technologies can do more than automate a house (Li et al., 2016). This lead to more and more companies diving in this area of business, for instance tech giants like Google LLC, Samsung, Apple and more (Alam et al., 2012). The need of extending capabilities is essential to make a difference in the market.

ONLY Smart Home brand, owned by Enancer Electrónica S.A, for now , is an home automation system. It is possible to control lighting, climatization, audio and security. ONLY Smart Home can be monitored and controlled remotely, over the Internet, by an app. The status of all connected devices and sensors, such as those for temperature or open windows can be viewed. The system is easily configure by the user, as well as set scenarios.

The increase of IoT devices has lead to an enormous amount of data that we have about our environment. This data when worked by machine learning algorithms, can be extremely useful to people needs and wants. Artificial intelligence comes in handy in this field. AI is specifically adept at finding and establishing patterns, especially when it's fed huge amounts of data. Data availability is not a concern in IoT, nevertheless we need to guarantee the quality of such data. With this knowledge, it is intended to achieve Smart Homes.

1.3 OBJECTIVES

The goal of this masters project is to enhance domotics system capable of decision making and/or suggestions based on the usage patterns of the system, without being intrusive to the user. In this way, the present-day capabilities of Domotics systems are enriched, approaching them to Smart Homes, and not only home automation.

Ultimately the goals are:

- To further understand the problems and difficulties of adding intelligence to a Smart Home environment, including privacy and user experience concerns;
- to prepare and develop a Smart Home/Home intelligence system capable of coexisting into a real world existing home automation system.

To fulfill the goals proposed, we need :

- To gather data from the occupant of the environment. This data is controlled by the user of the system,
- Define methods to extract knowledge of the data gathered,
- Define and Apply machine learning algorithms to learn usage patterns,
- Display the suggestions to the user.

It is really important to balance the system between features and security.

1.4 PLANNED WORK

Next, we have the schedule of the work planed, figure 1.

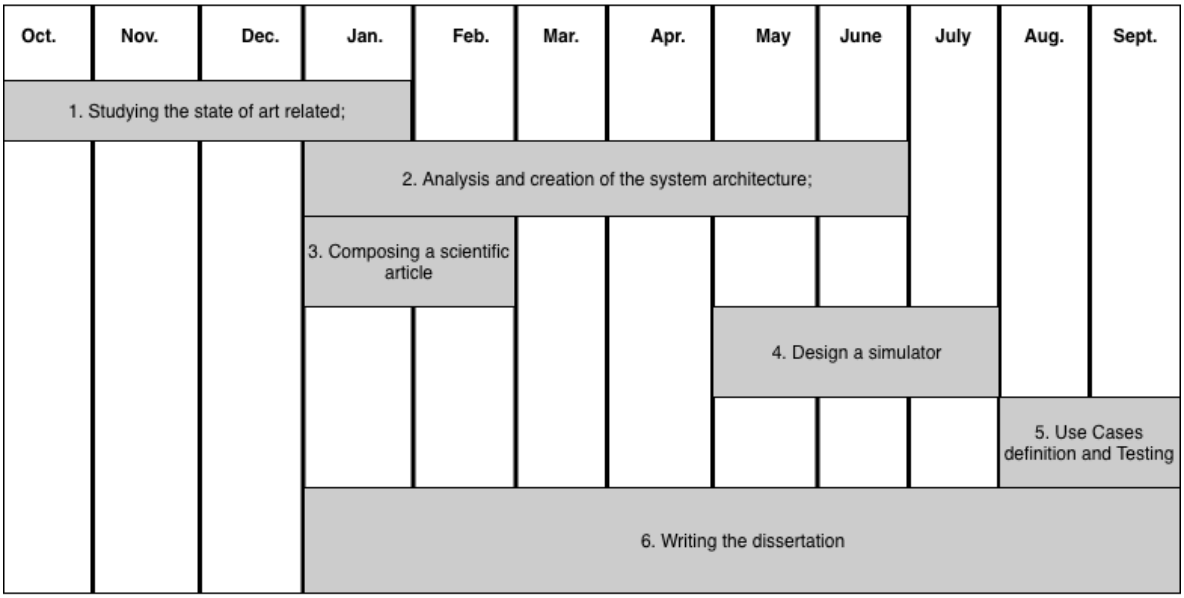


Figure 1: Work Schedule

- The first task was to read the related work and write on the state of the art within the study area. It is important to search for studies on the "feeling" users have when things are too intrusive on their lives and what is the limit and/or what they need to feel more in control of the system. It should also be possible for users to restrict the collected data; on the absence of data, the system should adapt to stay relevant. Research different algorithms to achieve a Smart Home is crucial, with this in mid, machine learning algorithms are expected.

- After understanding the subtleties of Machine Learning in Home Automation, Task 2 will develop the necessary means to prototype and carry out experiments on the subject. This will include an application where experiments can be carried out without the problems and subtleties of dealing with real world sensors and user choice/customization.

- Task 3 will consist in composing a scientific article about a system architecture to solve this project.

- No real data were provided to work with, leveraging the need to simulate data. This task 4 consist in the definition and creation of a simulated environment.

- With the system developed and with a simulated environment with sources of data, we can then test our system. In this task, we have created a case study to correctly test our system capability.

- Task 6 will be done along the full academic year, being supervised by both the academia and the industry partners.

2

STATE OF THE ART

This chapter presents an analysis of the areas of the project domain. In this way, it became necessary to elaborate an analysis of concepts, methodologies, related themes and applications already developed. The methodology used to elaborate the research of the relevant concepts discussed in this chapter, consisted mainly in the analysis of scientific papers presented at relevant conferences, scientific journals, scientific books and practical applications. The terms searched were Internet of Things, Smart Home, Smart environments, intelligent agents and machine learning . The purpose of this analysis is not only to gather relevant information, but also to reflect in order to understand the project area.

2.1 INTERNET OF THINGS

The Internet will continue to become ever more central to everyday life and work, but there is a new but complementary vision for an Internet of Things, which will connect billions of objects (things) like sensors, monitors, and Radio-frequency identification (RFID) devices to the Internet at a scale that far outstrips use of the Internet as we know it, and will have enormous social and economic implications (Dutton, 2013).

From a conceptual standpoint, IoT builds on three pillars, related to the ability of smart objects to:

- be identifiable (anything identifies itself),
- to communicate (anything communicates) and

- to interact (anything interacts)

either among themselves, building networks of interconnected objects, or with end-users or other entities in the network.The Internet of Things is emerging as one of the major trends shaping the development of technologies in the information and communicate technologies (ICT) sector at large (Strategy and Unit, 2005).

This trend has created a variety of applications areas based on data, represented in Figure 2 by Gubbi et al. (2013).

Figure 2: Internet of Things schematic showing the end users and application areas based on data.

As Gubbi et al. (2013) mentions, the Internet revolution led to the interconnection between people at an unprecedented scale and pace and the next revolution will be the interconnection between objects to create a smart environment.

Smart Objects

The combination of the Internet and emerging technologies such as near-field communications, real-time localization, and embedded sensors lets us transform everyday objects into smart objects that can understand and react to their environment. Such objects are building blocks for the Internet of Things and enable novel computing applications. The IoT will be based on the notion of "smart objects", or, simply, "things", which will complement

the existing entities in the Internet domain (hosts, terminals, routers, etc.) (Kortuem et al., 2010).

Miorandi et al. (2012) defines smart objects (or things) as entities that:

- Have a physical embodiment and a set of associated physical features (e.g., size, shape, etc.).
- Have a minimal set of communication functionalities, such as the ability to be discovered and to accept incoming messages and reply to them.
- Possess a unique identifier.
- Are associated to at least one name and one address. The name is a human-readable description of the object and can be used for reasoning purposes. The address is a machine-readable string that can be used to communicate to the object.1
- Possess some basic computing capabilities. This can range from the ability to match an incoming message to a given footprint (as in passive RFIDs) to the ability of performing rather complex computations, including service discovery and network management tasks.
- May possess means to sense physical phenomena (e.g., temperature, light, electromagnetic radiation level) or to trigger actions having an effect on the physical reality (actuators).

IoT can be directly associated with Ambient intelligence. Ambient intelligence is going to change the way technology is perceived by the users: as these devices become smaller, more connected, and more integrated into the environment, the technology gradually disappears into the surroundings until only the user interface remains perceivable by users. Users will interact directly or indirectly with the interfaces of the systems, and the system will respond to people presence, actions, and decisions, offering integrated solutions and improving their quality of experience (QoE). To this end, mobile information systems are crucial for the ambient intelligence, since the majority of the sensors and actuators rely on mobile/wireless technologies and on distributed information systems and big data to collect, store, and process the data.

In Ambient Intelligence (AmI), environments rich in sensing/computing/actuation capabilities are designed so to respond in an intelligent way to the presence of users, thereby supporting them in carrying out specific tasks. Ambient intelligence builds upon the ubiquitous computing concept, loosely defined as the embedding of computational devices into the environment. Ubiquitous computing provides therefore the distributed infrastructure necessary to enable the development of AmI applications.

2.2 AMBIENT INTELLIGENCE

Ambient Intelligence is the vision that techonology will become invisible, embedded in our natural surroundings, present whenever we need it (Weber et al., 2005). Computers are not typically standalone devices in the environment but many man-made and organic systems have built-in intelligence and computing abilities. The current Internet of Things (IoT) development, which involves outfitting just about any type of object imaginable with computing ability and connectivity, is leading us to embedded computing.

This master project focus on Smart Home, a case study of Ambient Intelligence.

2.2.1 *Smart Home*

The term Smart Home is well known in today's society, although is definition is misunderstood with home automation. The ability to control blinds, doors, lights, heating and even surveillance systems through a smart device, simply means an automated home, shown in figure 3.

Figure 3: Illustration of Home automation from Qureshi (2017)

An automated home allows control of devices from an external source such as a smart phone and provides limited action from other sensors such as motion sensors. A Smart Home makes it is decisions using multiple factors and sensors.

According to Dixit and Naik (2014), a Smart Home is defined as a living or working space that interacts in a natural way and adapts to the occupant. Adaptation refers to the fact that it learns to recognize and change itself depending on the identity and activity undertaken by the occupant with minimal intervention from the occupant. Hence, a Smart Home agent must be able to predict the mobility patterns and device usages of the inhabitants. Using these predictions, the home can accurately route messages and multimedia information, and can automate activities that would otherwise be manually performed by the inhabitants. The Smart Home behaves as a rational agent, perceiving the state of the home through sensors and acting on the environment through effectors. The goal of the Smart Homes is to maximize comfort and safety, optimize energy usage and eliminate strenuous repetitive activities. Prediction Algorithm helps in prediction of next probable state of the occupant and is a key component in developing an active Smart Home. (Dixit and Naik, 2014).

The broadest aspect of a Smart Home is to adapt to the occupant lifestyle, providing smart decisions to facilitate his daily routines. Smart Homes are made of many data gathering tools and sensors. With more data, more intelligent decisions are made based on that data. This explain the reason of IoT being present in this subject. More and more articles are being published with the keywords *Internet of Things*, *Connected Devices* and *Smart Home*. According to Sharma et al. (2017), IoT is the magic dust that turns the automated home into the Smart Home. IoT connects everyday objects (often called as ”things”) to a network, enabling those objects to complete tasks and communicate with each other, with no user input. A Smart Home is a domain of IoT, which is the network of physical devices that provide electronic, sensor, software, and network connectivity inside a home.

The benefits of using Smart Home applications based on IoT are evident and compelling. The article of Alaa et al. (2017), lists a few of the advantages reported in the literature, which are grouped into categories depending on similar benefits (Figure 4).

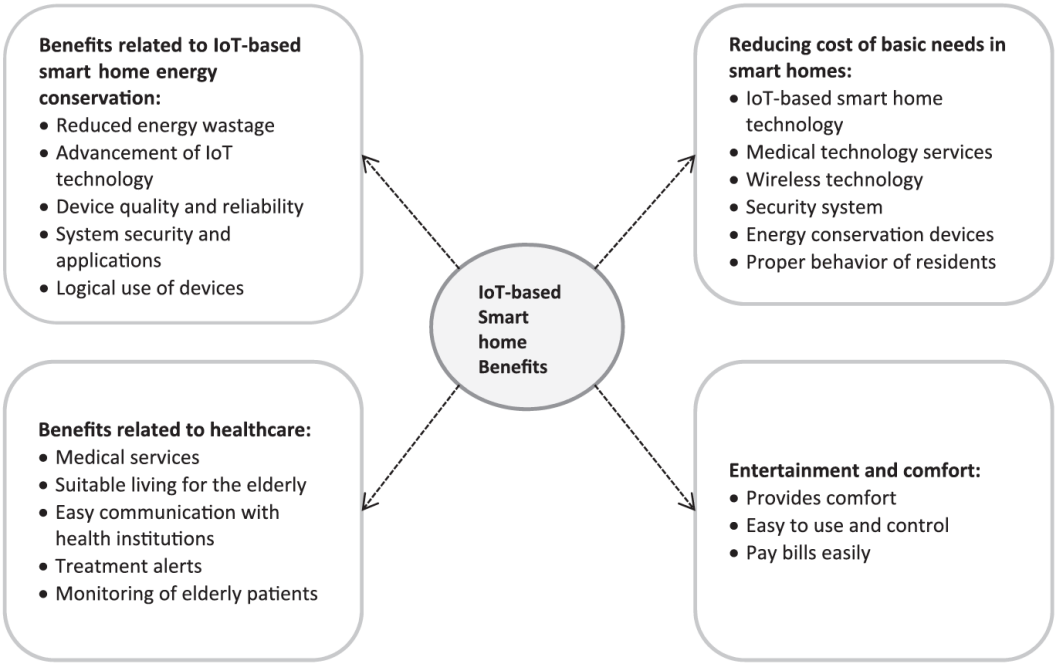


Figure 4: Categories of benefits of Smart Home applications based on IoT

2.2.2 *Social barriers in adoption of Smart Homes*

The increase of IoT devices ("things") available in our environment, has contributed to the evolution of domotics. Although this evolution brought a wide variety of benefits (Figure 4), there are some impasses regarding the social embrace of this technology.

In order to tailor its systems to best support the inhabitant's lifestyle, a Smart Home may collect information about them. A study made by Balta-Ozkan et al. (2013) explains the most important factors contributing to the barriers regarding adopting Smart Home technology. The public's main concerns pertain to: loss of control and apathy; reliability; viewing Smart Home technology as divisive, exclusive or irrelevant; privacy and data security; cost; and trust (represented in table 1).

Table 1: Social barriers summary presented by Balta-Ozkan et al. (2013)

		Literature		Experts	Public
		Smart home	Demand-side		
Fit to current and changing lifestyles	Integration of technology and services into the design, lifestyle and general sense of home	√			
	Meeting the evolving needs, demands and preferences of its occupants	√			
	Accommodating the integration of new technological components	√			
	Accepting automation and contributing to network flexibility		√		
	Convenience more important than financial savings		√		√
	Role of behavioural scientists to understand the 'fit' of consumers, products and business models			√	
	Technical gap between expectations and current solutions	√			
	Loss of control and apathy				√
	Smart home services as non-essential, luxurious, or '*gadgetry*'				√
	Smart technology as divisive (exclusive to tenants, elderly, computer illiterate, smart phone users, people living in older properties)				√
	Smart technology leaving people '*constantly worrying*' and feeling guilty				√
	Smart homes making more affluent people less conscientious regarding energy saving				√
Administration	Expertise for operational and management needs of smart homes	√			
	Knowledge deficit		√		
	Response fatigue		√	√	
	Satisficing behaviour in switching patterns		√		
	Difficult user interfaces		√		
	Reducing complexity			√	
	Less control over electricity use		√		
	Information fatigue for elderly in particular				√
Interoperability	Communicating with other devices and technologies	√			
	Interference of devices with each other in a home setting			√	
Reliability	Malfunctioning	√			√
	Inference of householders' desired outcome	√			
	Avoiding unintended consequences			√	
	Behaviour recognition forming key aspect of smart homes			√	
	Sensors going off by mistake				√
	Due to break down of remote control units house going in limbo				√
	Communications network breaking down and other systems getting out of control				√
Privacy and security	Ensuring safety and security of private data	√			
	Violations of privacy		√		
	Data falling into wrong hands			√	√
	Combining two sets of innocent data leading to 'non-innocent' data			√	
	Tension between interoperability and security			√	
	Big brother-like monitoring as too intrusive				√
	Concerns over third parties knowing daily routines and occupancy				√
	Systems being compromised				√
	Companies responsible for smart home services selling on personal data				√
	Lack of perceived privacy would not worth it for lower bills				√
	Smart health services invading privacy by leading to consumers being bombarded with marketing				√
Trust	Lack of trust that financial savings made by utility companies will be passed onto the consumers		√		√
	Prioritisation of issues by the UK Government				√
Costs	Transaction costs to seek out price and consumer information		√		
	Installation costs		√		√
	High repair and maintenance costs				√

This impasses, creates a space for improvement in this area. In this master project, we aim to neutralize this social stigma. The data collected by the system is controlled by the user, in this way, the feeling of control is maintain. These social barriers need not prevent the development of the Smart Home market.

2.2.3 *Smart Environments*

As the name says, a Smart Environment is defined as one that is able to acquire and apply knowledge about the environment and its inhabitants in order to improve their experience

in that environment Youngblood et al. (2005). As we can see in Figure 5, sensors monitor the environment using physical components and make information available through the communication layer. The database stores this information while other information components process the raw information into more useful knowledge. New information is presented to the decision-making algorithms upon request or by prior arrangement. Action execution flows top-down. The decision action is communicated to the services layers which record the action and communicate it to the physical components. The physical layer performs the action with the help of actuators or device controllers, thus changing the state of the world and triggering a new perception.

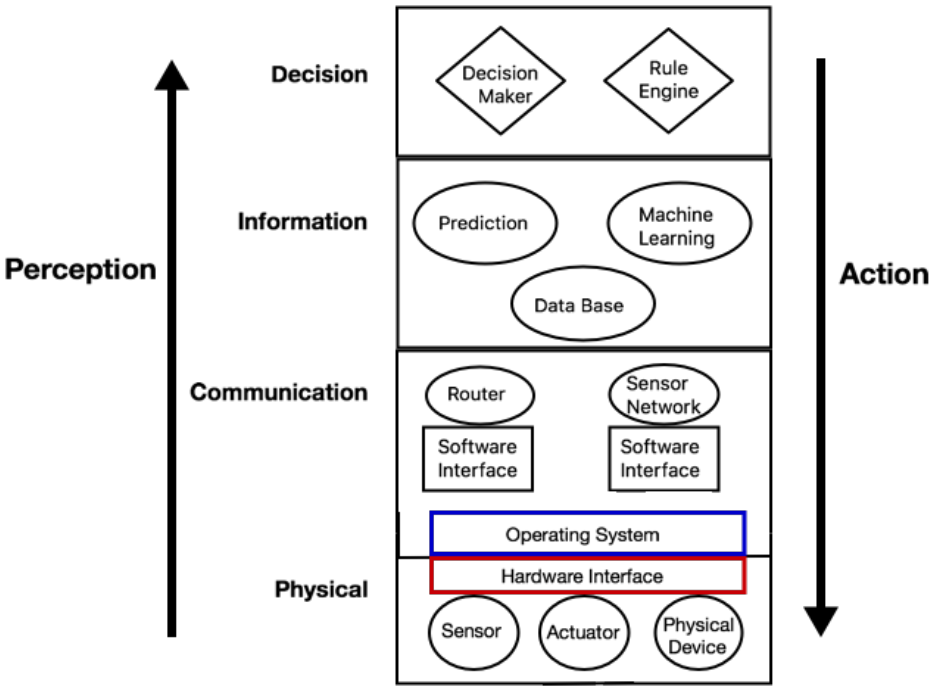


Figure 5: Components of a Smart Environment

Automation in a smart environment can be viewed as a cycle of perceiving the state of the environment, reasoning about the state together with task goals and outcomes of possible actions, and acting upon the environment to change the state. As smart environment research is being conducted in real-world physical environments, the use of physical components such as sensors, controllers, and smart devices is vital. This is because sensors enable us to observe, monitor and interact with the physical world in real time, and also allow us to take appropriate actions. (Cook and Das, 2007) The design and modeling of a smart environment can be abstracted to an intelligent agent paradigm.

2.2.4 *Intelligent Agents*

An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors. A human agent has eyes, ears, and other organs for sensors, and hands, legs, mouth, and other body parts for effectors. A robotic agent substitutes cameras and infrared range finders for the sensors and various motors for the effectors (Russell and Norvig, 2003). A software agent has encoded bit strings as its percepts and actions. A generic agent is diagrammed in Figure 6.

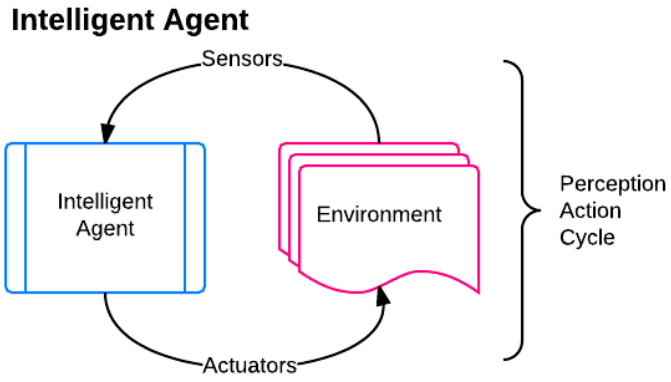


Figure 6: Intelligent Agent Architecture

In the context of smart-homes, agents represent the entities in the environment and can be considered autonomous, adaptive, context-aware and capable of making decisions about actions (behaviors) based on their observations. They can be the software interface of the ubiquitous devices that offer their services or the software interface from the user-side that demands such services. In other terms, agents are called soft-sensors in smart environments and will be designed to learn from previous experiences and to reason about their local information in order to improve their decisions and achieve their goals. In this way, a smart-home (or environment) can be naturally viewed as a multi-agent system with distributed intelligence. In multi agent systems (MAS), agents can additionally communicate and coordinate with each other as well as with their environment. MAS provide a valuable framework for intelligent control systems to learn building and occupancy trends. (Klein et al., 2012)

An example of this is the *MavHome* project by Cook et al. (2003), an home that acts as a rational agent which aims to maximizes comfort and productivity of its inhabitants and

minimizes operation cost. In order to achieve this goals, the house must be able to predict, reason about, and adapt to its inhabitants.

Intelligent agents can benefit clearly from machine learning. Machine learning in single and multi-agent systems is a significant and promising topic in Artificial Intelligence. Intelligent agents and agent based computer systems represent an important, fundamentally new way of dealing with many important software application problems for which mainstream computer science techniques offer no obvious solution. Many of these new problems are due to the recent dynamic and distributed nature of both data and applications.(Panayiotopoulos and Zacharis, 2001)

2.3 MACHINE LEARNING

Machine learning has started as pattern recognition, where its origins are in engineering, however, machine learning as we known today as grew out of computer science. However, these activities can be viewed as two facets of the same field, and together they have undergone substantial development over the past ten years. The problem of searching for patterns in data is a fundamental one and has a long and successful history. (Bishop, 2006).

Predictive modeling is the general concept of building a model that is capable of making predictions. Typically, such a model includes a machine learning algorithm that learns certain properties from a training dataset in order to make those predictions. Machine learning is capable of generalizing information from large and dynamically changing data sets, and then detecting and extrapolating patterns in order to apply that information to new solutions and actions.

In the Smart Home context the data captured by IoT devices present in a environment can be mined to discover significant usage patterns, that can be called activities of daily living (ADLs). In order to achieve a reasonable predictive model, which an intelligent agent could use in the decision making process, the follow phases are essential:

1. **Prepare and transform data (Data pre-processing).**

2. **Choose a Machine learning algorithm.**

3. **Train, Test and re-evalute model.**

illustrated in Figure 7.

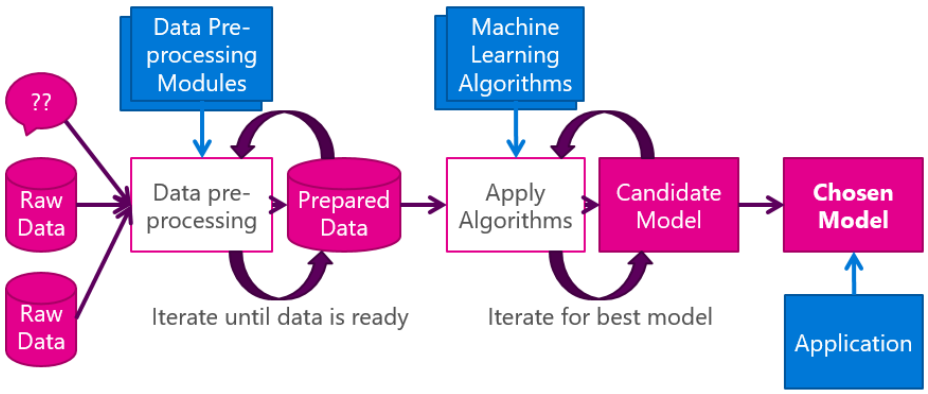


Figure 7: Predictive model architecture (Mokhtarian, 2016).

2.3.1 *Data pre-processing*

Data pre-processing is a very well step in data mining, however, it is of equal importance in machine learning. This is because machine learning is related to data mining, being a more automated and continuous version. pre-processing data is a crucial step prior to modeling since data preparation can make or break a model's predictive ability. A first objective in preparing the data set is to make things as easy as possible for the machine learning algorithm (Figure 8).

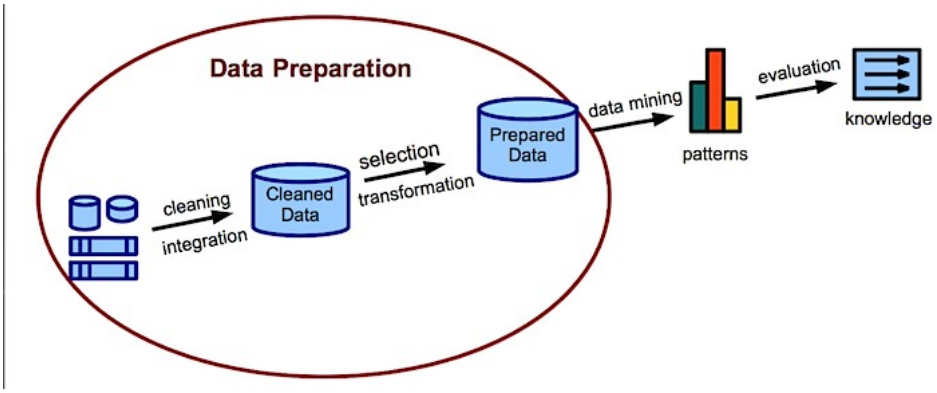


Figure 8: Data preparation process in extracting knowledge

According to Han et al. (2011), the common data pre-processing steps are:

- **Data Cleaning**: According to Rahm and Do (2000) data cleaning, also called data cleansing or scrubbing, deals with detecting and removing errors and inconsistencies from data in order to improve the quality of data. Real-world data tend to be incomplete, noisy, and inconsistent. There may be data instances that are incomplete and do not carry the data that is needed to address the problem, these instances may need to be removed. Other times it is necessary to fill the missing values. The most common imputing missing data methods are :

  1. Fill in the missing value manually.

  2. Use a global constant to fill in the missing value (e.g. "Unknown").

  3. Imputation (fill in the missing value using the feature mean or the most probable value).

  Additionally, there may be sensitive information in some of the attributes and these attributes may need to be anonymized or removed from the data entirely.

- **Data Discretization**: Data discretization or binning is defined as a process of converting continuous data attribute values into a finite set of intervals with minimal loss of information Jin et al. (2008). Histograms are an example of data binning used in order to observe underlying distributions.

  - **Equal-with Binning** - Equal-with Binning is the simplest method to discretize data and has often be applied as a means for producing nominal values from continuous on (Dougherty et al., 1995). It divides the range of values into N intervals of equal width, resulting in a uniform grid. Being A and B the lowest and highest values of the attribute, the width of intervals are defined by the following equation
  $$W = \frac{(B - A)}{N},$$

  - **Equal-depth Binning** - It divides the range into N intervals, each containing approximately the same number of samples (Han et al., 2011).

- **Data Integration**: As Lenzerini (2002) defines, data integration is the problem of combining data residing at different sources, and providing the user with a unified view of these data. The problem of designing data integration systems is important in current real world applications, given the context of this project, this phase will be

important, due to the fact that there are several data sources. Redundancy is another important issue. An attribute (such as annual revenue, for instance) may be redundant if it can be "derived" from another attribute or set of attributes. Inconsistencies in attribute or dimension naming can also cause redundancies in the resulting data set. Some redundancies can be detected by correlation analysis, for numerical attributes, we can evaluate the correlation between two attributes, A and B, by computing the correlation coefficient (also known as Pearson's product moment coefficient, named after its inventor, Karl Pearson). (Han et al., 2011), *page 68*.

$$r_{A,B} = \frac{\sum_{i=1}^{N}(a_i - \bar{A})(b_i - \bar{B})}{N\sigma_A\sigma_B} = \frac{\sum_{i=1}^{N}(a_i b_i) - N\bar{A}\bar{B}}{N\sigma_A\sigma_B}$$

- **Data Transformation**: In data transformation, the data are transformed or consolidated into forms appropriate for mining. As stated by Han et al. (2011) Data transformation can involve the following:

  - **Smoothing**, which works to remove noise from the data. Such techniques include binning, regression, and clustering.

  - **Aggregation**, where summary or aggregation operations are applied to the data. For example, the daily sales data may be aggregated so as to compute monthly and annual total amounts.

  - **Generalization** of the data, where low-level or "primitive" (raw) data are replaced by higher-level concepts through the use of concept hierarchies. For example, categorical attributes, like street, can be generalized to higher-level concepts, like city or country. Similarly, values for numerical attributes, like age, may be mapped to higher-level concepts, like youth, middle-aged, and senior.

  - **Normalization**, where the attribute data are scaled so as to fall within a small specified range, such as -1.0 to 1.0, or 0.0 to 1.0. Normalization is particularly useful for classification algorithms involving neural networks, or distance measurements such as nearest-neighbor classification and clustering. There are many methods for data normalization, for example, min-max normalization and z-score normalization (or zero-mean normalization).

  - **Attribute construction** (or feature construction), where new attributes are constructed and added from the given set of attributes to help the learning process. For example, calculation of the net price based on the gross price and VAT.

- **Data Reduction**: There may be far more selected data available than the one needed to work with. More data can result in much longer running times for algorithms and larger computational and memory requirements. A smaller representative sample of the selected data may be much faster for exploring and prototyping solutions before considering the whole dataset. There are a few data reduction methods, the author in ur Rehman et al. (2016) presents a hand full of them, they vary from pure dimension reduction techniques to compression-based data reduction methods and algorithms for pre-processing, cluster-level data deduplication, redundancy elimination, and implementation of network (graph) theory concepts.

2.3.2 *Machine learning algorithms*

Among the various frameworks in which pattern recognition has been traditionally formulated, the statistical approach has been most intensively studied and used in practice. More recently, learning algorithms such as neural networks and methods imported from statistical learning theory have been receiving increasing attention (Jain et al., 2000).

There are a enormous number of different learning algorithms, and the details about the most popular ones are perfect topics for separate articles and applications. However, they can be categorized as **supervised**, **unsupervised** and **reinforcement learning**.

Supervised learning is a machine learning paradigm for acquiring the input-output relationship information of a system based on a given set of paired input-output training samples. As the output is regarded as the label of the input data or the supervision, an input-output training sample is also called labeled training data, or supervised data. (Liu and Wu, 2012)

$$Y = f(x)$$

Y represents the label (output), given by a algorithm applied in a known input(x).

It is called supervised learning because the process of an algorithm learning from the training dataset can be thought of as a teacher supervising the learning process. It is known the correct answers, the algorithm iteratively makes predictions on the training data and is

corrected by the teacher. Learning stops when the algorithm achieves an acceptable level of performance. Supervised learning is used when building predictive models from known input and response data.

A large number of supervised learning methods are available in present time. Fortunately, Caruana and Niculescu-Mizil (2006) has presented a large-scale empirical comparison between ten supervised learning methods: Support Vector Machines, Neural networks, logistic regression, naive bayes, memory-based learning, random forests, decision trees, bagged trees, boosted trees, and boosted stumps. Considering this research, the following algorithms deserve to be highlighted, since they are best suited to this project.

1. **A Support Vector Machine (SVM)** is a classification method that samples hyperplanes which separate between two or multiple classes. Eventually, the hyperplane with the highest margin is retained, where "margin" is defined as the minimum distance from sample points to the hyperplane. The sample point(s) that form margin are called support vectors and establish the final SVM model.

2. **Bayes classifiers** are based on a statistical model (i.e., Bayes theorem: calculating posterior probabilities based on the prior probability and the so-called likelihood). A Naive Bayes classifier assumes that all attributes are conditionally independent, thereby, computing the likelihood is simplified to the product of the conditional probabilities of observing individual attributes given a particular class label.

3. **Artificial Neural Networks (ANN)** are graph-like classifiers that mimic the structure of a human or animal "brain" where the interconnected nodes represent the neurons. Although is not present in Caruana and Niculescu-Mizil (2006), **Deep artificial neural networks** are complex ANN's with a large number of layers that has recently surged has a need to solve more complex problems. In fact since 2009, supervised deep NNs have won many official international pattern recognition competitions. (Grosicki and El Abed, 2009),(Margner and El Abed, 2011).

4. **Decision tree** classifiers are tree like graphs, where nodes in the graph test certain conditions on a particular set of features, and branches split the decision towards the leaf nodes. Leaves represent lowest level in the graph and determine the class labels. Optimal tree are trained by minimizing impurity, or maximizing information gain.

Unsupervised Learning is a machine learning paradigm, opposing to supervised, input data has no corresponding output variables.

$$F(X_1, ..., X_p),$$

where $X = (X_1, ..., X_p)$ are variables X-space describing the problem.

The goal for unsupervised learning is to model the underlying structure or distribution in the data in order to learn more about the data.

These are called unsupervised learning because unlike supervised learning above there is no correct answers and there is no teacher. Algorithms are left to their own devises to discover and present the interesting structure in the data.

Unsupervised learning problems can be further grouped into clustering and association problems.

- **Clustering**: A clustering problem is where it is intended yo discover the inherent groupings in the data, such as grouping customers by purchasing behavior. Association: An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y.

- **Association**: Association rule problems is where it is intended to finds interesting associations (relationships, dependencies) in large sets of data items. The discovery of interesting associations provides a source of information often used by businesses for decision making. The concurrences found in a data set are represented in the form of association rules.

  $$LHS \Rightarrow RHS[support, confidence]$$

  where the left-hand side (LHS) implies the right-hand side (RHS), with a given value of support and confidence. (Cios et al., 2007)

Some popular examples of unsupervised machine learning algorithms are:

1. **K-means** is a clustering algorithm. The goal of this algorithm is to find groups in the data, with the number of groups represented by the variable K. The algorithm works iteratively to assign each data point to one of K groups based on the features that are provided (Hartigan and Wong, 1979).

2. **Apriori** is an association algorithm. Apriori is a classic algorithm for learning association rules. Apriori is designed to operate on databases containing transactions (for example, collections of items bought by customers, or details of a website frequentation) (Agrawal et al., 1994).

Reinforcement learning is learning what to do – how to map situations to actions – so as to maximize a numerical reward signal, Sutton and Barto (1998). The primary aim is to cast learning as a problem involving agents that interact with an environment, sense their state and the state of the environment, and choose actions based on these interactions. The twist in reinforcement learning is that the agent comes pre-equipped with goals that it seeks to satisfy. These goals are embodied in the influence of a 'numerical reward signal' on the way that the agent chooses actions, categorizes its sensations and changes its internal model of the environment. Despite the obvious connection of these terms to behavioral psychology, some of the more impressive applications of reinforcement learning have been in computer science and engineering applications. (Sutton and Barto, 1998). (Figure 9).

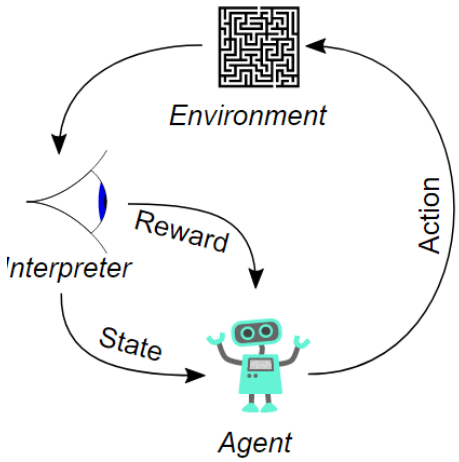


Figure 9: Reinforcement Learning Architecture

Some popular examples of reinforcement learning algorithms are:

1. **Q-Learning** is an Off-Policy algorithm for Temporal Difference learning (TD-Learning). It works by learning an action-value function, which ultimately gives the expected utility of taking a given action, in a given state, and following an optimal policy thereafter (Andrew, 1999).

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha * [r_{t+1} + \beta * MAX_a Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

Where the parameters used in the Q-value are:

- $Q(s_t, a_t)$: Value of taking an action a_t in a state s_t,
- r_{t+1}: Immediate reward,
- α: The learning rate, set between 0 and 1. Setting it to 0 means that the Q-values are never updated, hence nothing is learned. Setting a high value such as 0.9 means that learning can occur quickly.
- β: Discount factor, also set between 0 and 1. This models the fact that future rewards are worth less than immediate rewards. Mathematically, the discount factor needs to be set less than 0 for the algorithm to converge.

2. **Sarsa** algorithm is an On-Policy algorithm for TD-Learning. The major difference between it and Q-Learning, is that the maximum reward for the next state is not necessarily used for updating the Q-values. Instead, a new action, and therefore reward, is selected using the same policy that determined the original action (Andrew, 1999).

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha * [r_{t+1} + \beta * Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

Where the parameters are the sames as in the Q-Learning algorithm.

More recently, with the advance of deep learning, a new level of reinforcement learning has surged *Deep reinforcement learning*. Although it is challenges (Deep learning is often used in supervised learning), impressive applications has appeared. Mnih et al. (2013) from the University of Toronto shows how a computer can beat a human in an old Atari video game using deep reinforcement learning.

2.3.3 *Validate the predictive model*

Once the previous phases are done, we can spend a lot of time choosing, running and tuning algorithms. We want to make sure that we are using our time effectively to get closer to the goal. First of all, we need to define a test harness. The test harness is the data

that we will train and test an algorithm against. A performance measure is used to assess its performance.

The goal of the test harness is to be able to quickly and consistently test algorithms against a fair representation of the problem being solved. The outcome of testing multiple algorithms against the harness will be an estimation of how a variety of algorithms perform on the problem against a chosen performance measure. We will know which algorithms might be worth tuning on the problem and which should not be considered further (Brownlee, 2016).

Secondly, we need to define measures of success. They differ by the group of algorithm classification, Abbott (2014) state the following sucess criteria:

- **Success Criteria for Classification** - For classification problems, the most frequent metrics to assess model accuracy is Percent Correct Classification (PCC). PCC measures overall accuracy without regard to what kind of errors are made; every error has the same weight. Confusion matrices are also commonly used for classification and provide a summary of the different kinds of errors, called Type I and Type II errors, precision and recall, false alarms and false dismissals, or specificity and sensitivity; these are merely different ways of describing different ways the classifier makes errors. PCC and the confusion matrix metrics are good when an entire population must be scored and acted on. For example, if customers who visit a web site are to be served customized content on the site based on their browsing behavior, every visitor will need a model score and a treatment based on that score.

  If one will treat a subset of the population, a selected population, sorting the population by model score and acting on only a portion of those entities in the selected group can be accomplished through metrics such as Lift, Gain, ROC, and Area Under the Curve (AUC). These are popular in customer analytics where the models selects a sub-population to contact with a marketing message, or in fraud analytics when the model identifies transactions that are good candidates for further investigation.

- **Success Criteria for Estimation** - For continuous-valued estimation problems, metrics often used for assessing models are R^2, average error, Mean Squared Error (MSE), median error, average absolute error, and median absolute error. In each of these metrics, one first computes the error of an estimate—the actual value minus the predicted estimate—and the computes the appropriate statistic based on those errors. The values are then summed over all the records in the data.

Average errors can be useful in determining whether the models are biased toward positive or negative errors. Average absolute errors are useful in estimating the magnitude of the errors (whether positive or negative). Analysts most often examine not only the overall value of the success criterion, but also examine the entire range of predicted values by considering scatter plots of actual versus predicted values or actual versus residuals (errors).

2.4 POPULAR TOOLS

Regarding the tools used in the project from which the work plan is drawn up, a wide variety of tools were selected to study to understand their viability.

Some tools are WEKA, RapidMiner, KNIME, Python, Keras framework and TensorFlow framework.

- **The Waikato Environment for Knowledge Analysis (WEKA)** project aims to provide a comprehensive collection data pre-processing tools to researchers and practitioners alike. (Hall et al., 2009). WEKA contains a collection of Machine Learning algorithms and data pre-processing tools for tasks of Data Mining. The tool provides strong support for the entire experimental process, including data preparation, assessment and visualization of learning outcomes. WEKA is a Data mining tool extremely popular, especially as a academic. It has a mediocre graphical interface.

- **RapidMiner** is a powerful analytical collaborative approach to accelerate the delivery and maintenance of Data Science high value. The RapidMiner Platform is open and extendable to support all Data Science needs. RapidMiner is a software platform for data science teams that unites data preparation, machine learning, and predictive model deployment (RapidMiner). It has a pleasant graphical interface and a strong community.

  All attributes modeling and evaluation methods from WEKA are available on RapidMiner through the "Weka Extension". This extension allows to combine two of the most widely used open source data mining tools. This way, it is possible to extend RapidMiner to Weka allowing to have a complete analysis, pre-processing and visualization in RapidMiner. With this in mind, RapidMiner will be preferred over WEKA in data pre-processing.

- **The Konstanz Information Miner** (KNIME) is a modular environment, which enables easy visual assembly and interactive execution of a data pipeline. It is designed as a teaching, research and collaboration platform, which enables simple integration of new algorithms and tools as well as data manipulation or visualization methods in the form of new modules or nodes (Berthold et al., 2009). KNIME possesses a graphical user interface that allows assembly of nodes for data pre-processing, for modeling, data analysis and visualization without, or with only minimal, programming knowledge. KNIME is open source (figure 10).

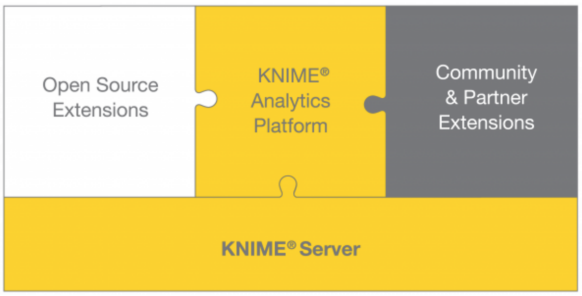


Figure 10: KNIME software Knime

KNIME is in the same category as RapidMiner, both are strong candidates to answer our challenges. Although, I personally found RapidMiner more user-friendly and better documented. For low quantity of data, RapidMiner is cheaper than Knime.

- **Python** is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales (Kuhlman, 2009).

  Python is the programming language of election for machine learning. Python is very simple, even when it comes to machine learning. Python has a huge set libraries which can be easily used for machine learning (for e.g. NumPy, SciPy, ScikitLearn, TensorFlow, Keras, etc). So, simplicity and wider applicability goes hand-in-hand to make it a so called machine learning language.

- **Keras** is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay

is key to doing good research. Keras supports complex neural network, such as Deep neural networks, supporting **Deep learning** (Keras).

- **TensorFlow** is an open source software library for high performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices. Originally developed by researchers and engineers from the Google Brain team within Google's AI organization, it comes with strong support for machine learning and deep learning and the flexible numerical computation core is used across many other scientific domains (TensorFlow).

The next step is to define which tool should be used and in which phase, we do not need to necessary use only one of the above. In fact, they can integrate with each other in order to create a good model. For example, TensorFlow is extremely usefull when it is needed to use complex neural networks, however in order to obtain good results, a normalization in data is a must. This is where RapidMiner comes in handy, RapidMiner pre-process the data available then export it to be used as input for Keras. This symbiosis has shown successful in the experiments done when studying the tools.

For this project, we use RapidMiner as the main tool to pre-process the data. As for the machine learning algorithm, we use Python 3.6 accompanied by TensorFlow, Keras, pandas, numpy, csv and matplotlib.

3

THE PROBLEM AND ITS CHALLENGES

Adding intelligence into a Home environment is not an easy task. The amount of data at our disposal is bigger than ever. However, there is a lot to do in order to extract knowledge from it. We need to define which source of data we want to use in order to achieve our goal. Firstly and mainly we need data from the user of the system, in order to find patterns in his behavior. Secondly, we can use data from the environment itself, and lastly, data outside the environment, like weather conditions. When we fuse different types of data, we get more parameters to work with, resulting in a complex and intelligent system. To achieve this, we need to record the user interaction with the system, plus install sensors in order to perceive the environment. With this said, we need some investment to apply the system in a real-world scenario.

As we have seen in the literature, extract personal data from an individual has some constraints. We must take this information into consideration when designing the system. We aim to build a system focused on the user, he must feel secure and he must not lose power over it. We aim to provide the possibility of choice, when it comes to the source of data, in this way, the user only shares the data that he is comfortable with. However, this can lead to some drawbacks. As we said before, the more data we have, the more intelligent the system becomes.

With the new law of data protection (GDPR) GDPR, record data from users from ONLY Smart Home customers has become very difficult, as we need express consent to use the data for each functionality. Implementation of sensors in customers houses was not done either, as they demand investment. In order to create a system without real world data, we were forced to simulate data. The simulation represents the data created by a diary inhabitant of the environment.

The lack of real data is clearly a challenge when we intent to apply the system in a real case study, however, we can take advantage of it. With less time consumed in preparing the data, we can focus more on define and elaborate the machine learning algorithm.

With the problem and challenges presented, we have started to select various machine learning algorithms based on the literature, representing candidates to solve our problem. In the next section 3.1 we will explore them and conclude their viability to our project context.

3.1 MACHINE LEARNING ALGORITHM ANALYSIS

In order to propose and define a solution to the problem in this project, we started to do some experiments with machine learning algorithms. To fed the algorithms we chose to use a Data Set related to this project, although, the ideal is to use real data captured by different IoT devices present in a smart environment. Unfortunately, ONLY Smart Home, currently does not possess such devices.

The following DataSet is used in supervised learning algorithms.

DataSet in this testing (*Building Test*). Attributes :

- **Room** - Numeric, number of the room (1 to 22),
- **Hour** - Numeric, hour (1,2,3,...,23),
- **Minute** - Numeric, minute (0,10,20,...,50),
- **Humidity** - Numeric, humidity recorded by the sensor,
- **Luminosity** - Numeric, Luminosity recorded by the sensor,
- **Temperature** - Numeric, temperature recorded by the sensor,
- **CheckEnvironment** - Nominal (False or True), means if environment is being checked or not,

- **Pmv** - Numeric, comfort level. Truly dependent of Humidity, Luminosity and temperature,
- **Ppd** - Numeric, "percentage of people unhappy" (0 to 100),
- **CheckPhysical** - Nominal (False or True),
- **Emotion** - Nominal (DOCILE, DEPENDENT, ANXIOUS, HOSTILE, EXUBERANT), emotion of the user,
- **checkEmotion** - Nominal (False or True), means if emotion is being used or not.

We started to define which attribute we wish to predict. We decided to predict the Emotion attribute for experiment purposes. The tested were carried in Macbook Pro mid 2014:

- **CPU** - Quad Core 2.6 GHz,
- **RAM** - 8 GB 1600 MHz DDR3.

The data was divided in two sub-groups: train data (70%) and in data test (30%). This applies to all test done in this chapter.

The work flow in RapidMiner is simple and intuitive, accompanied by a graphical interface. With this is mind, to get predictive models out of RapidMiner is easily done without programming knowledge.

- Retrieve Data Set,
- Split the Data in two sub-groups, train data and in data test,
- Select the role (define label attribute),
- Select the predictive operator,
- Apply the model,
- Apply the Performance model.

With this simple steps it is possible to create a simple predictive model in RapidMiner. Figure 11 illustrates the work flow of Decision tree predictive model.

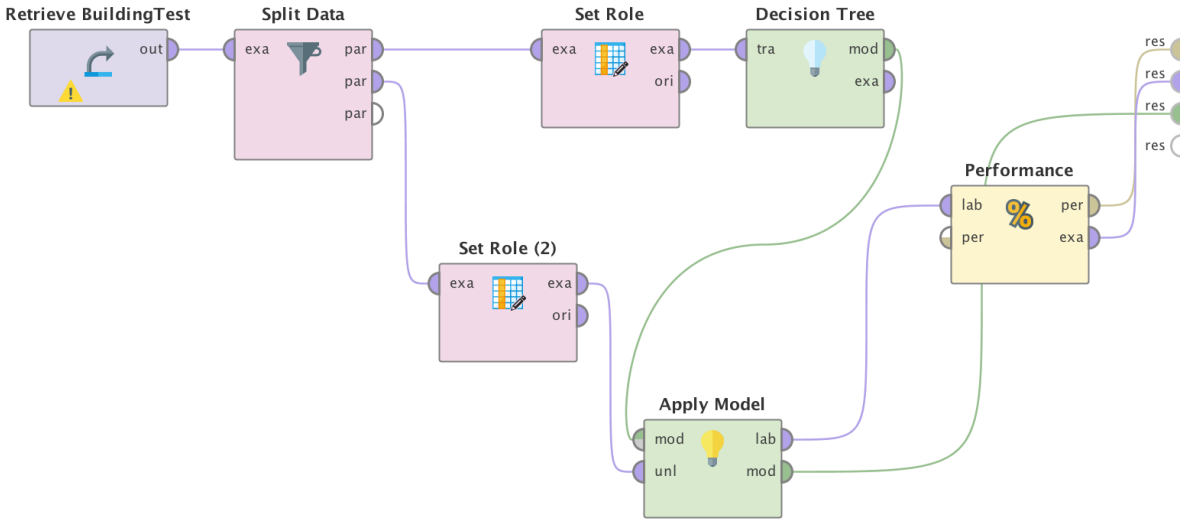


Figure 11: RapidMiner work flow: Decision tree predictive model.

As we can see in the figure above, the process is very intuitive. It is extremely modular, for instance, if we want another predictive operator, we just need to substitute one box, in this case, the decision tree by another operator available in the RapidMiner. The *res* present in the right side of the figure means what we want to be shown as output. In this example, we have the model (decision tree visualization), the performance and the test data-set.

The performance model in RapidMiner is complete, having a variety of parameters, it is possible to visualize the confusion matrix, ROC, lift and much more. It has the metrics explained in the chapter above. Example of confusion matrix obtained in RapidMiner in figure 12.

accuracy: 89.52%

	true DOCILE	true DEPENDENT	true ANXIOUS	true HOSTILE	true EXUBERANT	class precision
pred. DOCILE	38	5	0	0	0	88.37%
pred. DEPENDE...	0	395	0	0	40	90.80%
pred. ANXIOUS	1	1	1	19	3	4.00%
pred. HOSTILE	0	0	4	249	1	98.03%
pred. EXUBERA...	0	20	0	0	120	85.71%
class recall	97.44%	93.82%	20.00%	92.91%	73.17%	

Figure 12: Confusion matrix in RapidMiner: Naive Bayes example.

The following table represents the outcome of the following supervised classification algorithms in RapidMiner. The success criteria used in table 2 is the accuracy (classification problem).

Table 2: Supervised algorithms in RapidMiner

Algorithm (Default Parameters)	**Accuracy (%)**	**Total Execution Time (s)**
Decision Tree	94.76%	0.559 s
Rule Induction	91.97%	1.067 s
Naive Bayes	89.52%	0.062 s

As for artificial neural networks, the success criteria must be another, since the prediction is an estimate. In opposition to the previous algorithms, a pre-processing is needed. Neural networks only works with numeric values, so we need to transform them, to this taks, we will use RapidMiner operator "Nominal to Numerical". Alongside this operator, we will use normalization in order to test the importance of this process in NN, since the literature revealed to be an essential step in NN.

The neural network has 11 input neurons (number of attributes minus the label), 2 hidden layers, with 8 and 4 neurons respectively and the Output layer has 1 neuron. After transform the attributes to numerical, the emotion attribute vary from 0 to 4 (figure 13).

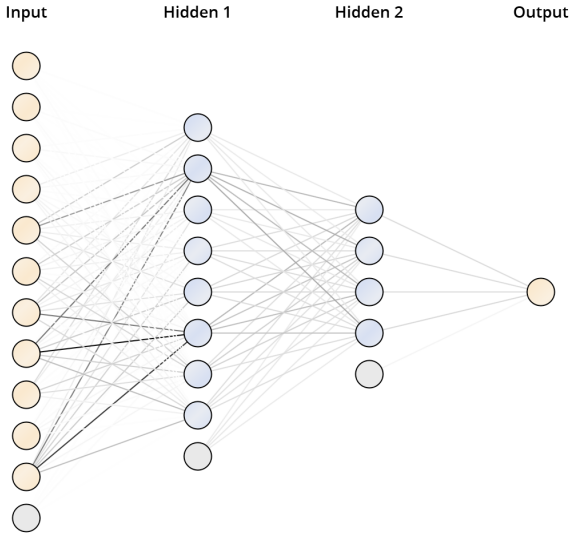


Figure 13: Neural Network representation in RapidMiner.

The following table represents the outcome of the ANN algorithm in RapidMiner. The data pre-processing is done exclusively on RapidMiner. The success criteria used in table 3 is the mean square root (estimation problem).

Table 3: Neural Networks in RapidMiner performance (500 epochs).

Normalization	**MSE**	**Total Execution Time (s)**
No	1.848 +/- 1.370	5.456 s
Yes	0.163 +/- 0.827	7.067 s

As the literature stated, the normalization has a huge impact in neural network performance. Now, let's see how neural networks behave in Keras using theano as back-end with the same parameters. The data-set is pre-processed in RapidMiner and exported as csv file extension, we then, import the data set as input to the algorithm. Neural Network model in Keras represented in figure 14.

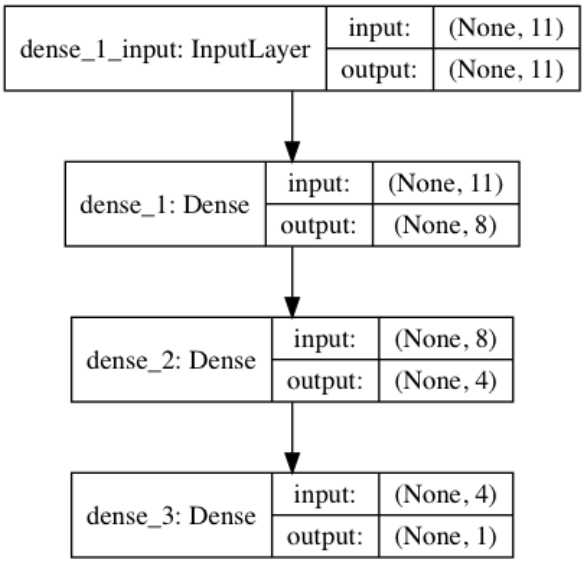


Figure 14: Simple Neural Network representation in Keras.

The following table 4 represents the outcome of the NN model in figure 14.

Table 4: Neural Network in Keras performance (500 epochs).

Normalization	**MSE**	**Total Execution Time (s)**
No	0.49	4.32 s
Yes	0.24	4.65 s

As we can see, once more, normalization has a huge importance, however in Keras it is not as advantageous as it is in RapidMiner. The model performance is similar in both tools, having RapidMiner a little advantage. In terms of execution time, Keras has shown superior.

Keras is known by it's complex neural networks, accounting that, let's model a complex NN and record it's performance. For this test the neural network has 3 hidden layers, with 64, 32 and 16 neurons respectively (illustrated in figure 15 . To complex even more, we train the NN with 3000 epochs.

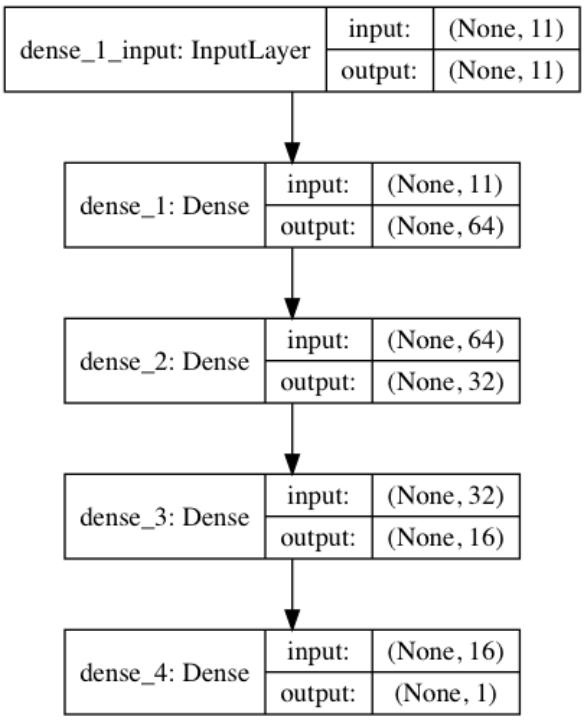


Figure 15: Complex Neural Network representation in Keras.

The following table 5 represents the outcome of the NN model in figure 15.

Table 5: Complex Neural Network in Keras performance (3000 epochs).

Normalization	**MSE**	**Total Execution Time (s)**
Yes	0.08	40.51 s

The model performance is extremely good, however it took 40.51 seconds to finish the task.

In this chapter, we have study the different algorithms in two different tools (RapidMiner and Keras). For simple supervisioned algorithms, RapidMiner has shown more than capable of resulting good predictive models, the accuracy and it's execution time is very satisfactory. RapidMiner has proven important in pre-processing data.

For artificial neural networks, Keras has shown more usefull, since it's capable of produce complex NN's, however a programming knowledge is needed to do so, imposing this ways some barriers to some Data Scientists. Despite this, the continuous execution of this algorithm can represent a future problem, since the time are high and the computational power can be limited in a real word application Smart Environment. However, in order to satisfy our objectives, there is no necessity to run the algorithm in a continuous way. An analysis will be made to determine when it's needed to execute without harming the proposed system. With this in mind, we intend to test this same algorithms and tools (and more) in a raspberry pi 3 model B, alongside with real data provided by IoT devices.

With Keras we can quickly build and test a neural network with minimal lines of code. We can build simple or very complex neural networks within a few minutes. For future work, we might use TensorFlow, as the complexity of the problem may increase.

As we said before, we aim to focus on the inhabitant of the environment, therefore, it is very important to give some decision power to the user side. We have seen supervised algorithms in the previous section. These algorithms can be applied to the smart home context, where they can predict the best action to take in a determined instance. However, Supervised and Unsupervised learning does not learn directly by the user action, abstracting our focus in the end user. With the different algorithms over the table, we have decided with the help of ONLY Smart Home a reinforcement learning approach. Reinforcement learning in this scenario is ideal, the algorithm can converge and adapt (with the help of the user) to the user lifestyle. Thus, this type of algorithm benefits from bidirectional interaction between the system and the individual.

Nevertheless, artificial neural networks can be used in a reinforcement learning algorithm, as they can predict which is the best decision that the system can take based in the user appreciation of the specific action. Deep neural networks have been proven to be the most effective Supervised learning algorithm, as we conclude in the section before.

3.2 PROPOSED APPROACH - SOLUTION

Smart Homes are environments that automate action and adapt themselves to user behaviours. In this sense, it is necessary to employ learning strategies to allow Smart Homes to truly become intelligent, in a sense that they anticipate needs and actions. This requires constant monitoring of environments, users and their actions, as well as, non-supervised dynamic learning strategies. With this said, we want a solution who fits within our needs.

We want to create a system around the user of the system, being in this case study, the inhabitant of the Smart Home, the user is the primary element in our system. With this in mind, we need to establish ways to connect the user to the logical unit behind the system, we want a bi-directional communication, as feedback is as much important as system suggestions. Figure 16 represents the proposal solution agreed.

This architecture was built thinking for the future. It is prepared to work with real data, from various sources.

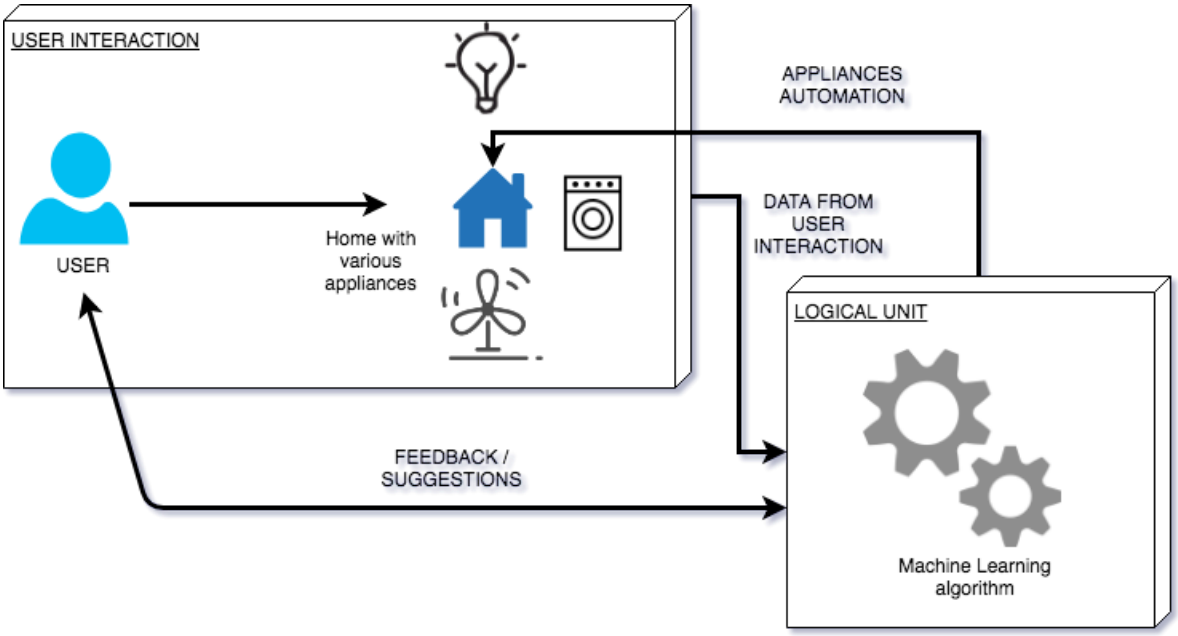


Figure 16: Initial architecture

A user interacts with his environment, which has various appliances. We intend to automate this different appliances present in the Home, as we see in the figure 16. The logical unit is responsible for determinate which action can be performed at each instance, it con-

tains a machine learning algorithm working with the data fed from user interaction. The communication between the two entity's is present in the diagram proposed.

After the analysis of different algorithm plus problem and challenges, we present our system allied with a machine learning algorithm as a solution to our objectives raised. This solution consists in a multi-agent system, who is capable to respond to our needs, both in communication and logic.

3.3 SUMMARY

With the proposed solution presented, it is possible to answer the problems raised in this project. It is capable of converging to different usage patterns and discern between different contexts in the form of user preferences. The possibility of communication between the user and the system is the principal distinct factor between this work and the previous one studied in the related work.

The following chapter 4 will be dedicated to discuss the implementation. All the steps will be detailed explained.

4

DESIGN AND IMPLEMENTATION

In this chapter, our system will be presented. An elaborate explanation and analysis will be performed, arguably showing the system capability to resolve our established problems.

In order to implement our proposed solution, we need to establish a stable environment able too meet our needs. As our implementation uses python with various frameworks and tools related to machine learning, we have created an environment with all the required libraries and dependencies. In this way, we are able to compile and test the code easily and faster, independent of the working machine. We have accomplished this with the help of PyCharm, a Python IDE.

4.1 PROTOTYPE SYSTEM

In order to build an intelligent system, to be applied in a Smart Home context, we must use data from the user interaction with the system. As said before, unfortunately, ONLY Smart Home did not provide the required conditions to this end. However, we establish a set of data related to the user interaction, achieving it by simulating a set of actions taken in an environment by a user, day by day. With this data, we have built a prototype system, representing our current system.

For this prototype, we have only simulated actions involving light and blinds. In this simulation, we have used a Gaussian Distribution to determine which hour of the day, the user interacts with the lights or blinds. The possible actions for both are ON / OFF and OPEN / CLOSE, respectively. We have created various scripts, in order to simulate different types of user. In addition, every profile defined and simulated has a certain deviation in

order to introduce some unpredictability on the data generated. This unpredictability is essential, as we want to prove the capacity of our system adaptation, as well as different profiles.

The data generated through our scripts, do not need to be pre-processed, as they are generated specifically to this project. However, the other sources of data, external to the user interaction with the environment, continues to be subject to pre-processing procedure.

With our objective being to automate these two appliances present in our simulated environment at a certain time of the day, we needed to create a time interval. The period of the day must be somewhat specific, as the actions are punctual. For prototyping, we configured our system to divide a day into 96 intervals, allowing a significant possibility of choice for our algorithm. Each interval represents 15 minutes, limiting the chosen time to perform an action to:

$$x : 00; x : 15; x : 30; x : 45; \in 0 \leq x \leq 24$$

Our developed application will be able to accept different intervals, but for testing purposes, these intervals above will be used.

With the data simulated by our scripts, we have the needs to understand the user patterns related to the possible actions. The system is capable of determine the best interval, of all the 96 intervals, to do an action related to the lights and blinds. In the future, more actions and appliances will be introduced, as well as more sources of data. The system reacts to the user feedback, the decision making of the algorithm is dramatically changed, provided via the interface agents. This prototypes is represented in figure 17.

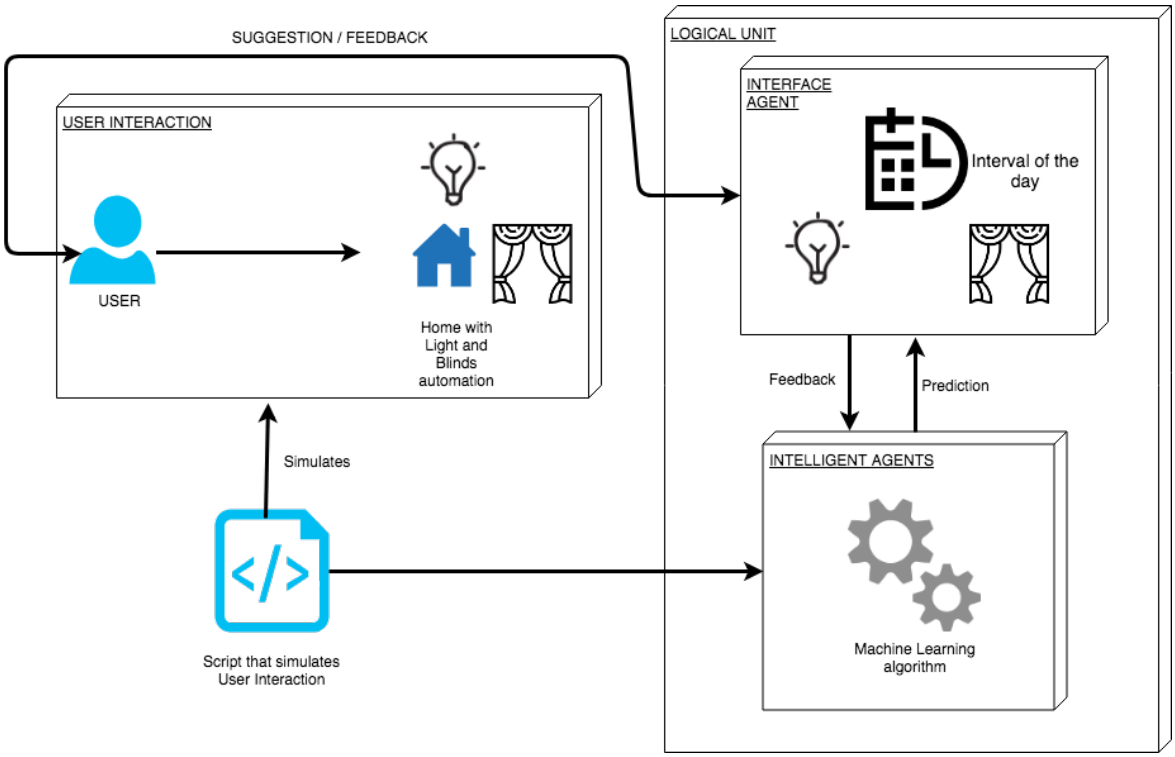


Figure 17: System prototype architecture

In order to answer the problem defined, which focuses in the importance of the user in an automation environment, it is needed to establish a solid connection between the automation system and the user. With this in mind, we developed a multi-agent system, with distinct agents. This multi-agent system is defined as Logical Unit in this project.

The **Logical Unit** is responsible for the decision making in our system. It contains two types of agents:

- **Interface agents** - Interface agents are responsible for communicating with the user and the system. The system is capable of suggesting the user of the action chosen by our intelligent agents, and the user can express his opinion about the decision of the agents. This results in our system converging to the user personal preference actions. Interface agents are vital to accomplishing this, as they serve as intermediates between the system and the user.

- **Intelligent agents** - Intelligent agents can view and perceive the environment through sensors and act upon that environment through effectors. The environment is composed by different rooms, these rooms have multiple sensors regarding the appliance.

The logical unit has multiple intelligent agents, one for each appliance in the environment, lights, shutters, temperature and others. These agents perceive their respective sensor in the different rooms and after the deep reinforcement learning process, they actuate through actuators in the ambient. For instance, the shutter agent, is responsible for the automation of the shutters in the different rooms.

This prototype is used in the following chapters, it is used to prove and test our approach. The system prototype in figure 17 represents our system architecture adapted to our use cases defined for testing and validation purposes. However, a more generic architecture, capable of using other sources of data was presented before (figure 16).

In the following sections, we will talk about our prototype system, how it works, how it fits in our problem and how it behaves. This prototype system was presented in the The 14th International Conference on Intelligent Environments - IE '18 Rome 25th-28th June 2018, as a paper, by the name of Reinforcement Learning Approach for Smart Homes Mirra et al. (2018).

4.2 INTERFACE AGENT

Interface agents are [C]omputer programs that employ artificial intelligence techniques in order to provide assistance to a user dealing with a particular application. ... The metaphor is that of a personal assistant who is collaborating with the user in the same work environment.' Maes and Kozierok (1993). They represent the bridge between the system and the user. They manage to gather the user feedback and communicate with the system, and contrariwise. Our system has a special attention to the user, as they provide a crucial role on the intelligent agent decision.

An interface agent works alongside the user, as they are bonded by the same goal. This relationship happens in our system, where we implemented one interface agent capable of receiving feedback from the user, serving as an intermediary. To the user satisfaction, his iteration with the system is simple and not complex. The end user merely needs to express his feeling about an action by a simple yes or no. He does not need to comprehend how the system interprets this information, this responsibility is on the interface agent, where it certificates that the user feedback reaches the system. On the other hand, this agent also communicates information originated in the system, providing the user with information regarding the system decision.

In our application, a single interface agent is implemented, being capable of a bi-directional communication between the user and the various intelligent agents. Represented in figure 18

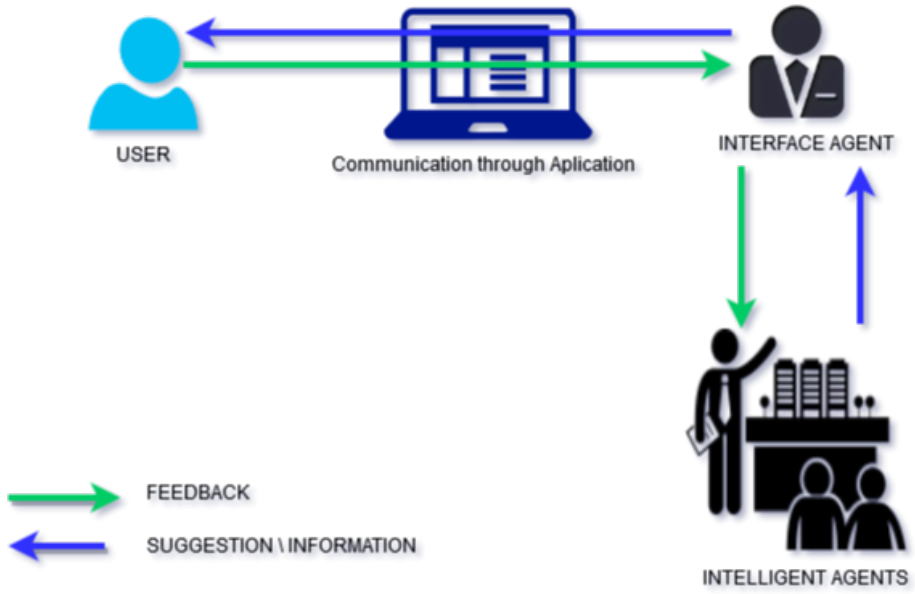


Figure 18: System Communication

The interface agent communicates with the different intelligent agents in order to understand their predictions, in other words, the intelligent agent has access to the reward data structure of each intelligent agent. With this knowledge, the interface agent presents to the user the best interval to automate each action he pretended.

Gather feedback from the system user through interaction on the application is simple, represented in figure 19. This user interaction with the system occurs at the end of every algorithm execution, in other words, in the of the day. With this frequency, the system can use this feedback to change its decision in the next day, although this constant system necessity of asking for feedback can be invasive and consuming to the end user. The system demands feedback for each room present in the user profile.

```
The agent thinks time Interval 34 || 08:30 is the best to open Shutters for day Monday in room Room1
The agent thinks time Interval 79 || 19:45 is the best to Close Shutters Monday in room Room1
The agent thinks time Interval 79 || 19:45 is the best to Turn on the lights Monday in room Room1
The agent thinks time Interval 90 || 22:30 is the best to Turn off the lights Monday in room Room1
Please enter an interval of the day (0-96):  34
You entered 08:30
The best action for Interval 34 is Open Shutter in the day Monday
Do You Agree? (YES/NO) :  NO
You entered NO
```

Figure 19: System user feedback

The figure above represents an example of our feedback system. The interface agent after informing the user the intelligent agent predictions and suggestions, asks for feedback in a specific interval. With the introduction of an interval, the interface agent requests a response for that interval action, being the possible answers: ”YES” or ”NO”. These two possibles answers can critically change the course of the algorithm for the next iterations, as a positive value or negative is added to that specific action, in a certain time and day of the week.

In the future work, we plan to remodel the feedback system, by introducing intelligence in the interface agent. This way, the interface agent would decide the best moment to ask for feedback, reducing the need for an interaction every day. It is important to think in the user comfort when developing a system, however, is feedback is essential. An equilibrium must be achieved to satisfy the end user.

4.3 LEARNING AGENTS

Our system architecture is composed of multiple agents. These autonomous entities communicate with each other, causing our system to be a multi-agent system. We already presented one type of agent: the interface agent, however, there is only one in our implementation. Differently, other types of agents are not alone, they are called learning agents.

Learning agents are responsible for perceiving the past interactions of the user and, based on that information, choose the best time of the day to perform an action. Each them, one for each possible action, convey his decision according to what he thinks its best. Figure 20.

To accomplish that, we establish them a learning procedure. With every learning agent, they have a machine learning algorithm that instructs them how to behave, albeit this formula

can be changed. Let's think of an intelligent agent as a human. The brain represents the algorithm and the body the means to actuate. In our project context, the brain can be changed, and this process represents the introduction of a new algorithm in the agent.

These agents are critical in our system, as they define the decision making. They work independently, but they do communicate with the interface agent. When developing these agents, we elaborated two distinctive types of machine learning algorithms, a **supervised deep neural network** and a **reinforcement learning algorithm**. In the following section, we test these two different type of "brains". We compare them, concluding their viability.

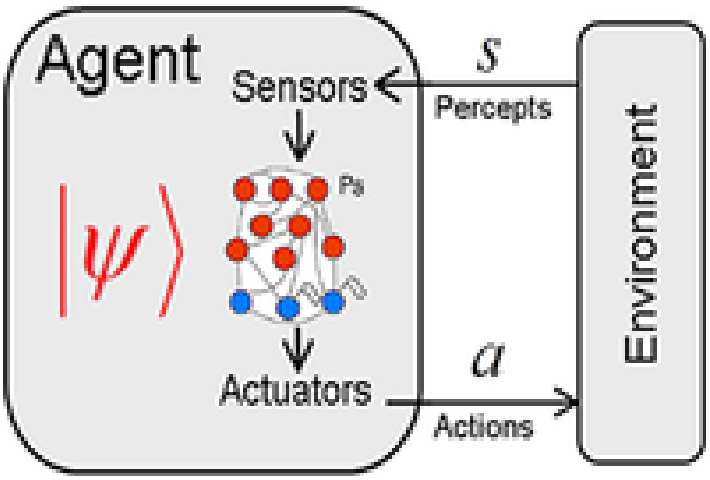


Figure 20: Learning Agent

4.3.1 *Deep Neural Network*

With the application prepared to run the machine learning algorithm, we started to develop a reinforcement learning approach. The intelligent agent, as the responsible for decision making process, needs to learn how to behave with the information gathered and provided by the interface agent, by its machine learning algorithm.

In this phase, we developed a supervised algorithm applied in our intelligent agent, with the objective to understand how the agent behave with knowledge. To this end, we selected one algorithm previously studied in machine learning algorithm analysis, 3.1. We have selected the deep neural network (NN) developed in Keras, using TensorFlow as backend.

This NN was adjusted to this concrete problem, where the objective is to automate two appliances (lights and shutter). It is constituted by 4 input neurons, representing the number of variables presented in the historical data script, 3 hidden layers, with 64, 32 and 16 neurons respectively and the Output layer has 1 neuron (figure 21). Each intelligent agent predict one action, as the NN only has one output layers, in this case, in order to automate 2 appliances, 2 intelligent agents are needed, both with the same input neurons, but with different outputs. These agents communicate with the interface agent, as soon as they finish the decision making process, their predictions.

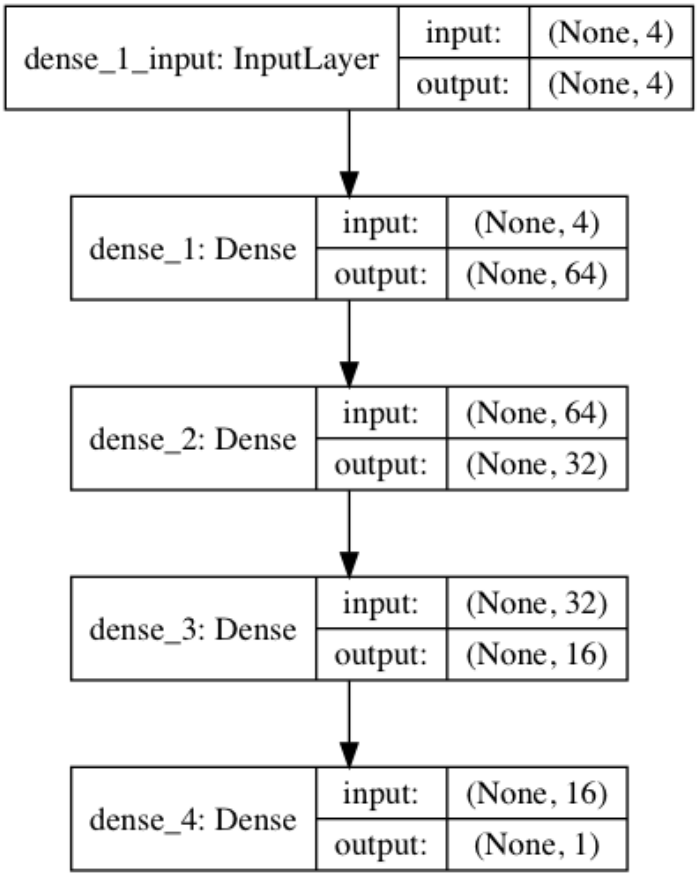
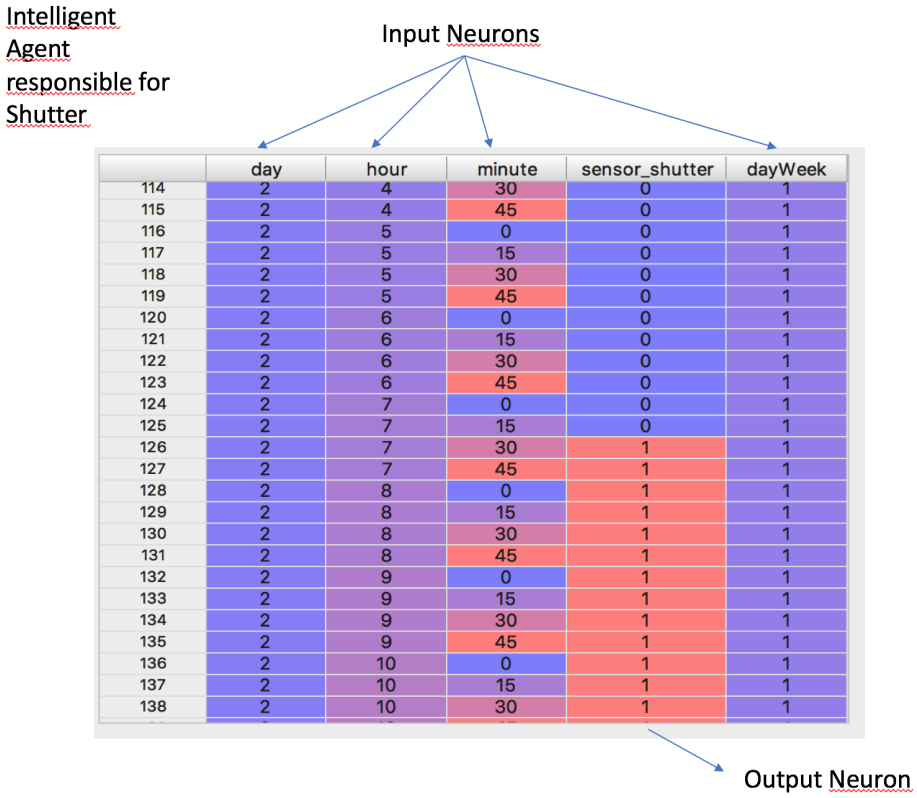


Figure 21: Neural Network Layers

The neural network is fed by the historical data of a specific profile, this data is simulated through scripts. The data need to be normalized and numeric to be understood by neural networks. In our implementation, the data is imported and saved in a *DataFrame* structure. *DataFrame* is a two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). Figure 22.



	day	hour	minute	sensor_shutter	dayWeek
114	2	4	30	0	1
115	2	4	45	0	1
116	2	5	0	0	1
117	2	5	15	0	1
118	2	5	30	0	1
119	2	5	45	0	1
120	2	6	0	0	1
121	2	6	15	0	1
122	2	6	30	0	1
123	2	6	45	0	1
124	2	7	0	0	1
125	2	7	15	0	1
126	2	7	30	1	1
127	2	7	45	1	1
128	2	8	0	1	1
129	2	8	15	1	1
130	2	8	30	1	1
131	2	8	45	1	1
132	2	9	0	1	1
133	2	9	15	1	1
134	2	9	30	1	1
135	2	9	45	1	1
136	2	10	0	1	1
137	2	10	15	1	1
138	2	10	30	1	1

Figure 22: Data Frame of Shutter Agent

With this data structure, the agent can learn and then predict which is the value of the sensor at every interval of time, when it is zero, it means that the shutter is closed, when it is one, it means that the shutter is open. The agent relates the past shutter states, to every different day of the week.

In order to test this intelligent agent, we split the data present in the DataFrame in two distinct DataFrames, one using for testing the agent, and the another to test his performance and accuracy. We have divided them in 60%/40%, respectively. For shutter example, we have obtained a high accuracy, rounding the 100%, meaning that the agent predicted correctly the different states of the shutters at every interval (Figure 23).

However, this result is for our simulated user activity, with a determined randomness attributed. The simulated data was achieved with a normal distribution. For each day, a different interval is picked according to our normal distribution. This pseudo-randomness is easily dismantled by our neural network, understanding the logic behind the data variation.

```
Train Score: 0.03 MSE (0.17 RMSE)
Test Score: 0.04 MSE (0.21 RMSE)
['loss', 'acc']
acc: 96.28%
```

Figure 23: Neural Network accuracy

With a model trained with 30 days and tested for 15 days, the agent predicted with an accuracy of 96,28%. This value was calculated based in 5 repetitions, being the mean value. The data variation is not high, explaining the high accuracy. Furthermore, this decision is very dependent on historical data, used to train the model, causing overfitting.

We aim that the algorithm will also perform well on predicting the output when fed "validation data" that was not encountered during its training. However, with this algorithm, not so prepared to learn with the user activities and change of behaviour, it will perform badly.

If we create data without any logical behind, only at random, the neural network would not find any pattern and will fail when predicting. In a real-world scenario, both of the previous situations dont fit. With real data, there is some logic behind it, some pattern, however, this pattern is not created by a machine, is a little hard to find a perfect receipt.

As the agents predict with extremely high accuracy the state of an appliance for every interval, it doesn't take account user feedback to converge to his habits. This approach has proven to be extremely efficient when using only historical data without user feedback. However to our goals, an adapting system is crucial, and this implementation lacks this feature, it is even nonexistent. The communication is unidirectional, opposed to desired bidirectionally, represented in figure 24.

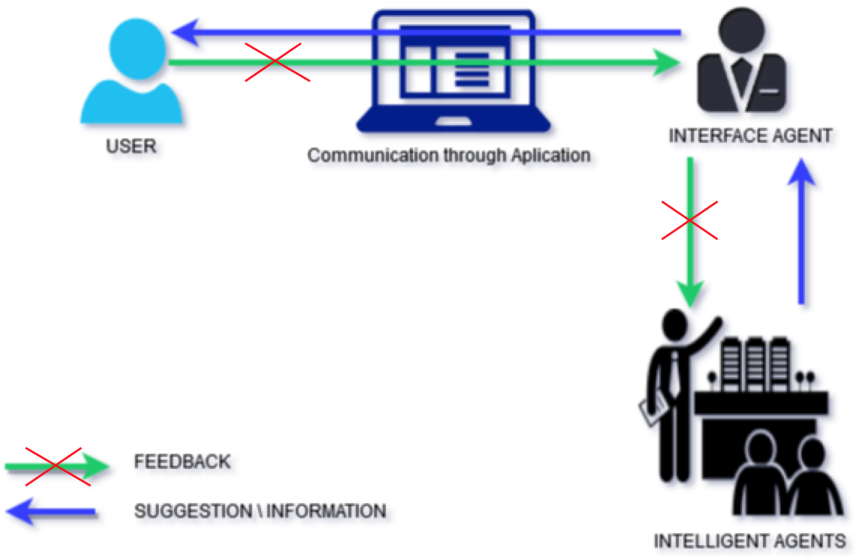


Figure 24: Unidirectional communication opposed to desired bidirectionally

This lack of communication goes against our objectives and purposes. The algorithm incapacity of gathering user feedback to adjust his future decision making makes this algorithm non-viable to our problem. In a supervised learning, what is correct must be defined. In our case study, the perceptual learning must be accompanied by the user feedback, who does not tell the algorithm what is correct or incorrect in an explicit way.

4.3.2 *Reinforcement Learning*

The system must be prepared for communication, an adaptive, tangible algorithm is necessary. Reinforcement learning is well known as an adaptive algorithm. Its actions and decisions carry results in its environment and the human world. With the help of the occupant of the environment, the system learns its best suitable approach method with the view to delight the user.

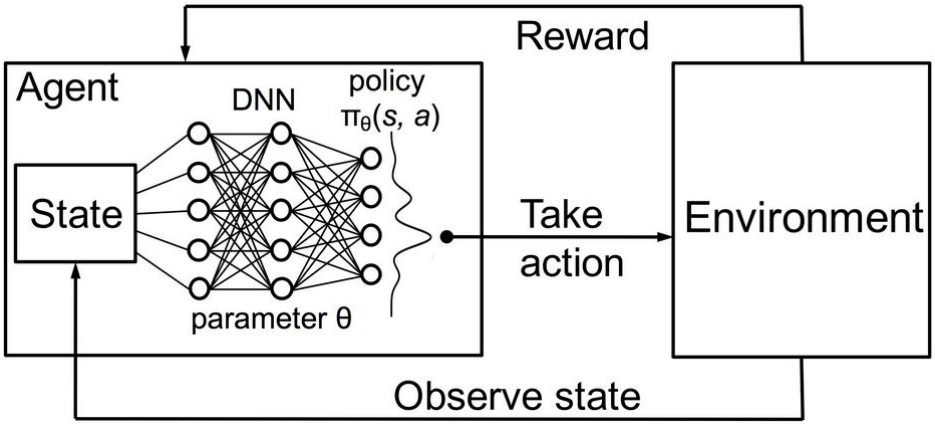


Figure 25: Policy based reinforcement learning

The purpose of this solution is to develop a system capable of taking the best action possible based on its environment and user preferences. An intelligent agent perceives the ambient and the past interactions of the user with the home in order to learn what is the best action to perform, which action has a certain reward associated in order to inform the agent his behavior.

Reinforcement learning is learning what to do, how to map situations to actions, so as to maximize a numerical reward signal Qi-ming et al. (2009). The primary aim is to cast learning as a problem involving agents that interact with an environment, sense their state and the state of the environment, and choose actions based on agent, environment and user interactions. In reinforcement learning the agent comes pre-equipped with goals that it seeks to satisfy. These goals are embodied in the influence of a 'numerical reward signal' on the way that the agent chooses actions, categorizes its sensations and changes its internal model of the environment. Despite the obvious connection of these terms to behavioral psychology, some of the more impressive applications of reinforcement learning have been in computer science and engineering applications.

Reinforcement learning is an iterative process, the more rounds of feedback, the better the agent's strategy becomes.

The reinforcement learning algorithm present in the logical unit, constituted by multiple intelligent agents and by a machine learning algorithm, is responsible for making the intelligent agents intelligent. It allows them to learn about the environment and acting on them, these actions are rewarded in order to converge to an optimal automation. Our goal is to know when to do a respective action, for example, in which time of the day we should

open or close the shutters. First, we need to define the global variables and initialize them, define the number of intervals in the day, initialize the weight and define the loss function, initialize total number of episodes to train agents on, initialize total reward for intervals for each agent and define the chance of taking a random interval e. Then we proceed to define the multiple reward functions, one for each intelligent agent procedure. Lastly, agents will train by taking actions in our environment through the following procedure in algorithm.

1.

Algorithm 1 Deep Reinforcement Learning

```
1: procedure TENSERFLOW SESSION TO OPEN SHUTTERS
2:     while Number of Rooms do
3:         tf.Sessions() as sess:
4:         dataSet ← User activity data
5:         Initialize loop counter i=0
6:         while Number of Days do
7:             Initialize loop counter j = 0
8:             while Number of Episodes do
9:                 if random number < e then
10:                    interval ← Random interval
11:                else
12:                    interval ← chosen interval
13:                reward Open ← Reward Function for interval
14:                Update the network and the Reward
15: procedure TENSERFLOW SESSION TO X ACTION
16:     Same procedure as open shutters Session, Reward Function differs.
```

For each room present in the profile this algorithm is applied. These procedures are replicated to the different intelligent agents present in the system, with an exception for reward functions. Each intelligent agent has is own reward function as is own reward variable. This is absolutely needed since each action is independent of other actions present in the system, the user has the power to decide each action wants to be automatized, thus the need of independence on these intelligent agents. The number of iterations is for each decision making agent is 5000, although this block of algorithm presented can be repeated for a number of days, only for testing purposes.

This approach will not consider states, this means that rewards for an action is not conditional on the state of the environment. All we need to focus on is learning which rewards

we get for each of the possible actions, and ensuring we choose the optimal ones, this is called learning a policy and we will discuss our policy.

For now, the user activity data represents the past user activities. Nonetheless, this data can be real time data gathered by the user activity and feedback, being our algorithm prepared for this effect.

Policy Gradient Network

The simplest way to think of a Policy gradient network is one which produces explicit outputs. In the case of our case study, we don't need to condition these outputs on any state. As such, the network will consist of just a set of weights, with each corresponding to each of the possible time of the day to do an action, being turn the lights on, shutter, and so on, and will represent how good our agent thinks it is to do the respective action in that interval. Weights are all initialized at the same value, being all intervals equal to the agent at the initial state.

All of our intelligent agents present in the system have its neural network with the same policy, as we have multiple agents, one for each action possible in our system. The introduction of new agents are possible as soon as we need new actions, associated with new appliances. However, they need to be configured, as each action for a determined appliance, has a specific reward function.

Our neural network learns a policy for picking actions by adjusting it's weights through gradient descent using feedback from the environment. There is another approach to reinforcement learning where agents learn value functions. In those approaches, instead of learning the optimal action in a given state, the agent learns to predict how good a given state or action will be for the agent to be in. Both approaches allow agents to learn good behavior, but the policy gradient approach is a little more direct.

The figure 26 below established our simple neural network. It consists of a set of values for each of the intervals. Each value is an estimate of the value of the return from choosing the interval. We use a policy gradient method to update the agent by moving the value for the selected action toward the received reward. We feed the reward and chosen interval into the network.

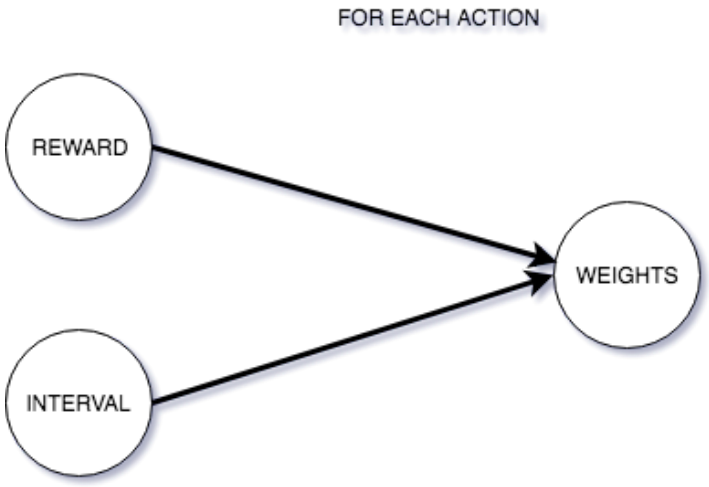


Figure 26: Policy Network

The policy is responsible for predicting the best interval possible at determinate time. This way, we improve the performance of the algorithm, eliminating the necessity of exploring all the possibilities. However, when we validate the decision of the neural network policy, by giving it a reward, the network takes this rewards on consideration and in the next algorithm iteration, the prediction for the same action can alter.

Each interval has a weight for a determined action, this weights are initialized at 1. In the first iteration of the algorithm, there is no knowledge in order to do an intelligent decision, since the expected reward for an action is the same for all the intervals. After each iteration, the network is updated with the respective reward for that interval. In this way, we choose the interval with the highest weight, corresponding to the policy prediction. However, in order to find the best interval for a specific action, we must explore all possible intervals, to find out which has the best reward associated. This leave us an interesting topic, called exploration versus exploitation.

Exploration vs Exploitation

When developing the network, we aim to define an optimal policy. To that end, we need to understand two concepts very common in the reinforcement learning. Illustrated in figure 27

- **Exploration** is a very common practice in Reinforcement Learning. The idea is that the agent needs to explore in order to get a good understanding of the reward dynamics

of the environment it is in. By sometimes acting randomly, it can end up in states it may know less about, and can learn from them to obtain a more optimal policy.

- **Exploitation** is where we use what we know based in the current information to pick the best action possible.

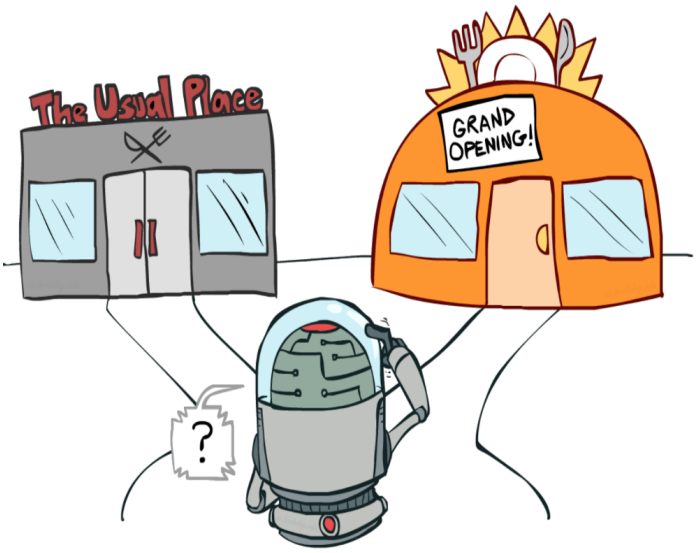
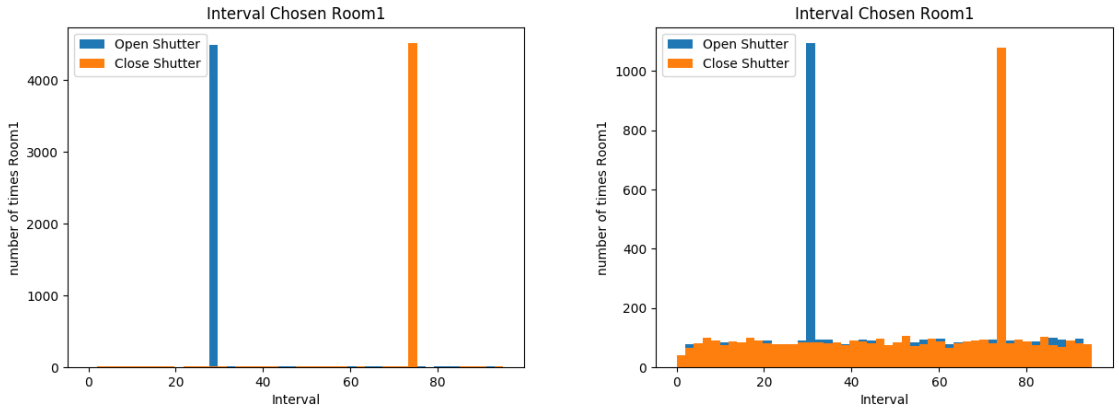


Figure 27: Exploitation VS Exploration

To achieve the optimal policy, we need to balance between exploration and exploitation. In this project context, to update our network, we will implement an e-greedy policy. Figure 28 represents the agent prediction with and without e-greedy policy. We can clearly see that our agent is way more inclined to one interval than the others, mainly when the epsilon value is zero. However, for epsilon = 0.5, the agent in 5,000 iterations, has explored all the 96 intervals, obtaining a vast knowledge of all possibilities. In this example (figure 28), the decision of the agent does not differ from different epsilon values, although for different user profiles and habits, this simple technique can alter the decision of the agent, as he gets full knowledge.



(a) Prediction Interval for epsilon = 0 (b) Prediction Interval for epsilon = 0.5

Figure 28: E-greedy policy comparison

This represents that our agent will choose the action that corresponds to the largest expected value (exploitation), but occasionally, with e probability, it will choose one interval randomly (exploration). This randomness is crucial, it allows the agent to try out each of the different intervals of the day to continue to learn more about them, without it, the agent would ignore potential better intervals than the one chosen.

Once our agent has taken an action, it then receives a reward based in the reward function. With this reward, we can then make an update to our network using the policy loss equation.

$$loss = -log(\pi) * A$$

In the equation above, A is advantage, and is an essential aspect of all reinforcement learning algorithms. It corresponds to how much better the agent action was than the baseline, being 0. Intuitively, a positive reward indicates a increase weight to that specific interval, and a negative reward, a decrease weight.

The parameter π is our policy. In this case, it corresponds to the chosen interval weight.

This loss function allows us to converge to the best interval to do the respective action. The agent will be more or less likely to pick that action in the future. By taking actions, getting rewards, and updating our network in this circular manner, we will quickly converge to an agent that can solve our problem.

In order to predict the best interval of the day to execute possible actions. We developed multiple agents, each with their respective weights and rewards. These agents run simultaneous, being possible by our chosen machine learning framework, Tensorflow.

The agent will be trained by the actions taken in the environment, and receiving positive or negative rewards. Using rewards and actions, it allows us to know how to properly update our network in order to more often choose actions that will yield the highest rewards over time.

Reward Function

Reward functions are essential in the reinforcement learning paradigm, they describe how the agent ought to behave. In other words, they have normative content, stipulating what we want the agent to accomplish. So to an extent, reward functions determine what the agent's motivations are, so, we have to make it well. There are no absolute restrictions, but if our reward function is "better behaved", the agent will learn better. Practically, this means a speed of convergence, and not getting stuck in local minima. But further specifications will depend strongly on the species of reinforcement learning we are using.

The problem is considered solved if the agent learns to always choose the interval of time that most often provides a positive reward. In such a case, we can design an agent that completely ignores the state of the environment, since for all intents and purposes, there is only ever a single, unchanging state (represented in figure 29).

Figure 29: Only action effect reward

In our work, we have multiple distinct actions that we want to accomplish, therefore, we created different agents. As reward functions are crucial in the learning process of the agents, a universal reward function is not applied to our work. Each action possible in our system, is different, as we have multiple appliances, for this reason, a dedicated reward

function is developed for each action. For example, in our prototype, the reward function associated with the action of turning on the light's, can not be used to reward the action of turning off the blinds, as they have different parameters and sources of data when calculating the reward. We believe this way we diminish the complexity, as the algorithm gets simpler. Also, if we want to add another appliance to the environment, we must only add a new agent to that purpose. There is no need to change the algorithm.

Reward functions are dependent of the user pattern, present in user activities data sets, in account which action the user took in that instant, if it correspond to the agent decision, a positive reward is granted. The reward function associated with each action can drastically change after receiving the user feedback, indicating the possibility of adapting to the user preference.

The reward function present in our project has suffered several modifications, as we tried different approaches, in order to refine the capacity of decision making of our agents. The evolving process was based on testing and analyze how the agent converged to an optimal solution, the number of variables that interferes with the reward process has changed along the time too. The definition of a final reward function was very demanding in terms of time, as we pursue-ted optimal ones.

Before starting developing our reward functions, we need to understand which variables and conditions would interfere with the attribution of a reward. For instance, it is normal to have different behaviours on different days of the week, especially on weekends. Consequently, we have determined to introduce the day of the week in the reward function. In summary, these are the conditions that affect our reward functions.

- **User activities** - User activities data are used when attributing a reward.
  - **Day of the week** - The different days of a week, Monday, Tuesday, etc, have an impact in the reward attribution.
  - **Different weight for most distant days** - The distant the days, the less weight they have when attributing a reward. This was applied, as the user can change its habits.
- **Different Rooms** - It makes sense to include the room in the process. Each room is treated differently in the reward functions.

- **Environment Sensing** - Data from various sensors, like luminosity, can be use when attributing a reward.

- **External data from the environment** - Data from APIs, like the present weather, can be used when attributing a reward.

- **User preferences** - The user can control which data can be used in the decision making process.

- **User feedback** - The user explicitly gives his feedback on a determine action, consequently, altering the reward associated.

With these conditions listed, we must define a structure to store the reward, for each action. We defined a data type that is capable of satisfying this conditions. It consists of an interval by day of the week multidimensional matrix. The number of dimensions is equal to the number of rooms in the environment. The table 6 below represent the proposed reward data structure.

Dimension 0								
		Day of the week						
		Monday (0)	***Tuesday (1)***	***Wednesday (2)***	***Thursday (3)***	***Friday (4)***	***Saturday (5)***	***Sunday (6)***
Intervals	***0***	r (0, 0, 0)	r (0, 0, 1)	r (0, 0, 2)	r (0, 0, 3)	r (0, 0, 4)	r (0, 0, 5)	r (0, 0, 6)
	1	r (0, 1, 0)	r (0, 1, 1)	r (0, 1, 2)	r (0, 1, 3)	r (0, 1, 4)	r (0, 1, 5)	r (0, 1, 6)
	2	r (0, 2, 0)	r (0, 2, 1)	r (0, 2, 2)	r (0, 2, 3)	r (0, 2, 4)	r (0, 2, 5)	r (0, 2, 6)
	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.
	95	r (0, 95, 0)	r (0, 95, 1)	r (0, 95, 2)	r (0, 95, 3)	r (0, 95, 4)	r (0, 95, 5)	r (0, 95, 6)

Table 6: Reward Data Structure

e.g. Reward for room 0, for interval 23 at Monday.

$$reward = r(0, 23, 0)$$

The reward function output represents an explicit number, known as the score. This final score is composed by the addition of two sub-reward functions. One related to the user past activities, and one, with information based on API and sensors. These two sub-reward functions also produce a numeric value, the addition of these two values produces the final score of a determined action. Represented in the equation below.

$$Reward\_Total = User\_activity\_reward + API + Ennvironment\_Sensing$$

This division originated to facilitate the implementation of the final reward function. To understand the user pattern, we must analyze the available user activity data. This process has demonstrated to be complex, justifying the necessity of having a dedicated reward function to this end. For the user activity function, a file with the previous user activities is analyzed by the algorithm for that specific action, however, only a portion of the data is analyzed. If he wants to automate an action in the kitchen, at Tuesday, it solely considers the previous actions in the kitchen, in the past Tuesdays, with the most recent Tuesdays receiving a vaster weight in the score.

When calculating the user activity score, the agent carefully compares his predicted interval for an action by the tangible interval it took place in the past days. In the first version of the function, we compensate it with a positive reward when the agent performs the precise action in the right period, otherwise, in other all actions possible, we reward it with a negative reward. There are only two possible rewards, in this specific case, it is typically called a binary reward function, where we specify a particular action to be rewarded in a specific time of the day: to facilitate the implementation of the final reward function.

$$R(a1, interval) = 1$$

$$R(a2..n, interval) = -1$$

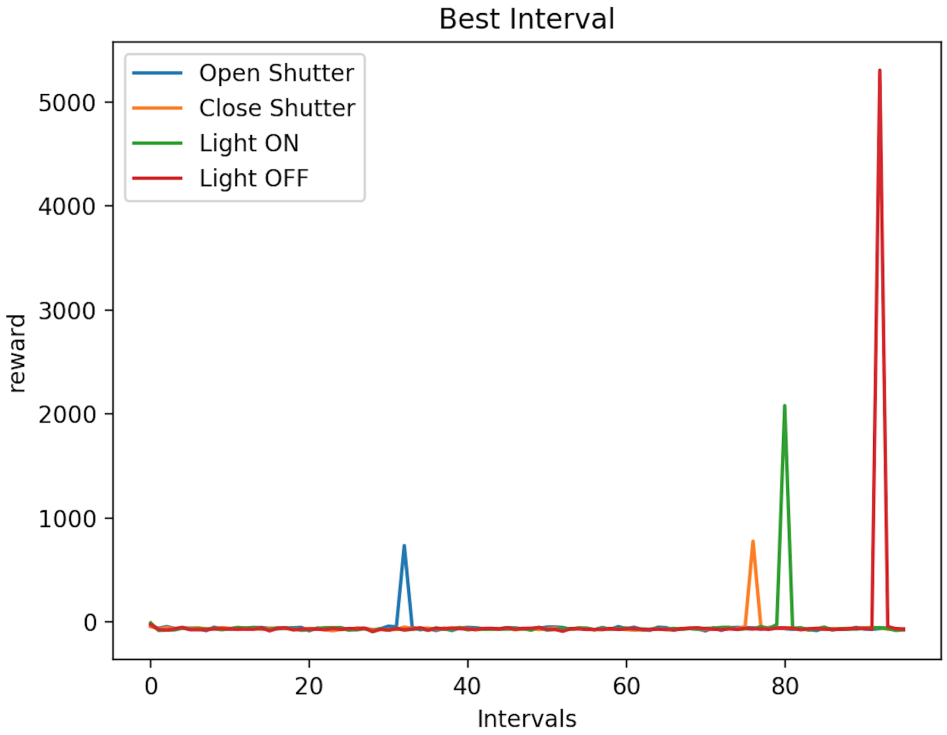


Figure 30: Binary reward function

For example, when the agent predicts the most suitable hour to turn on the lights, if it chooses to illuminate them when the user typically turns them on ($a1$) it receives a positive reward, otherwise ($a2..n$), receives a -1.

This version has resulted to be insufficient, has the agent only knew if it got correct or incorrect, not knowing how it could do better to get it right in the next iteration. To resolve this, a binary reward function is not the answer. In figure 30 we can see that the agent has explored in an inefficiently way the possibilities, this binary reward function does not influence exploration, or in another word, learning.

We want to provide the agent with some clues about their behaviour, so, if the agent prediction is close, we give it a minor reward, indicating he is getting there, in other hands, if the agent prediction is too far away, we give it a negative reward, influencing him to predict another interval. The present reward function for past activities is given below:

$$past\_reward = \frac{\log_{10} day}{dist}$$

The equation has 2 key elements, $\log_{10} day$ and $dist$. $\log_{10} day$ represents the different weight for most distant days condition defined, being directly proportional to the reward value. $dist$ represents the distance relative to the agent chosen interval prediction and the user historical data interval, being inversely proportional to the reward value. The closes days have a more weight in the reward, as well as the prediction accuracy of the agent. $dist$ can be negative, resulting in a negative past reward.

Negative rewards are a must in reinforcement learning, we have learned that if we only give positive rewards, even if they are differentiated, the agent gets confused, he always thinks he got it right. A negative reward is a good way to tell the agent his prediction is wrong, forcing him to find another solution.

For the user activities data, in this prototype, we use a total of 28 past days (representing historical data), corresponding to 4 weeks. The $\log_{10} day$ function is ideal to adequately distinguish the far-off days, defining a proper curve to this end. Figure 31.

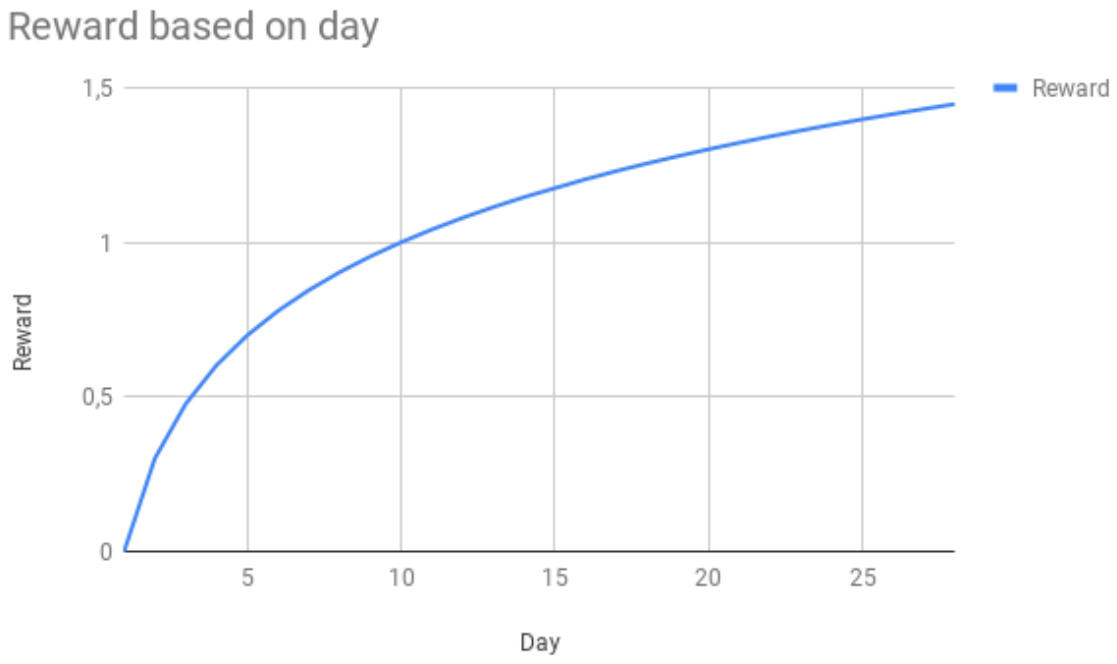


Figure 31: $\log_{10} day$ function

The parameter $dist$ represents how for much the agent allegedly failed the prediction, varying between 0 and 95. In our system, when the agent fails by 5 or more intervals, a negative reward is given, based on the number of intervals it failed. When the agent predicts the

exact interval or fails by 5 or fewer intervals, it receives a positive reward, the closer he is by the actual interval observed in the historical data, the greater the reward. This is represented in figure 32.

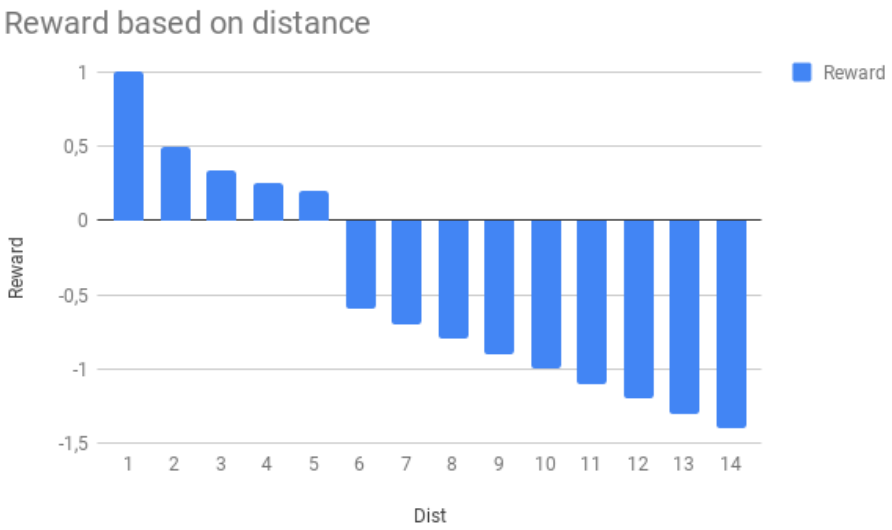


Figure 32: *dist* based function

In this figure, only 18 intervals are represented. However, we can clearly perceive how the reward variate, being its value dependent on the missing margin. When we combine these two elements of the past reward function ($\log_{10} day$ and *dist*), we are present with a solid and complex past reward function capable of properly resolving our problem. This sub-reward function has the same data structure as the Reward Total.

Figure 33 illustrate the reward distribution over the intervals with this last reward function. The agent has clearly explored in a efficient way the environment, getting a vast panoply of reward values.

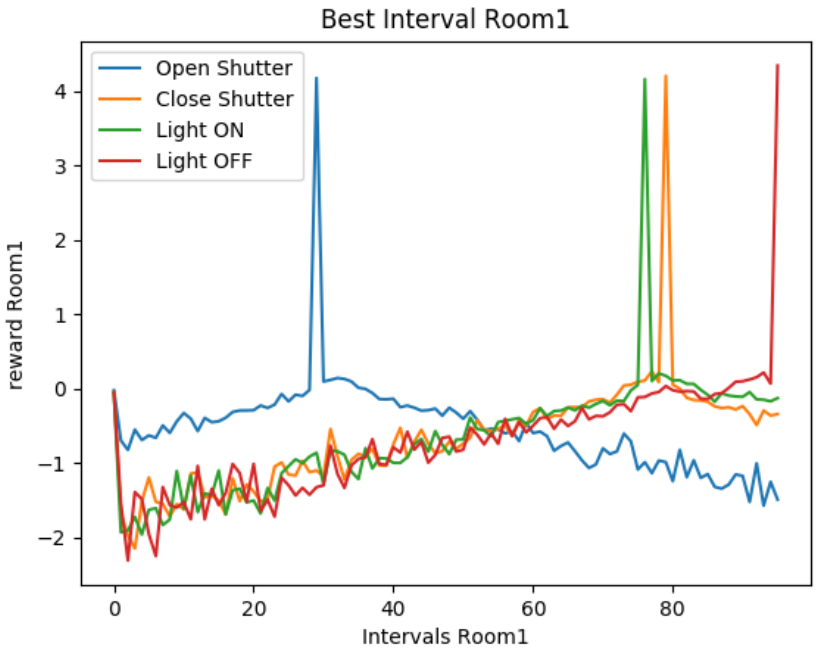


Figure 33: Reward final version

The API data is added to the reward function after we calculate the value of the user activity function for a determined action for a determined day and room. This new reward resulted from the sum of the past reward function with the API data, is updated in our neural network. Consequently, this new score can change our agent prediction in the subsequent iterations. The same procedure is made until the last iteration, defined by our developed algorithm (Epochs). API data in our project context, is external data from our environment acquired through API's, for example, the meteorology prediction. The reward function is prepared to receive this data, independent of the API source.

The other portion of the total reward is updated in real time. It is not a prediction; it is a surplus value added based on the sensing and observation of the environment at that precise moment, for that interval of time. For testing purposes, we did not wait for a full day of the system execution to see the agent action to each interval. As a replacement, we simulated days of system execution by generating a new source of data, containing the "real-time" values of the sensors. This way, after the agent assigns a value to each interval corresponding to its prediction, he checks the recent column simulating the sensor value for that specific interval and updates the total reward with a new value. For example, the agent predicts the best interval to open the shutter is at 8:00h, however, when this hour comes, he checks the luminosity sensor and updates the reward value for open shutters. If the

updated reward is still higher than a defined threshold, the agent maintains his decision. Contrary, if the updated reward is lower the defined threshold, he modifies his decision regarding the most appropriate interval to open the shutters. This step is essential to avoid a decision when the conditions doe not meet, in the same examples, it does not make sense to open the shutters if its raining and there is no sunlight, even if the system user normally does it at that certain time.

For last, and very important, the user feedback. The appreciation of the user is added to the final reward function to that specific action at a determined time and room *reward[room, interval, day Week]*. To this end, we need to establish a connection between the intelligent agent and the user, originating the interface agent. After each day of system execution, our intelligent agents predict, based on their experiment with the environment, the most appropriate interval of the day to automate an action. These multiple decision makings are passed to the responsible interface agent for reaching these decisions to the user. However, the user can communicate with the system to, influencing it to change its behaviour and decision in the future.

$$Total\_reward = Total\_reward + User\_Feedback$$

where: $User\_Feedback = 100$, if user approves

else $User\_Feedback = -100$

The $user- > system$ communication is simple, the user narrowly approves or allegedly denies the action taken at a specific interval, when the user approves it, the final reward acquires a significant amount of extra positive reward. In other hands, when the user disapproves, an immense negative value is added to the final reward.

4.4 APPLICATION

We have developed an application in our developer environment, with the objective of evaluating our system. Our application has a simple user interface, permitting the introduction of new profiles for automation. A user profile is defined in a JSON file, containing the configuration of an environment. The process of loading a profile to our application is accomplished by simple select a required JSON file in the system explorer, represented in figure 34.

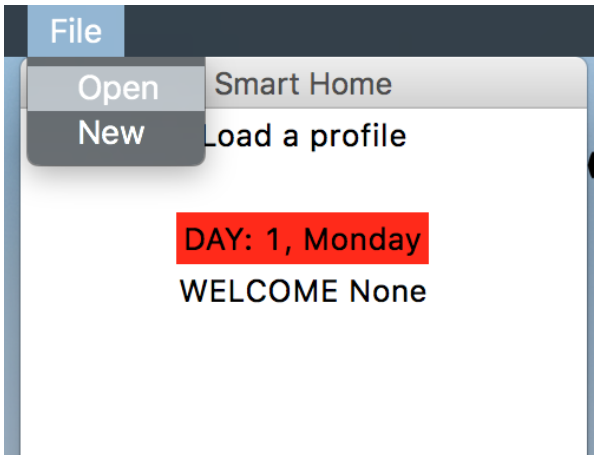


Figure 34: Load a user profile

The user can afterwards, select which component he wants to automate, having total control over deciding which room and appliance he wants to automate, as well as the source of data used by the algorithm.

Note that we have two options regarding the chosen algorithm. We have implemented two different algorithms responsible for the agent decision making. As we can see in the figure 35, there is a reinforcement learning and a neural network button. These algorithms will be discussed more further, being equally analyzed and compared.

This interface was constructed with the help of TkInter. [1]. We aimed to develop a simple interface, as it serves to test our system adaptability, and in a sense, the communication between the system and the user. We have also introduced the ability to select various execution iterations, allowing to observe easily the algorithm evolution.

1 Tkinter is a Python's de-facto standard GUI (Graphical User Interface) package.

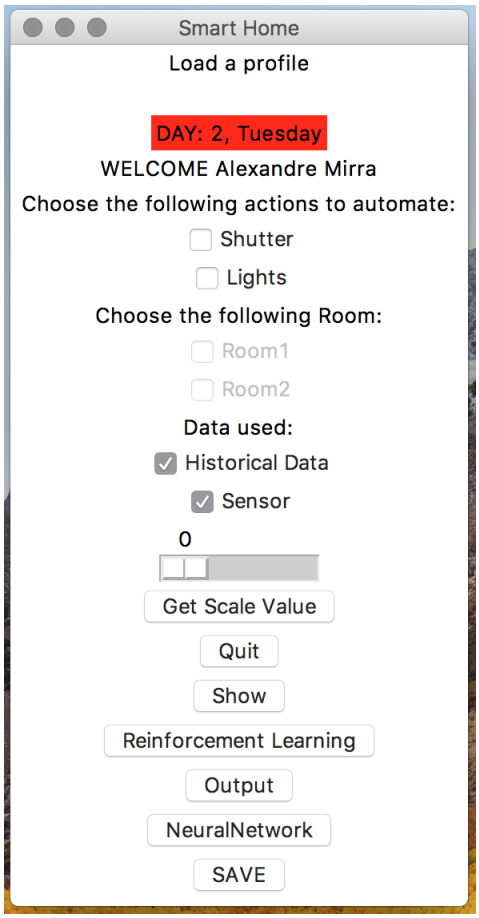


Figure 35: Main screen

The application after execution will display the decision to the user by its interface agent through visualization, informing the user the best intervals to automate the actions he requested. Alternately, the user can ask for a specific interval, being answered by an action (agent prediction).

We can save our predicting model whenever we want. When we save, an instance of the prediction model is transcribed in the respective profile (JSON file). A saved profile can be loaded afterwards, allowing portability of an environment between different machines.

The application has changed through the whole development of our system. It has evolved to meet our criteria. The possibility of having a functional application to test our architecture and solution has accelerated the development process to an optimal resolution.

This application has proven to be essential, as it provides the tools needed to validate and analyze our different implementations. In the section below, explanation and testing will be accompanied by this application.

4.5 ANALYSIS

The multi-agent system developed has demonstrated to be suitable for our project. It provides active communication between the end-user and the application, enabling an adaptive and evolving system.

The application serves as a client to the end-user, providing a direct interface where its possible to execute all the features of our project. This interface must be enhanced to be applied in a real-world environment, but, for testing purposes, is more than enough.

Our intelligent agents are capable of having multiple machine algorithms implemented, trough the application and the interface agent, one is called to be the decision making responsible. We have applied two different algorithms in our intelligent agents, present in the logical unit. One of the approaches has proven to be inadequate to our problem. As we mentioned hitherto, our system is projected to enable communication in both directions, however, with the neural network algorithm, this communication had no effect in the decision making.

On the other hand, the reinforcement learning algorithm has demonstrated to be flexible and adaptive, taking user feedback into consideration when predicting the most suitable action. It additionally provides user comfort and control, responding to our demands. We have spent the most of our time developing this algorithm after we pointed as the most appropriate to this project. We believe that is capable of automating numerous actions, regarding the environment where it is installed, as it supports different environments through profiles.

We have developed a system that permits expansion. Hence, the number of features and the numbers of appliances possible to automate are not fixed. With our multi-agent system with intelligent agents equipped with a reinforcement learning algorithm, it is possible to include new appliances and even more conditions in the decision making. Our reward functions are properly prepared to receive more parameters. The reward system is complex and robust, leveraging adaptation. The solution has satisfactorily proven to be dynamic.

In the following section, we will implement our system to hypothetical cases, demonstrating its function in such cases. All functionalities present in the application will be covered and analyzed.

5

CASE STUDIES

In this section, we will demonstrate the system operation in different cases studies created. The results will be presented and subsequently analyzed. We have selected two user story's, representing different behaviors in our application. These cases distinguish between them, arguably showing the system capability of adaptation.

The user activities data will be represented by historical data, as we do not have sensors capable of generate real-time data. As we are using simulated data, this will be considered past user activities. These simulated data will be used to prove the capability of the system to use user activity data, captured in real time.

5.1 EXPERIMENT SETUP

Before conducting the experiments, we must define a profile to be loaded in our application.

We have started creating a profile, representing an environment with two room, a bedroom1, and a bedroom2. This home is inhabited by two single individuals, which one is a day laborer and the another a night worker. These two live in the same home, however, we must distinguish the rooms, as they are inhabitant by two different users. For this section, we have prepared the system to automate two appliances, the lights, and the shutters.

Bedroom1, the day laborer:

He normally gets up at 7:30h, however, there is days that he gets up later or even earlier. The time to get home also varies, as so, he does not close the shutters at the same time,

however, the first thing he typically does when arrives at home is closing the shutters. As he needs waking up early in the following day, he normally goes to bed at 23h, turning off the lights to sleep. At weekends, as he doesn't have work, he gets out of bed at around 10h. These records are from the last 28 days, 4 weeks.

Bedroom2, night worker:

Bedroom2 is inhabited by a nocturne laborer. This individual differs from the other present in the bedroom1 dramatically. He wakes up around 15:30h and immediately opens the shutters. Before going to work, he closes the shutters (around 21h). This individual only needs to turn on the lights when he arrives at home after the work, as a night worker, he arrives at 5:00h. Just before the bedtime, he turns off the light.

With the user story specified, we can subsequently create the profile. There are two rooms in this environment to automate, thus two elements in the room list are needed. These elements point to a certain bedroom present in the environment, containing the past 28 days information about his habits and actions. The reward matrices are initialized at zero, as we do not have previous knowledge of this profile. And finally, the name of the profile is created (working as an identifier) and the present day is defined (day 29). This way the algorithm knows that working with 28 days of data. This is in a JSON file like the one represented below.

```
{
    "day": 29,
    "dayWeek": 0,
    "name": "Jose Mirra",
    "reward_pa_close_shutter": [...],
    "reward_pa_light_off": [...],
    "reward_pa_light_on": [...],
    "reward_pa_open_shutter": [...],
    "rooms": [
        "BedRoom1",
        "BedRoom2",
    ]
}
```

The past user activities referring to a specific room need to be generated through our developed scripts, as we do not have real data. To generate the data, we have analyzed the user story's and we started to simulate their actions. To add randomness in the interval at which the user conducts his actions, we applied a normal distribution, in other words, the data-set contains some noise, adding some challenge to the algorithm to find a pattern. For example, in the case of the day laborer, the referring interval to open the shutters at weekdays has a mean of 30 (represents 7:30) with a deviation of 1 interval. Represented in figure 36.

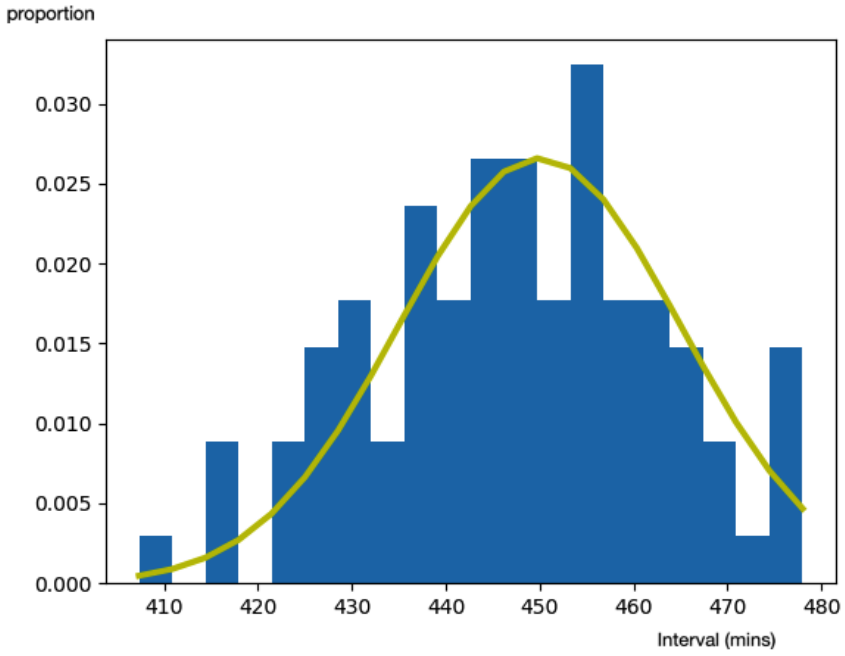


Figure 36: Normal distribution of the selected time to open the shutters in weekdays.

For each action, we have adopted this strategy to simulate variation in the data. For each day, a different interval is picked according to our normal distribution. We have defined multiple distributions, differing in the parameters, reflecting the different actions and the days of the week. Each action has its own mean and deviation value.

The generated data will after be interpreted by our agents as *DataFrame* object type. In order to do this, we will us the Pandas library. Pandas provides data structures and data analysis tools that make manipulating data in Python much quicker and more effective. It has a lot of built-in functions for analyzing data, such as looking up columns by name. Pandas gives us access to these features, and generally makes working with data much simpler. An example *DataFrame* for the simulated data, figure 37.

	day	hour	minute	sensor_shutter	dayWeek
116	2	5	0	0	1
117	2	5	15	0	1
118	2	5	30	0	1
119	2	5	45	0	1
120	2	6	0	0	1
121	2	6	15	0	1
122	2	6	30	0	1
123	2	6	45	0	1
124	2	7	0	0	1
125	2	7	15	0	1
126	2	7	30	1	1
127	2	7	45	1	1
128	2	8	0	1	1
129	2	8	15	1	1
130	2	8	30	1	1
131	2	8	45	1	1
132	2	9	0	1	1
133	2	9	15	1	1
134	2	9	30	1	1
135	2	9	45	1	1
136	2	10	0	1	1
137	2	10	15	1	1
138	2	10	30	1	1
139	2	10	45	1	1
140	2	11	0	1	1

Figure 37: DataFrame of the past activities

With a very low probability, 5%, the simulation will not follow the logic presented. A random value to the sensor attribute will occur to introduce impeccability in the system. There are errors in sensing a real environment and rarely, the user can take unpredictable actions, outside his routine.

5.2 RESULTS

In this section, the results obtained are discussed for the case study presented. It is considered the light and shutter appliances.

In the results presented in this section, we consider 96 intervals, each one of 15 minutes, representing a day. The training procedure consists of 1000 repetitions of daily data. In order to explore all intervals defined, the probability to choose a random interval was delineated to 40 percent.

The system does not start without knowledge. We have loaded the profile with 28 days of previous user activity, representing 4 full weeks. In his first execution, it will predict the following day, 29 , a Monday. The end-user, "Jose Mirra" in this case, has selected *Past*

activities as source of data and lights and shutters as the appliances to automate through the application, as shown in figure 38. This means that the machine learning algorithm only takes on consideration the user activity in his decision.

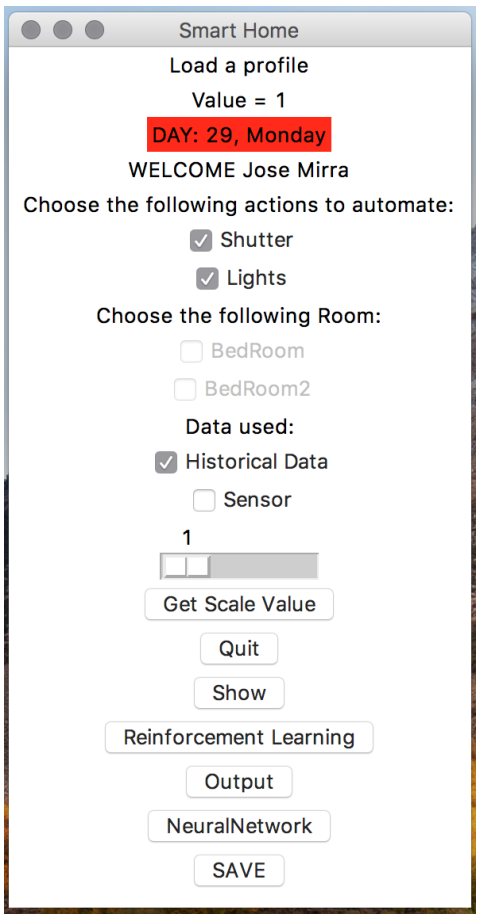
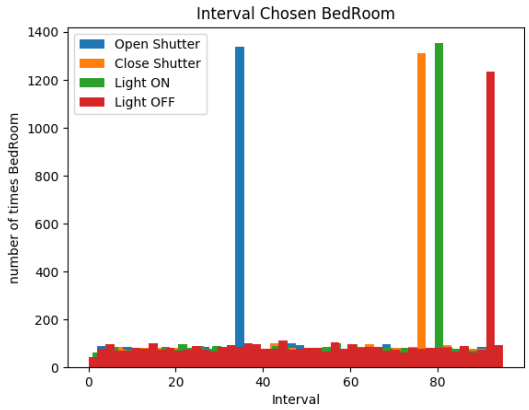
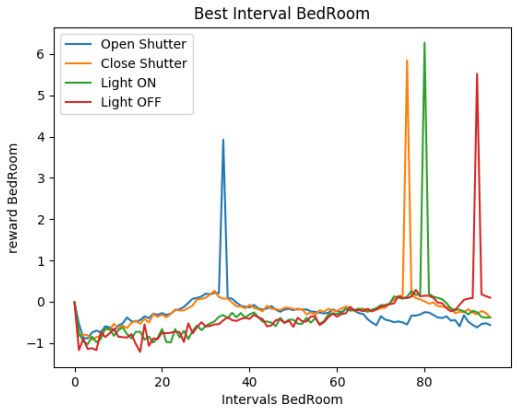


Figure 38: Application interface in this case study, for one day of execution

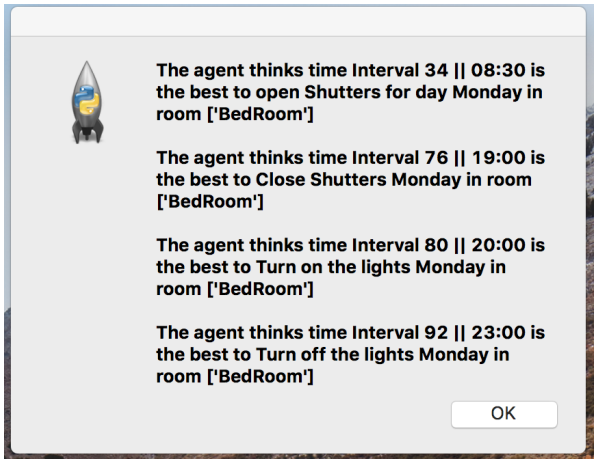
With 28 days of training, the algorithm can predict the best action for the available appliances in the environment for the next day, being 29 (Monday in this case). With 28 days, only 4 days were took in consideration by the intelligent agents, as in 4 weeks, there is 4 Mondays. The following figures, figure 39 and 40 represents the decision making on both rooms, respectively in 1 and 2, for Monday (week day).



(a) Agent chosen action



(b) Rewards for the different intervals and actions

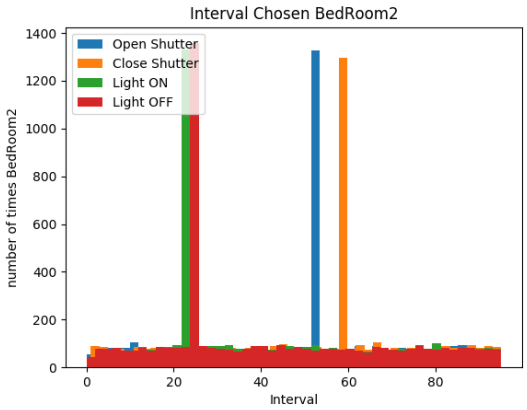


(c) Decision making output

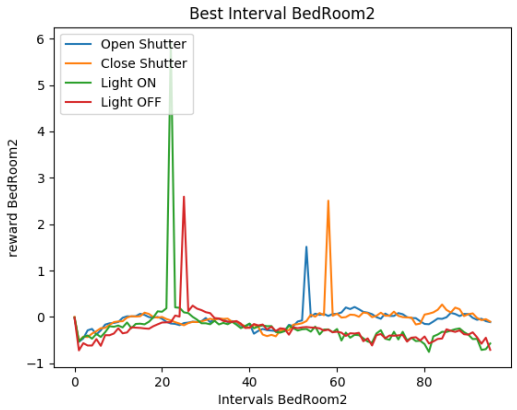
Figure 39: Decision making for BedRoom1 at Monday

BedRoom1 is inhabited by a day laborer. It can be clearly seen in figure 39a, that the agent inclines significant more to a determinate interval than others, when considering the action. This is caused by the policy based neural network prediction. For example, the agent predicts that the interval 34 is best to open the blinds in this room (figure 39c), representing 8:30 am, which makes sense for the day laborer we defined.

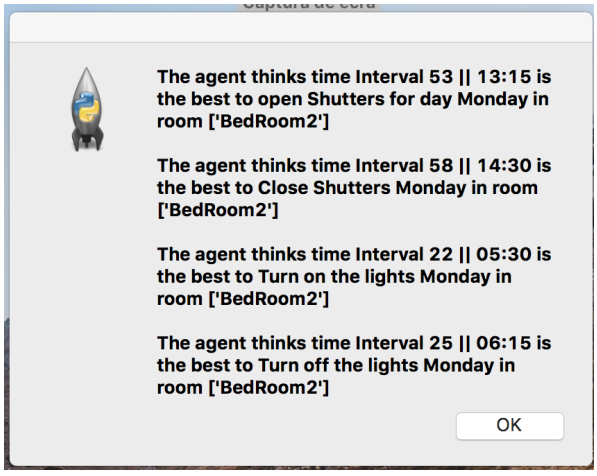
For each action chosen, the agent will confiscate his reward. On figure 39b, we can see the reward associated to each action at a certain interval. In this case, our algorithm prediction is according to the system reward. Seeing the shutter example once more, interval 36 is the interval with most reward for opening the shutters, solidifying our agent prediction.



(a) Agent chosen action



(b) Rewards for the different intervals and actions



(c) Decision making output

Figure 40: Decision making for BedRoom2 at Monday

BedRoom 2 is inhabited by a nocturne laborer. Figure 40a represents the interval predicted by the agent. It differs significantly when compared to room 1, with this, it can be affirmed that the system is capable of adaptation to different behaviors in the environment.

Figure 40b represents once more, the reward associated with each action at a certain interval. Some actions have a bigger rewards than others, for example, our system is more confident at the hour at closing the shutters than to open it. This is simply because of the variation of the waking time in the user past activities. The more unpredictable the user is, the less confidence the system is.

We can testify that the system easily distinguish the different rooms, it treats them as individual components in the decision making process. However, the result for both rooms is

presented in the application, managed by an intelligent agent. A user is responsible for giving his feedback for both rooms, as they are defined in one profile. To be a full dependent system, different profiles had to be made, each with the specific user and room. For testing purposes, we have grouped them in the same profile, easily representing a family.

After seeing the results for both rooms, the user happily approves the decision of the system. Trough the applications he introduces the intervals chosen by the agent and displayed to the user by the interface agents. He informs the approval with a "YES" for all the 4 intervals, one for each action. This way, the system is more inclined to the same intervals in the next Monday. Represented in figure 41.

```
Please enter an interval of the day (0-96):  34
You entered 08:30
The best action for Interval 34 is Open Shutter in the day Monday
Do You Agree? (YES/NO) :  YES
You entered YES
Open Shutter
Conclude? 0/1 : 0
Please enter an interval of the day (0-96):  76
You entered 19:00
The best action for Interval 76 is Close Shutter in the day Monday
Do You Agree? (YES/NO) :  YES
You entered YES
Close Shutter
Conclude? 0/1 : 0
Please enter an interval of the day (0-96):  80
You entered 20:00
The best action for Interval 80 is ON Light in the day Monday
Do You Agree? (YES/NO) :  YES
You entered YES
ON Light
Conclude? 0/1 : 0
Please enter an interval of the day (0-96):  92
You entered 23:00
The best action for Interval 92 is Off Light in the day Monday
Do You Agree? (YES/NO) :  YES
You entered YES
Off Light
Conclude? 0/1 : 1
BedRoom2
Processing Next Room
```

Figure 41: Feedback process for this case study

Lets go ahead 6 days and see how the system behaves in a day of the weekend for the user at the BedRoom1, as he does not work at weekends. Fortunately, we do not need to wait 6 days to see the results. In the application, we can control how many days we want to iterate (figure 42. This way, only the day we want is displayed in the output, however, the algorithm internally iterate all days that were introduced in the interface, populating data in the historical data-set. Figure 43 represents the decision for the day 35 , Sunday. The user feedback will be assumed as positive when iterating various days.

The intervals predicted are different, resulting in the change of behaviour of the user at weekends. In figure 47a, we can see that the agent has predicted various possible intervals for opening the shutters. This has occured, as the waking time of the user at Sunday is very variable, as he do not have compromises. The system has successfully distinguished the day of week in the decision making.

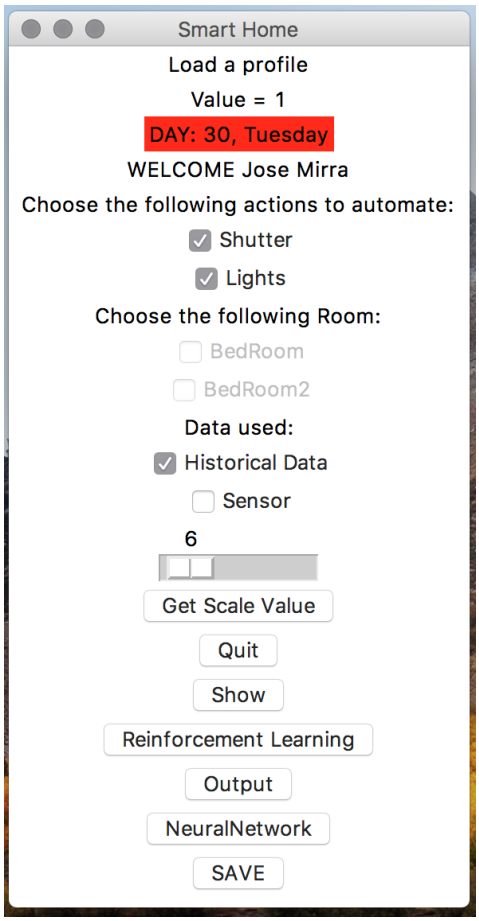
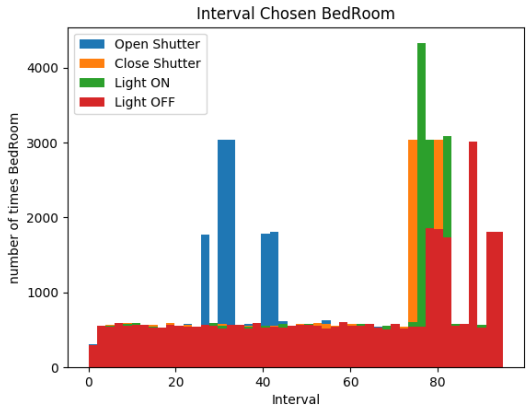
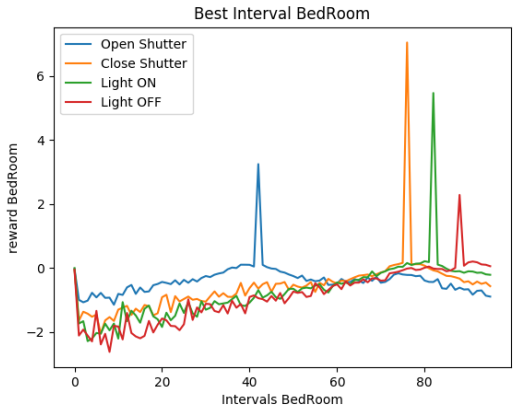


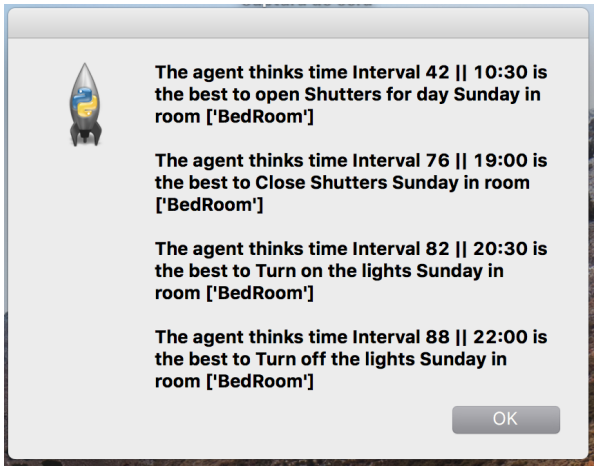
Figure 42: Application interface in this case study, for 6 days of execution



(a) Agent chosen action



(b) Rewards for the different intervals and actions



(c) Decision making output

Figure 43: Decision making for BedRoom1 at Sunday

Before continuing, we can save the profile. This process saves the state of the predictive model, enabling portability and convenience. Every variable is saved, maintaining the model knowledge.

Let's now see how the system behaves to more 2 months of the same lifestyle. It's expected to predict the same intervals as before to every action as the model we are using is pre-trained with approval of the end-user, as it has more data and feedback helping his decision. Figure 44.

Intervals for 56 days (4 weeks).png

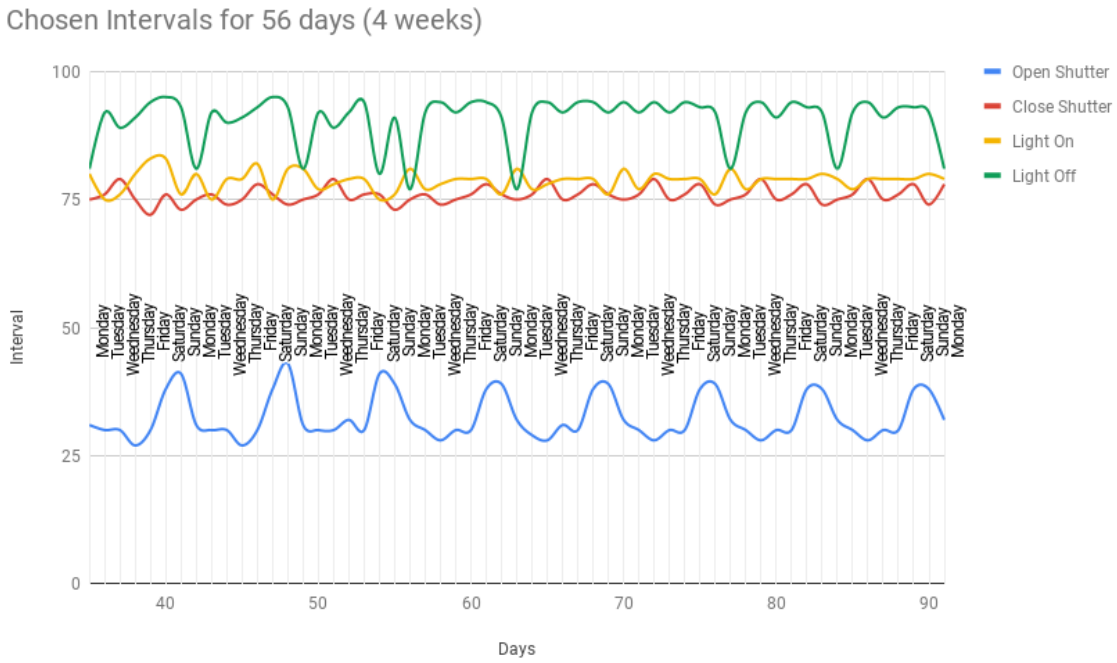


Figure 44: Chosen Intervals for 56 days (4 weeks) BedRoom1

We can easily see a pattern in the figure above. Also, the weekends are easily differentiated for the rest of the week.

Another experience, is to use different parameters as input in our Application. We load once again the profile saved before and instead of only selecting historical data, we can select "sensor" option, corresponding to the predicted luminosity. We expect to obtain different results than the experience before (45).

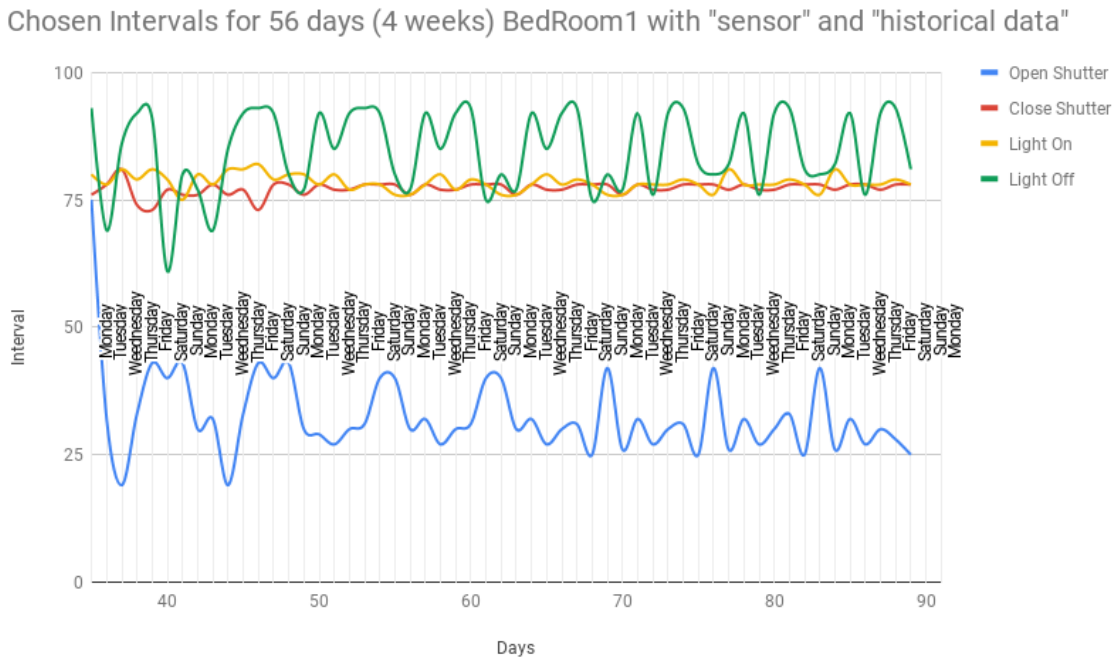


Figure 45: Chosen Intervals for 56 days (4 weeks) BedRoom1 with ”sensor” and ”historical data” as source of data.

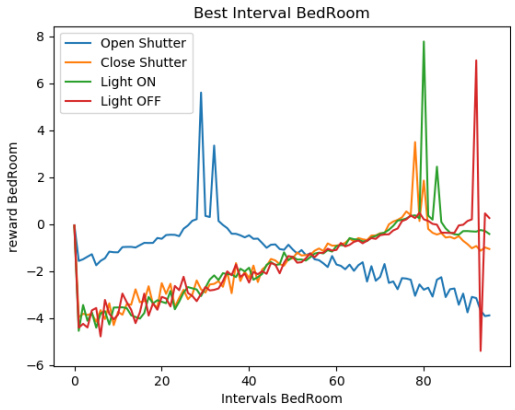
Some appliances automate in different intervals, in ones which are not frequents. This happens, because the value of the luminosity is sufficiently high to change the agent decision. For instance, at the first Monday, the agent has decided to open the shutters at a very late time, independently of the user behaviour. This can be explained by very low luminosity value (normally represented by bad weather) and it does not compensate to open the shutters when this happen. A pattern can also be seen, but not so well defined, caused by outlier actions.

In order to show the impact of the user on the system, lets go back and remind the day laborer example in BedRoom1 for day 29 (figure 39). We load the profile within it's initial state, without training. This time, after the first iteration, we change the end-user feedback to be negative. Figure 46 represents the carried out process to indicate the system of the unwanted action, consequently in the next Monday, the system will change his decision, represented in figure 47. As expected, the user has a great impact in the system decision, giving more weight to the user feedback than to the historical data in the reward function.

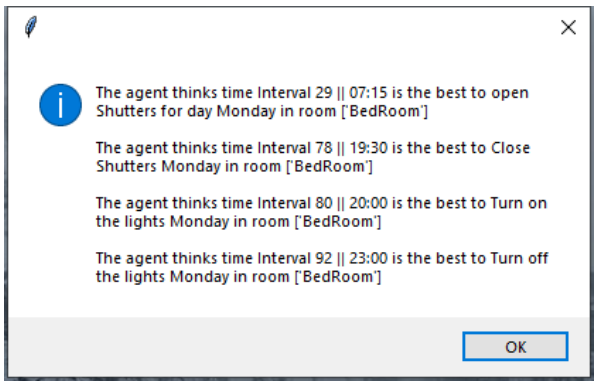
This exact exercise was carried in other machine, with a different operative system, noticeable in the figures below by a different interface. We did this in another machine intentional, with the aim of proving the portability of this solution.

```
The agent thinks time Interval 32 || 08:00 is the best to open Shutters for day Monday in room BedRoom
The agent thinks time Interval 78 || 19:30 is the best to Close Shutters Monday in room BedRoom
The agent thinks time Interval 80 || 20:00 is the best to Turn on the lights Monday in room BedRoom
The agent thinks time Interval 93 || 23:15 is the best to Turn off the lights Monday in room BedRoom
Please enter an interval of the day (0-96):  >? 93
You entered 23:15
The best action for Interval 93 is Off Light in the day Monday
Do You Agree? (YES/NO) :  >? NO
You entered NO
Off Light
Conclude? 0/1 : >? 1
```

Figure 46: User disagrees with the action of turning off the lights at interval 93.



(a) Agent chosen action



(b) Rewards for the different intervals and actions

Figure 47: Decision making for BedRoom1 at Monday after user feedback

Different features and examples were presented, however, it will be interesting to see how the system reacts to a drastically change of behaviour, with and without feedback. This last addiction can demonstrate the effectiveness of a feedback system in this case examples. In the example before, we have demonstrated the feedback influence to one day example.

An interesting experiment is to swap the Rooms, that is the inhabitant in the BedRoom1 goes to the BedRoom 2 and so on. With this, the model trained for the user present in the BedRoom1 will be occupied by another user, with a drastically difference in his behaviour. It will be interesting to see how fast the system converge to new lifestyle, with or without feedback. Once more, we will use a pre-trained model, corresponding to the one demonstrated in 48, containing the user habits in BedRoom1 with 90 days of knowledge.

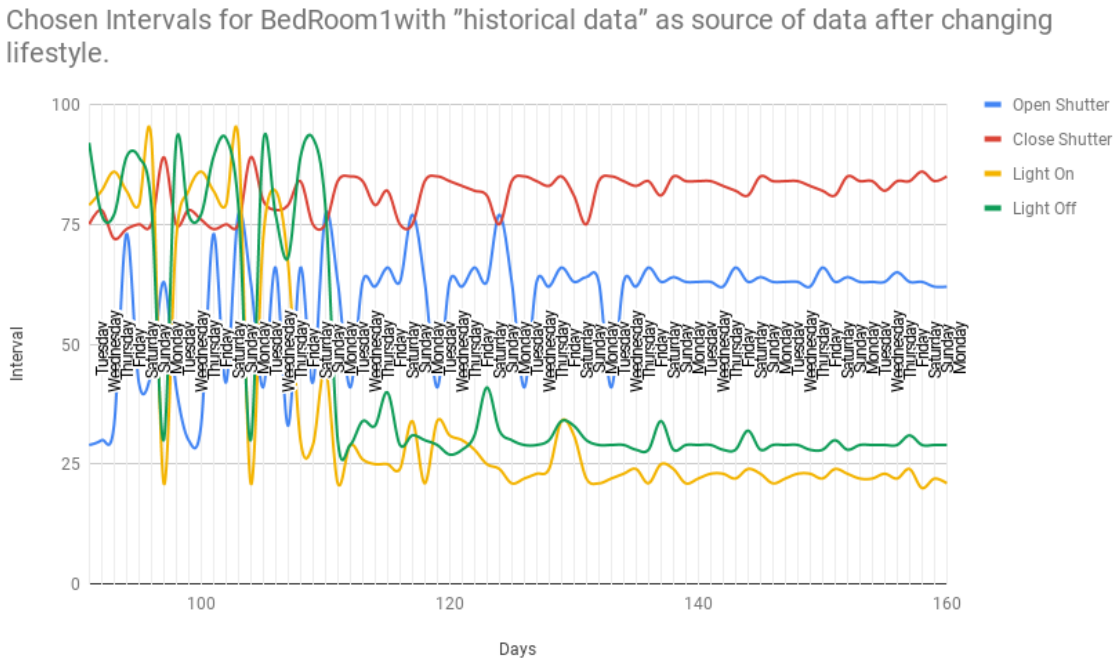


Figure 48: Chosen Intervals for BedRoom1 with "historical data" as source of data after changing lifestyle.

For the first week with the new habits, the system will still predict the old habits actions, as it uses the last week actions to predict. However, after the first week, the system identify a dramatically change of behaviour and tries to converge to a new set of predictions. This adaptation takes times, as the system gets new data to work with every week. As we can see in the figure above, there is a number of days of adaptation, where the system is still finding the new habits. Meanwhile, after the adaption interval, it starts to stabilize, finding a new pattern in the user actions.

The introduction of feedback can accelerate the adaptation phase, helping the system reaching the stabilization state in a faster way. This process requires user interaction, being dependent of his participation. Figure 49 represents the system chosen intervals for a total of 70 days of a new lifestyle, the end-user has contributed with his feedback in order to accelerate the learning process.

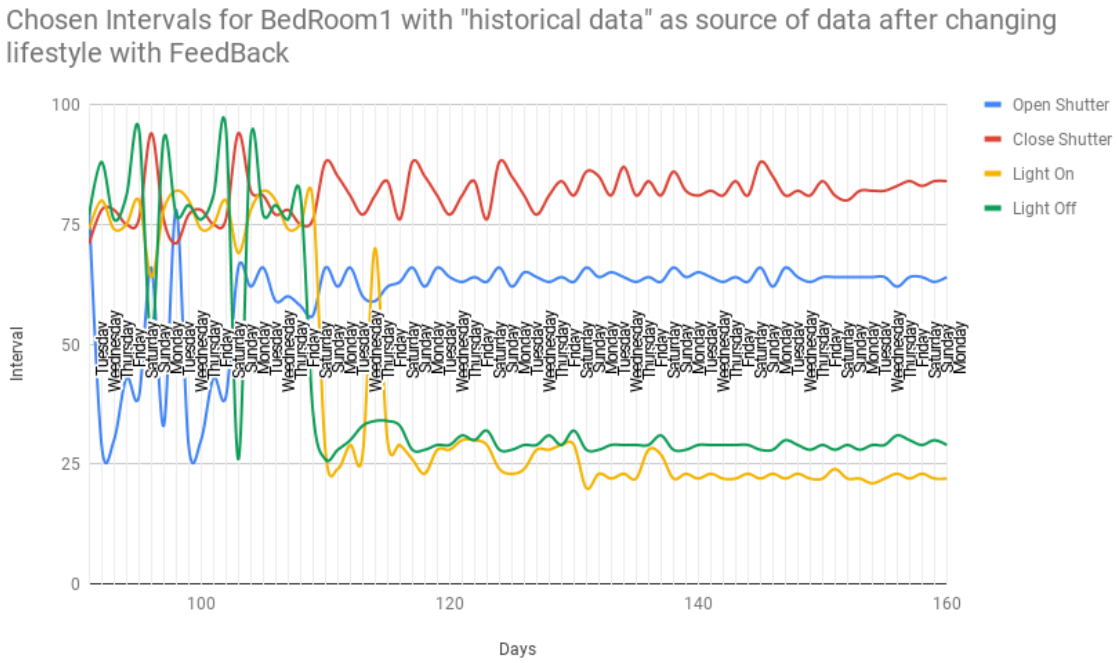


Figure 49: Chosen Intervals for BedRoom1 with ”historical data” as source of data after changing lifestyle with FeedBack.

We can observe the differences with and without user feedback. The learning process is substantially faster, as the system receives an extra hint when deciding. In the production of this specific example, feedback was given every day in the first week of iteration, representing the critical phase of adaptation. After the end-user input, the system starts to readjust the intervals, avoiding selecting the unwanted ones. When the system finds new intervals, we provided feedback once more, with the objective to incentive the system to follow this new chosen intervals.

In summary, in the first iterations, negative feedback was given to point incorrect decisions. In the adaptation phase, both positive feedback and negative was given. In stabilization phase, mostly positive feedback was given, in order to keep the system motivated to follow the chosen intervals.

5.3 ANALYSIS

We have demonstrated the capabilities of our developed system. It has proven to be capable of choosing an interval to automate an action based on the user habits. It equally can capture external data, as sensor data and weather predictions, such as luminosity, as conditions when making a decision.

The system easily identifies patterns in the user behaviour, distinguishing the different days of the week, accurately predicting the intervals to each day. Each room present in the environment is also clearly detected, treated them independently when automating the appliance.

The chosen intervals to automate an action can be changed by the user feedback. The end-user feedback is extremely influential in the decision making, capable of modifying the course of the system. This is ideal, as the system objective is to satisfy the user, adapting to his life.

The system has been tested to a significant number of days, showing its capabilities of continuing providing an automation solution. The save and load method has proven to be useful when we want to transfer the predictive model to another environment, in case the user changes his room, or even, his home.

The capability of the system converges to a user lifestyle was tested and successfully proven. Even if we dramatically change the user lifestyle, the system can detect this change of behaviour and rapidly finds a new solution. The feedback system can be extremely useful in these situations, as can accelerate this process of adaptation.

6

CONCLUSION

This chapter aims to present a summary of the work carried out in this project, as well as some conclusions and presentation of the scientific contribution. There are also some aspects in which the project can be improved and some ideas for future work.

6.1 WORK SUMMARY

Smart Homes has remained until now an exciting concept, even after decades of research. The rise of IoT devices has brought smart environments to a level never seen before. A smart home contains a connection between wireless communication, sensors, monitoring and tracking. Smart homes are an enormous system that includes multiple technologies and applications that can be used to provide security and control of the home easily.

This project focuses on automate various appliances present in a Smart Home environment, extracting information from various sources to make this possible. Making this information use-full can be a challenge, as the literature suggests. Quantitative information about this subject is not a problem, although find qualitative information in such vast repositories was not a trivial task. The fact that this project was done with a real company in a smart home context, it helped resolve this problem.

Through the vast literature analyzed, we were presented with a vast number of algorithms capable of this process. However, they require an invasion in the end-user privacy and comfort. We have learned the existence of social barriers, that worries and prevents the

embrace of this theme. Our solution is focused on the user comfort, for this reason; a complex system was created with this in mind.

After clarifying concepts and analyzing the related work, we have started to develop a system capable of automating various appliances present in a Smart Home. The system is fully configurable, where the source of data and application are customized. A feedback feature was successfully implemented, turning possible a bi-directional communication between the system and application.

The system was tested based on its ability to predict the best interval present in a day for lights and blinds in different rooms present in a home. User Feedback was also been tested and it demonstrates how influential it is upon the system. Testing our system has proven successful, we have concluded the capability of the system in a real-world scenario. We have built a prototype and we conducted experiments with this in mind, constructing a future-proof system, capable of working with dynamic data and user behaviour, like in a real-world scenario.

A set of challenges were raised while developing the system. The alluring possibility of working with real data provided by IoT devices owned by the company did not happen. Working with clients, with real environments was impossible, arising the need of developing a simulated environment. A study alongside the company was made to replicate by an accurate way a real case study. The system was built and tested to satisfy this change.

Concluding, the system developed is capable of working with a proper case study, as we have prepared it to do so.

6.2 REVELANT WORK

As presented in the work plan of this dissertation and parallel to the development of this project, it was intended to make a scientific contribution with the publication of a scientific article, which was presented in Rome at the conference IE'18: The 14th International Conference on Intelligent Environments - IE '18 Rome 25th-28th June 2018, in the Citizen-Centric Smart Cities Services (CCSCS'2018) workshop.

(Mirra et al., 2018)

As relevant work in the paper is to highlight the reinforcement learning approach conducted. Unlike the previous reinforcement learning approaches, the paper highlights the role of the inhabitant to combat the social impasses of Smart Homes. With the help of deep learning and neural networks, the ability of the system to predict the subsequent action given the state of the environment and the user pattern was successfully demonstrated based on non-supervised techniques. The results shown in the paper reveal that the model was able to adapt to different usage patterns and discern between different contexts in the form of user preferences.

Alongside the industrial partner, a prototype system was built to test the proposed solution to a real case scenario. A case study was created based on a real case, accurately replicating the behavior of the system in a real house with a real user. The project was continuously monitored by the industrial partner, thus keeping a non-academic side to the work.

6.3 FUTURE WORK

Although the system developed has satisfied our raised objectives, there is space for improvements. Testing our system to a real environment is the next step to do. This way, we can prove the robustness of our system implementation, that was built with this challenge in mind.

An improvement in the interface represents a process to lift the system capability of gathering feedback. In the current state, the system captures feedback with a simple application. However, as future work, this interaction must be planned to be more efficient and natural. A mobile application would also represent a plus in our system, as it can be an easy platform to gather user feedback and even record user activities.

More sources of data and appliances would be a good addition to our solution. Every intelligent agent presented in our system is prepared to automate a specific action. Notwithstanding it is capable of accepting more agents, representing this way more appliances. The introduction of a new intelligent agent represents the addition of a new appliance, there is no need to alter the rest components of the system. Adding new appliances to an existing system is always possible.

The increase of sources of data would require an increasing complexity of the reward functions. Nonetheless, the reward functions need to be tweaked, increasing the complexity

by including new parameters. To use real devices as sources of data, their communication with the system needed to be implemented.

As the system monitors the environment, an alarm feature would be interesting to implement. For example, when our agents detect abnormal behaviour or even abnormal data from sensing, the system could send an alarm regarding the cause. This functionality is relative easy implemented in the logical model, however, the application is unprepared to show this.

A full hand of features can be added to the system, as it is built upon intelligent agents capable of sensing and learn the surrounded environment. The interface agent is capable of communicating these new future features to the interface, however, an improvement to the interface is always needed to accompany new features.

It is intended to increase the complexity of the reward functions, being more dependent on the user feedback and environment sensing instead of being majority dependent on their usage pattern. In this way, the aim is to encourage the user to avoid wrong habits, directing them to a routine they desire. Also, more external data from public Application programming interfaces (API's.) are planned, allowing this system to be unrestricted to sensors to perceive the environment.

It was demonstrated that this model for home appliances, but we could extend this approach to other environments, either public or private.

BIBLIOGRAPHY

Dean Abbott. *Applied predictive analytics: Principles and techniques for the professional data analyst*. John Wiley & Sons, 2014.

Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499, 1994.

Mussab Alaa, A.A. Zaidan, B.B. Zaidan, Mohammed Talal, and M.L.M. Kiah. A review of smart home applications based on internet of things. *Journal of Network and Computer Applications*, 97(Supplement C):48 – 65, 2017. ISSN 1084-8045.

Muhammad Raisul Alam, Mamun Bin Ibne Reaz, and Mohd Alauddin Mohd Ali. A review of smart homes—past, present, and future. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1190–1203, 2012.

Alex M. Andrew. Introduction to artificial life by christoph adami, telos, the electronic library of science, santa clara, an imprint of springer-verlag, new york, 1998, xviii 374 pp, isbn 0-387-94646-2 (hardback, with cd rom, dm 118). *Robotica*, 17(2):229–235, 1999.

Nazmiye Balta-Ozkan, Rosemary Davidson, Martha Bicket, and Lorraine Whitmarsh. Social barriers to the adoption of smart homes. *Energy Policy*, 63:363–374, 2013.

Michael R Berthold, Nicolas Cebron, Fabian Dill, Thomas R Gabriel, Tobias Kötter, Thorsten Meinl, Peter Ohl, Kilian Thiel, and Bernd Wiswedel. Knime-the konstanz information miner: version 2.0 and beyond. *AcM SIGKDD explorations Newsletter*, 11(1):26–31, 2009.

Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

Jason Brownlee. *Master machine learning algorithms discover how they work and implement them from scratch*. 2016.

Rich Caruana and Alexandru Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 161–168, New York, NY, USA, 2006. ACM. ISBN 1-59593-383-2. doi: 10.1145/1143844.1143865. URL http://doi.acm.org/10.1145/1143844.1143865.

Krzysztof J. Cios, Roman W. Swiniarski, Witold Pedrycz, and Lukasz A. Kurgan. *Unsupervised Learning: Association Rules*, pages 289–306. Springer US, Boston, MA, 2007. ISBN 978-0-387-36795-8. doi: 10.1007/978-0-387-36795-8_10. URL https://doi.org/10.1007/978-0-387-36795-8_10.

Diane J. Cook and Sajal K. Das. How smart are our environments? an updated look at the state of the art. *Pervasive and Mobile Computing*, 3(2):53 – 73, 2007. ISSN 1574-1192. doi: https://doi.org/10.1016/j.pmcj.2006.12.001. URL http://www.sciencedirect.com/science/article/pii/S1574119206000642. Design and Use of Smart Environments.

Diane J Cook, Michael Youngblood, Edwin O Heierman, Karthik Gopalratnam, Sira Rao, Andrey Litvin, and Farhan Khawaja. Mavhome: An agent-based smart home. In *Pervasive Computing and Communications, 2003.(PerCom 2003). Proceedings of the First IEEE International Conference on*, pages 521–524. IEEE, 2003.

Aditi Dixit and Anjali Naik. Use of prediction algorithms in smart homes. *International Journal of Machine Learning and Computing*, 4(2):157, 2014.

James Dougherty, Ron Kohavi, Mehran Sahami, et al. Supervised and unsupervised discretization of continuous features. In *Machine learning: proceedings of the twelfth international conference*, volume 12, page 197, 1995.

William H Dutton. The internet of things. 2013.

EU GDPR. Google home help. https://eugdpr.org/the-regulation/. Acessed: 2018-13-08.

Carles Gomez and Josep Paradells. Wireless home automation networks: A survey of architectures and technologies. *IEEE Communications Magazine*, 48(6), 2010.

GoogleLLC. Google home help. https://support.google.com/googlehome?vid=0-920759277500-1515065442909#topic=7029677. Accessed: 2018-01-04.

Emmanuèle Grosicki and Haikal El Abed. Icdar 2009 handwriting recognition competition. In *Document Analysis and Recognition, 2009. ICDAR'09. 10th International Conference on*, pages 1398–1402. IEEE, 2009.

Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645 – 1660, 2013. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2013.01.010. URL http://www.sciencedirect.com/science/article/pii/S0167739X13000241. Including Special sections: Cyber-enabled Distributed Computing for Ubiquitous Cloud and Network Services & Cloud Computing and Scientific Applications — Big Data, Scalable Analytics, and Beyond.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979. ISSN 00359254, 14679876. URL http://www.jstor.org/stable/2346830.

Urs Hunkeler, Hong Linh Truong, and Andy Stanford-Clark. Mqtt-s—a publish/subscribe protocol for wireless sensor networks. In *Communication systems software and middleware and workshops, 2008. comsware 2008. 3rd international conference on*, pages 791–798. IEEE, 2008.

Anil K Jain, Robert P. W. Duin, and Jianchang Mao. Statistical pattern recognition: A review. *IEEE Transactions on pattern analysis and machine intelligence*, 22(1):4–37, 2000.

Ruoming Jin, Yuri Breitbart, and Chibuike Muoh. Data discretization unification. *Knowledge and Information Systems*, 19(1):1, May 2008. ISSN 0219-3116. doi: 10.1007/s10115-008-0142-6. URL https://doi.org/10.1007/s10115-008-0142-6.

Keras. Keras documentation. https://keras.io/. Accessed: 2018-01-06.

Laura Klein, Jun-young Kwak, Geoffrey Kavulya, Farrokh Jazizadeh, Burcin Becerik-Gerber, Pradeep Varakantham, and Milind Tambe. Coordinating occupant behavior for building energy and comfort management using multi-agent systems. *Automation in construction*, 22:525–536, 2012.

Knime. Knime software. https://www.knime.com/software. Accessed: 2018-01-06.

Gerd Kortuem, Fahim Kawsar, Vasughi Sundramoorthy, and Daniel Fitton. Smart objects as building blocks for the internet of things. *IEEE Internet Computing*, 14(1):44–51, 2010.

Dave Kuhlman. *A python book: Beginning python, advanced python, and python exercises*. Dave Kuhlman, 2009.

Maurizio Lenzerini. Data integration: A theoretical perspective. In *Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '02, pages 233–246, New York, NY, USA, 2002. ACM. ISBN 1-58113-507-6. doi: 10.1145/543613.543644. URL http://doi.acm.org/10.1145/543613.543644.

Rita Yi Man Li, Herru Ching Yu Li, Cho Kei Mak, and Tony Beiqi Tang. Sustainable smart home and home automation: Big data analytics approach. 2016.

Qiong Liu and Ying Wu. *Supervised Learning*, pages 3243–3245. Springer US, Boston, MA, 2012. ISBN 978-1-4419-1428-6. doi: 10.1007/978-1-4419-1428-6_451. URL https://doi.org/10.1007/978-1-4419-1428-6_451.

Pattie Maes and Robyn Kozierok. Learning interface agents. In *AAAI*, volume 93, pages 459–465, 1993.

Volker Margner and Haikal El Abed. Icdar 2011-arabic handwriting recognition competition. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 1444–1448. IEEE, 2011.

Daniele Miorandi, Sabrina Sicari, Francesco De Pellegrini, and Imrich Chlamtac. Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, 10(7):1497 – 1516, 2012. ISSN 1570-8705. doi: https://doi.org/10.1016/j.adhoc.2012.02.016. URL http://www.sciencedirect.com/science/article/pii/S1570870512000674.

Jose Mirra, Fábio Silva, and Cesar Analide. Reinforcement learning based approach for smart homes. In *Intelligent Environments 2018 - Workshop Proceedings of the 14th International Conference on Intelligent Environments, Rome, Italy, 25-28 June 2018*, pages 38–47, 2018. doi: 10.3233/978-1-61499-874-7-38. URL https://doi.org/10.3233/978-1-61499-874-7-38.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Payam Mokhtarian. From data processing until model selection in machine learning. 2016.

Steven Ovadia. Automate the internet with "if this then that"(ifttt). *Behavioral & social sciences librarian*, 33(4):208–211, 2014.

Themis Panayiotopoulos and Nick Z. Zacharis. *Machine Learning and Intelligent Agents*, pages 281–285. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001. ISBN 978-3-540-44673-6. doi: 10.1007/3-540-44673-7_16. URL https://doi.org/10.1007/3-540-44673-7_16.

Fu Qi-ming, Liu Quan, Cui Zhi-ming, and Fu Yu-chen. A reinforcement learning algorithm based on minimum state method and average reward. In *Computer Science and Information Engineering, 2009 WRI World Congress on*, volume 5, pages 534–538. IEEE, 2009.

Sajid Qureshi. The hidden benefits of installing smart home automation. 2017. Accessed: 2018-01-04.

Erhard Rahm and Hong Hai Do. Data cleaning: Problems and current approaches. *IEEE Data Eng. Bull.*, 23(4):3–13, 2000.

RapidMiner. Rapidminer. https://rapidminer.com/products/. Accessed: 2018-01-06.

Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, second edition, 2003. ISBN 0137903952.

D. Rye. My life at x10. *AV and Automation Industry eMagazine*, Oct 1999.

ENANCER Eletrónica S.A. Ftc232 bus specification, june 2016.

Vishal Sharma, Hrishikesh Thakur, Pankaj Gaud, Hrishikesh Yadav, and Mahavir Devmane. Iot enabled smart-home. 2017.

Stefan Soucek, Gerhard Russ, and Clara Tamarit. *The smart kitchen project-an application of fieldbus technology to domotics*. Citeseer, 2000.

ITU Strategy and Policy Unit. Itu internet reports 2005: The internet of things. *Geneva: International Telecommunication Union (ITU)*, 2005.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

TensorFlow. Tensorflow documentation. https://www.tensorflow.org. Accessed: 2018-01-06.

Muhammad Habib ur Rehman, Chee Sun Liew, Assad Abbas, Prem Prakash Jayaraman, Teh Ying Wah, and Samee U. Khan. Big data reduction methods: A survey. *Data Science and Engineering*, 1(4):265–284, Dec 2016. ISSN 2364-1541. doi: 10.1007/s41019-016-0022-0. URL https://doi.org/10.1007/s41019-016-0022-0.

Werner Weber, Jan Rabaey, and Emile HL Aarts. *Ambient intelligence*. Springer Science & Business Media, 2005.

Dominic West. Amazon echo: 2017 edition-user guide and manual-learn it live it love it. 2016.

Feng Xia, Laurence T Yang, Lizhe Wang, and Alexey Vinel. Internet of things. *International Journal of Communication Systems*, 25(9):1101, 2012.

G Michael Youngblood, Edwin O Heierman, Lawrence B Holder, and Diane J Cook. Automation intelligence for the smart environment. In *International Joint Conference On Artificial Intelligence*, volume 19, page 1513. LAWRENCE ERLBAUM ASSOCIATES LTD, 2005.