A Context-Aware Personalized Recommender System for Automation in IoT based Smart Home Environment

SAURABH GUPTA

An applied project submitted in partial fulfilment of the requirements for the degree of

Master of Science in Data Analytics

at Dublin Business School

Supervisor: Ms. Terri Hoare

August, 2020

DECLARATION

I, the undersigned, hereby declare that this applied project, which I have submitted to Dublin Business School for the award of degree of MSc Data Analytics, is solely my own work and is a result of my own investigations written in my words, except where otherwise stated and acknowledged clearly by references. Furthermore, this work has not been submitted, in part or in whole, for the purpose of obtaining any other academic credit/grades or degree.

Signed: SAURABH GUPTA
Student Number: 10519384
Date: August 25th, 2020

ACKNOWLEDGEMENT

At the very outset of this report, I would like to extend my sincere and heartfelt gratitude to Ms. Terri Hoare, my supervisor for this research work and professor for other academic modules, for her invaluable support throughout the planning as well as the development phase of this project. I'm ineffably indebted to her for the conscientious guidance, the willingness to give her time generously, and the unceasing encouragement to accomplish this project.

I would also take a moment to thank DBS for providing access, through their online library and web services, to a vast number of books, literatures, journals and other resources, which has been pivotal for the accomplishment of this work.

I also acknowledge with a deep sense of reverence, my gratitude towards my parents, family members and friends, who have been there as a moral support during these trying times of COVID-19 pandemic, and helped me keep motivated and focused towards this research work, throughout the period.

Thank you,
Saurabh Gupta

ABSTRACT

An IoT enabled smart-home fulfils many needs of the inhabitants such as providing conveniences through personalised home automation, saving energy through optimizations, etc., based mostly on the user's behaviour and interaction with these devices. This research proposes a methodology to build a system that learns user's habits/preferences in a given situation, merges them with contextual information and finally recommends relevant automation services, which the user would like to perform at that instance. The user preferences are learned using unsupervised algorithm such as association analysis, trained upon inhabitant's prior interactions captured passively through the IoT enabled sensors while they live their daily lives. The contextual information, such as location, time, etc. are extracted from the given data. The preferences and the contextual information are fed to supervised learning algorithms to predict desired user action on the basis of current sensor outputs and the contextual setting at that moment.

Keywords: Recommender System, IoT, Context-Aware, Smart Home Automation, Personalization, Machine Learning, Behavioural Analytics

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AI – *Artificial Intelligence*

ANN – *Artificial Neural Network*

CRISP-DM – *Cross-Industry Standard Process for Data Mining*

CSV- *Comma Separated Values*

EDA – *Exploratory data Analysis*

GBT – *Gradient Boosting Tree*

GLM – *Generalized Linear Model*

GUI – *Graphical User Interface*

IoT – *Internet of Things*

ML – *Machine Learning*

Chapter 1 - Introduction

1.1 Background

Perceiving technology as the predominant means to attain better quality of life has been the intrinsic nature of humans since the inception of humanity, be it the invention of the wheel during the caveman period or the internet during the recent decades. And today, we already are witnessing the era of digital transition where everything around us, from the toothbrush to the television, is equipped with internet capabilities -generating, storing and transmitting myriads of data. These devices have been coined with the term '*Internet of Things*', or basically any device that connects to the internet (Marr, 2015).

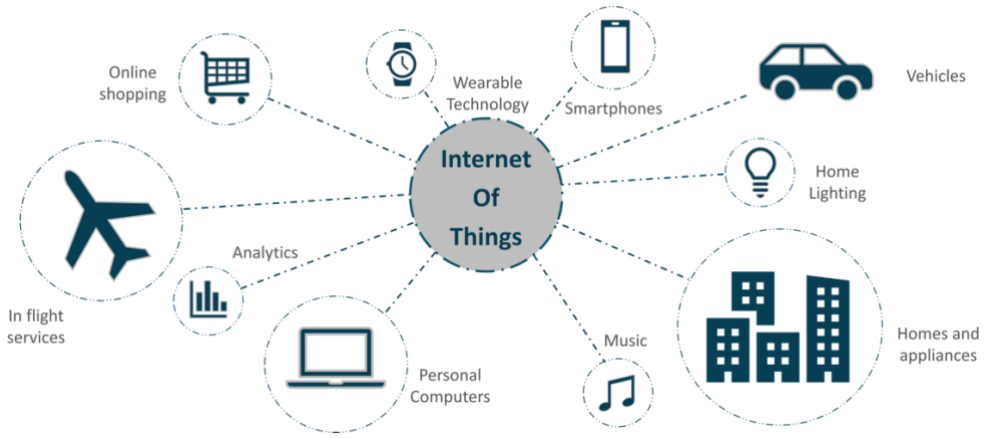


Figure 1.1 *Application of IoT in Different Domains of the Real World*

The advent of IoT has been an outcome of the proliferating internet users and the advancing technologies, and the consumers are elated with the ease these devices have to offer in daily life, keeping them longing for more and more of these conveniences. And it is no more just about the ease, the businesses are taking it up a notch and are transforming these conveniences into personalised experiences, whether it is the personalised playlist you love that *Spotify*[1] offers or your personalised coffee which *Nespresso*[2] blends as per your preferences. According to a research, there has been significant positive impact in customer satisfaction and brand loyalty when personalized services are incorporated (Tong, et al., 2012). The consumers are loving it, the brands are delivering and leveraging it. *Netflix*[3] says

[1] A music streaming and media services provider https://en.wikipedia.org/wiki/Spotify
[2] A brand delivering coffee products and coffee machines https://en.wikipedia.org/wiki/Nespresso
[3] A streaming service broadcasting variety of TV shows, movies, etc. https://en.wikipedia.org/wiki/Netflix

that “Personalization is one of the pillars of Netflix because it allows each member to have a different view of our content that adapts to their interests and can help expand their interests over time” (Netflix, 2020).

" We see our customers as invited guests to a party, and we are the hosts. It's our job every day to make every important aspect of the customer experience a little bit better ” – Jeff Bezos, CEO, Amazon (Forbes, 2013)

Among the application of IoT in various domains, the use of IoT in Smart Home stands out to be the most popular among all (Lueth, 2015), and has occupied a fair share of market where the tech giants like *Google, Amazon* and *Apple* are competing each other to please their consumers. The homeowners embrace this technology for simple reasons such as saving energy and money through optimization of consumption; making life simpler by automating mundane household tasks as well as controlling appliances remotely; making the home more secure by using connected camera and intrusion alert systems; and lastly entertaining them with the very popular media devices such as smart speakers, which also acts as a control centre for many other IoT devices. User interaction with these devices generates data, such as usage patterns, can reveal interesting information about the users such as their preferences, interests, routines etc which could be utilized to deliver real-time personalised experiences, making the inhabitant’s day to day life better.

A smart home environment typically consists of multiple units that are IoT enabled-composed mainly of sensors, smart devices and actuators – which gathers data from the surroundings as well as the user interactions with these devices, and transmits them over the wireless network for processing and other applications, analysis being one of the critical

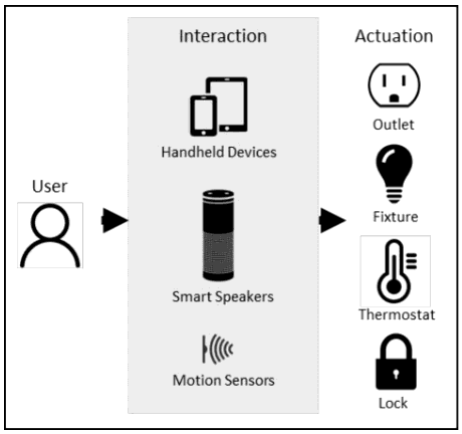


Figure 1.2 *A typical Smart Home Ecosystem and its Component*

among these. A typical smart home environment can be shown in the *Fig 1.3* as depicted by (Louis & Rashid, 2018).

Another key element which add ups and builds the foundation of most of the larger smart systems is the sensors. Whether it's the industries, smart cities, or the wearable devices, they all are equipped with arrays of this wireless rudimentary elements that senses a variety of physical attributes such as heat, proximity, pressure, speed orientation etc. and transmits over the internet to a computing resource for further processing. These form the physical/perception layers of an IoT system which gathers data from real world and relays it through different layers, depending on the system architecture, for further processing and making actionable sense out of these data. A typical end-to-end IoT architecture simplified down to three layers is shown in *Fig 1.2*, which shows how that data is transmitted to and fro the actual device and main computing unit for analysis.

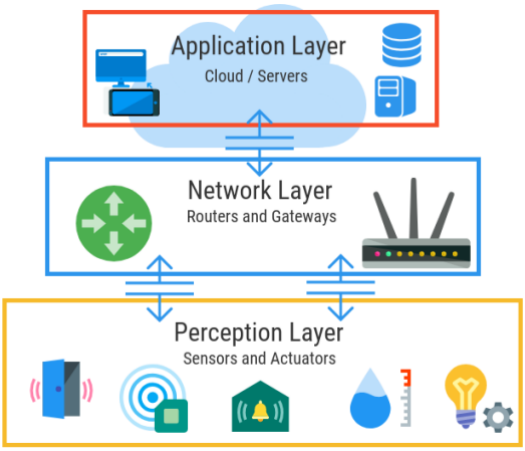


Figure 1.3 *A fundamental three layer architecture of an IoT system*

1.2 Overview

Both the consumers as well as the businesses in today's world are striving to emulate the recent technological advancements, among which IoT is the latest. And smart home is one of the dominant application of these inter-connected wireless devices. The smart homes of the future intend to heighten comfort, assist older and incapacitated individuals, and facilitate inhabitants to conserve vitalities like energy (Rasch, 2014). The primary purpose of this study is to build a predictive model that suggests the inhabitant of the Smart Homes with useful recommendations for automation services, which the user would want to perform anyway, for relevant smart home devices by sensing the present state of the user and the

contextual set up at that moment. This is proposed to be achieved by learning useful habits or preferences of the inhabitant, based on his/her prior interactions with the array of devices installed, adding contextual meaning to these behaviours and finally recommending automation services in future instances when the current behaviour of the user as well as the current context matches from the established rules. This recommender system could then be implemented to highlight these interesting choices on a central interface like a smart wearable device etc. where the user could actuate these desired actions through voice, touch, or gestures, making the lives more convenient around the smart home ecosystem.

For example, assume that an inhabitant just after having dinner arrives in the living room and sits on the couch. Fairly many services could be relevant in this situation, depending on the context as well as on the user's habit. If the user typically watches TV after dinner when he is in the living room on the couch, then recommending a service to turn on the TV would be useful here. Or if the user likes just to relax on the couch and usually dims the light and open the windows, then a recommendation of opening the window and dimming the lights would be relevant here. These are the user's habits which could be learned from previous behaviours. But blindly learning and recommending the user habits without any contextual information would not suffice. Imagine if it's an unusual cold day, recommending to open the windows won't be a good idea as the user typically does that on a comparatively warmer night. Or instead of watching TV, the user tunes in into the radio on a weekend night. So, the contextual information also matters.

The contextual attributes may refer to relevant elements in connection to which the behavioural usage information were found (Verkasalo, 2015). The user context may contain personal information about the user such as location, or heart rate etc., or information about user's surrounding environment such as temperature (Rasch, 2014). The contextual information may also contain knowledge about the time of the day, day of the week, special occasions like a holiday as well. The contextual knowledge is an added intelligence to the technical or the raw data from the sensors (Verkasalo, 2015), which elevates the overall logical sense that data makes altogether. Therefore in this study, contextual information has been derived from the available data and made use for better accuracy of the recommendations.

Why a Recommender System, not an autonomous Automation System?

A question that may come to mind would be why to make recommendation system rather than a system that actually implements the recommendation directly. This research work proposes to build a recommender system for the following rationales:

- Inflexibility: To anticipate human actions and desire in any situation beyond a certain extent is unrealistic, at least for now. A system that actually implements the desired control may turn out to be rigid for some users and may annoy them. A participant in a study conducted by (Brush, et al., 2011) mentioned how he retracted back from the idea of waking up to music and lights turning on automatically, because his life was not as structured as this arrangement to be established as a routine. He also said scheduling or implementing the automation as a routine was not something that he desired.
- Concern of losing control: It should also be kept in mind that all inhabitants may not fancy the possibility that the smart system in their home self-sufficiently plays out some activity on their behalf. In (Eggen, et al., 2003) the author stresses that users desire control over how, when and to what extent the system takes over their house.

Therefore, instead of an autonomous system, a system that recommends services to the inhabitants with a certain degree of confidence and probability is proposed for this system, in order address the potential issues of control and inflexibility, and make the model pragmatic and efficient.

1.3 Research Problem

The predicament associated with this research work is to coalesce the generated rules and their respective contexts to a suitable form that could be fed into a predictive model. The primary goal of this work has been centered around to explore and design the methodologies

and algorithms to learn frequently occurring meaningful rules or habits that the inhabitants typically performs in a given set up, and then extract and combine the contextual information

associated with these rules, and finally compare supervised machine learning algorithms that trains a model that anticipates and recommends services to the inhabitants.

RESEARCH QUESTION: "How can historical user-interaction data, which is gathered from multiple devices within an IoT-based smart home ecosystem, be utilized to discover user-preferences under various contexts to build a predictive model that recommends services to automate those preferred actions in the given contextual setting?"

OBJECTIVE: To build a model that extracts useful user-preferences and delivers relevant context-aware recommendations.

HYPOTHESIS: The user-preferences could be mined as association-rules from frequently occurring events, using unsupervised machine learning algorithms like association analysis. The recommendations could be delivered using various predictive supervised learning models after combining the user-preferences with their associated contexts.

1.4 Scope

Reckoning that this research is one of the initial attempts to build such a system, this study focusses on exploring ways to build such a model and lay an analytical groundwork to build more sophisticated models in further studies.

1.5 Limitations

This research work is currently capable of learning user-preferences in a single-occupant home only. Also, the amount of data and the number of contexts extracted here are limited due to availability of data because of privacy concerns.

1.6 Dissertation Layout

The research work is organised as follows: Section 2 reviews previous academic works and literatures in the related field of study, followed by Section 3 where the research methodology and the proposed plan for the research is discussed. Section 4 is a mentions in detail a step-by-step approach for the preparation and the analytical modelling to build the system. The results of Section 4 are evaluated in Section 5, followed by exploratory discussion around the findings. The study is concluded in Section 6 by answering the framed research and business problems, summarizing the entire study while suggesting future possibilities to advance the laid groundwork in this study.

Chapter 2 – Literature Review

This research work intents to find out ways to utilise user behaviour data in a smart home to enable personalised experiences for inhabitants in form of recommendations for automation. Exploring various machine learning algorithms, both supervised and unsupervised, for the intended results would be a vague effort without having gaining knowledge from prior works in the related field. Therefore, to comprehend the concepts and models favourable for the current study, relevant research papers and journals were referred which will be evaluated in the following section.

The work done by (Rasch, 2014) identifies the key issues faced by inhabitants of the smart homes -mainly complex interfaces, configuration issues and inhabitants apprehensions- and address it by proposing a flexible recommender system that highlights the most interesting choices of automation available for the user in the current situation, simplifying the entire process. The author here emphasises that the recommender system should be context aware, meaning it must be aware of the requirements of the inhabitant in the current situation. The author also asserts that automation of routine work is not desirable by the inhabitants. This is achieved by learning the user preferences in the training phase where the temporal relations of the preferences are also preserved. In the next phase, the systems takes the current inputs from the sensor, deriving the contextual information, which is then utilised to recommend automation services by predicting the behaviour of the inhabitant. The author uses a very distinctive approach of extracting the preferences by utilising an observational tuple which accounts the frequency of a unique event or context, and by using an interval look up for temporal relations which accounts for maximal time elapsed for a given context and ignores the temporal information where the time period is extremely large. For predicting the change in context and recommending the services, the author uses a generalized version of Bayesian probability (Naïve Bayes), called Dempster-Shafer theory which achieves a consensus among many distinctive data sources rather than one. The final result, which is a ranked list of recommendation based on its probability, of the proposed method is compared with other algorithms such as Naïve Bayes and Random Forest and turns to give a better recall and precision in all cases. The paper lays out an intricate yet efficient way of delivering a

context aware recommender system while highlighting on the use and the processing of temporal information.

(Belghini, et al., May 2016) in a theoretical only work, lay out a pragmatic framework to handle the contextual information extracted from the physical sensors and combine it with the mined historic user preferences to predict the user action within current situation. The authors recommend to use unsupervised learning such as association analysis to mine the rules from a frequent item sets by setting a minimum frequency threshold on the recurring items found. After combining these rules with their relevant contextual information, the decision making process is recommended to be solved by a classification problem by using a three layered ANN (Artificial Neural Network) employing back propagation algorithm, where the input is the contextual parameters, the output is the next action desired by user, and a hidden layer for computational purposes. Even though the work would have been more sufficing if the ways to combine the contextual data with rules have been elaborate, this paper builds a subtle bodywork for a context aware system.

As explained in (Verkasalo, 2015), the author presents patented invention that lays out a physical architecture and methodology for behavioural and contextual data analytics in a wireless network of devices. The paper defines the input logic for acquiring the observational or user-interaction data from these devices that could be further utilised for mining key usage patterns within a given context. The paper emphasizes on deriving at least on contextual dimension related to the device in use to enrich the raw data available or make the system more intelligent. These derived contextual information is said to be utilised by the data mining engine for cluster/association analysis, correlations, and pattern recognition to facilitate the system to adapt its behaviour in accordance to these subtle intelligence.

(Engelmann, et al., 2016) presents a corpus of a real dataset that enables researchers to make advancements in learning behavioural rules from the user interactions in a smart home environment. The work imparts some knowledge of the available sensors types, the pre-processing techniques, and suggests unsupervised learning methods to discover association between sensor events to form rules that could be implemented in the proposed recommender system. The work also suggests incorporation of semantic or logical

information, probably annotated manually in the initial phase, so as to amplify the quality of the desired outputs.

(Alshammari, et al., 2018) provides a dataset that mimics the activities of inhabitants in smart-home environment by using a simulator named Open Smart House Simulator that circumvents the privacy issues related to data collection of sensitive information made publicly available, and provides a rich dataset gathering information from around 20 sensors along with activities of the inhabitants annotated. This dataset has been made use of in the current research work. This work also helps in gaining a background knowledge of understanding the nature of dataset and its distribution, typically available for the real businesses to enhance the services by giving a personalised experience to the consumers.

The works mentioned above and others contributes to a fair comprehension of the background and the techniques necessary for the current research work to proceed, although no complete working system or prototype of the proposed system has been perceived in any of the literatures as current work is novel in its modus operandi. A significant thing to notice from the literatures reviewed was the emphasis on the usage of contextual information for added intelligence to understand and predict the user actions in a more efficient way. The use of unsupervised learnings has also been underscored to learn the relevant rules from the historic data available, and therefore it has been considered for the proposed system.

Chapter 3 – Research Approach and Project Roadmap

Incidentally, even the best and most talented minds among all, can sometime succumb to petty negligence of something as simple as structural organisation of work, particularly when taking a shot at complex activities like a research. Therefore, observing and adhering to an industry-identified approach can make the tasks easier and attain better results. In the following section a well-known cross industry approach, to be applied in this research work, for data mining processes is discussed. Also, a tailored step by step strategic plan for developing the proposed model is laid out, while keeping in mind the processes of research methodology adopted.

3.1 Research Approach: CRISP-DM Model

Cross-Industry Standard Process for Data Mining, CRISP-DM, is a recognised model that describes the structured procedures and approaches to plan a data mining project. It is a robust yet flexible approach that helps solve complex data analytics problems. This model lays an idealised course of actions to be performed, while also offers flexibility to reposition these steps if necessary in real practise and also recommends to go back and forth or repeat certain processes wherever required. A general flow of process prescribed by CRISP-DM model is shown in *Fig 3.1* .

Figure 3.1 *Steps in CRISP-DM Methodology*

The circular blocks in *Fig 3.1* the represent the steps involved in the process, and the arrows show the flows, the inter-dependencies and the backtracks wherever applicable within the processes. The micro-level activities involved in each of the six-step process of CRISP-DM methodology is summarised in the *Table 3.1* below.

Business Understanding	Identifying the business problem, gathering the requirements, assessing the risk & contingencies, and finally setting the success criteria in accordance to these
Data Understanding	Creating a data description report, exploring the data and verifying the data quality for later implementations
Data Preparation	Creating rationale for inclusion or exclusion of attributes/subset of data, deriving attributes, transformation, integration and aggregation of data in required formats
Modelling	Selecting appropriate algorithms, creating test/train records where applicable, finetuning the parameters, and executing the model
Evaluation	Formulating key evaluation parameters, assessing the results of the model against desired parameters, comparing the efficiency of different algorithms, reviewing the results and stating the decisions along with the rationale
Deployment	Planning the deployment strategy with steps involved, and building a monitoring and maintenance plans

Table 3.1 *Activities involved in each of the six CRISP-DM steps*

3.2 Project Roadmap

Keeping in mind the CRISP-DM methodology, a strategic roadmap for implementing the dissertation has been devised. As the CRISP-DM offers sufficient flexibility to manoeuvre through the variations of different projects and it's practical implementation, few changes has been made in this plan as well which is worth mentioning. The Fig 3.2 outlines a roadmap for carrying out the steps involved to achieve the intended goal of this research.

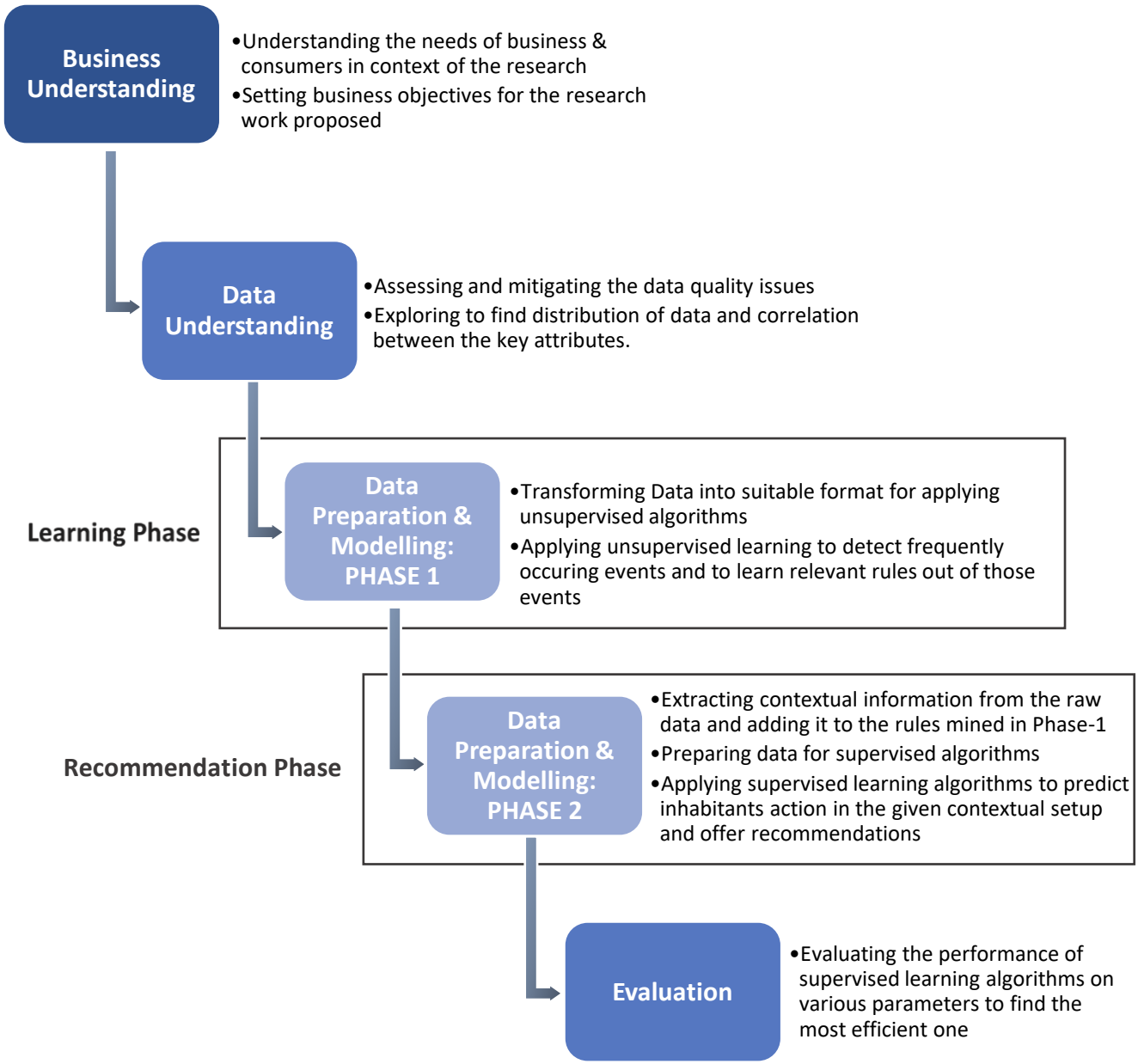


Figure 3.2 *Flow of research process using CRISP-DM methodology*

As seen in *Fig 3.1*, the data preparation step and the modelling step as described in the CRISP-DM schematic is merged and modified into two phases, as required by the proposed work. The process employs machine learning algorithms twice, first in unsupervised *Learning Phase* to discover the rules, and second in supervised *Recommendation Phase* to predict the user action. Both of these require their own preparation, and therefore has been modified so. Also, the deployment phase is not in scope of this research work, and therefore has been skipped.

Chapter 4 – Methodology

The proceedings of the research was conducted using the CRISP-DM methodology as mentioned in *Chapter 3* of this document. This methodology provides a guided approach to accomplish the research work. The steps involved will be elaborated in details, with respect to the proposed methodology, in this section.

4.1 Business Understanding

The smart homes of today are equipped with numerous devices and technologies that facilitate the occupants with a comfortable living environment by providing options such as automation, security, etc. (Belghini, et al., May 2016). However, sometimes the interfaces are a bit complex and the routines have to be configured into automation manually, which is not congenial in most cases. Also, the routines are more than often sub-conscious in nature and cannot be catalogued or depicted in detail to encompass all the variation (Rasch, 2014). "In order for smart homes to achieve their promise of significantly improving the lives of families through socially appropriate and timely assistance, they will need to sense, anticipate and respond to activities in the home", stated (Davidoff, et al., 2006).

The businesses that offer such systems constantly receive criticism from the consumers around grounds such as accuracy or rationality of autonomous activities carried out by systems, inflexibility of controls and adaptation to user preferences. In the current era of a consumer-centric economy, addressing these issues relentlessly have become a challenge, as well as a way, for businesses to stay strong in competitive market. In order to confront these issues, learning through the past activities of the inhabitants to offer suggestions to automate services in a given situation or context, seems a fair idea to tackle these problems collectively. Therefore, a recommender system that ensures learning not only on the basis of incidence of past frequent events, but also on the contextual setup of the occurrence of these events, meet the needs of the businesses developing such interconnected smart home systems.

The business objective of the proposed system would be to present a list of context appropriate personalised automation suggestions at the consumer's disposal, at a convenient interface such as the display of a wearable technology or smart phone that could be actuated

by a touch or through voice commands. Building such a systems requires a corpus of user-interaction data collected over certain amount of time, ideally few weeks, from an array of connected devices or sensors installed in a home, having temporal information and at least one given or derived contextual attribute. Using appropriate machine learning algorithms and data mining principles, the dataset would be utilised to learn the rules and then apply those rules to offer recommendations to user in any future contextual situation.

4.2 Data Understanding

As the data required for developing the proposed system contains sensitive information and falls under Data Privacy laws around various regions of the world, dissemination of such data publicly, which is collected from real users, is prohibited. Therefore, the datasets available publicly are mostly simulated efforts of researchers in this field, collecting or generating data in a real or virtual environment, and hence such datasets are limited in numbers. Finding the right dataset for this work has been a challenge because of the prerequisites of the structure and dimensions of the data. The dataset[4] chosen for this research has been provided by (Alshammari, et al., 2018) in their attempt to create a realistic dataset using simulation tools to facilitate research in smart home design utilising emerging machine learning techniques.

The dataset has been generated by creating a virtual home where real participants control and navigate the a game-like avatar in first person view to performs ADLs (activities of daily life). Through this period of ADLs, the simulation tools records the state of sensors installed in various parts of the house. The smart home consists of a bedroom, kitchen, living, bathroom, office and hallway passage connecting all as shown in *Fig 4.1*. The actions like turning on the television can be performed by a using text dialogue as shown in *Fig 4.2.* The dataset used consists of data equivalent to simulated activities recorded through the period of two months of February to March 2016.

[4] Dataset: https://openshs.github.io/datasets/

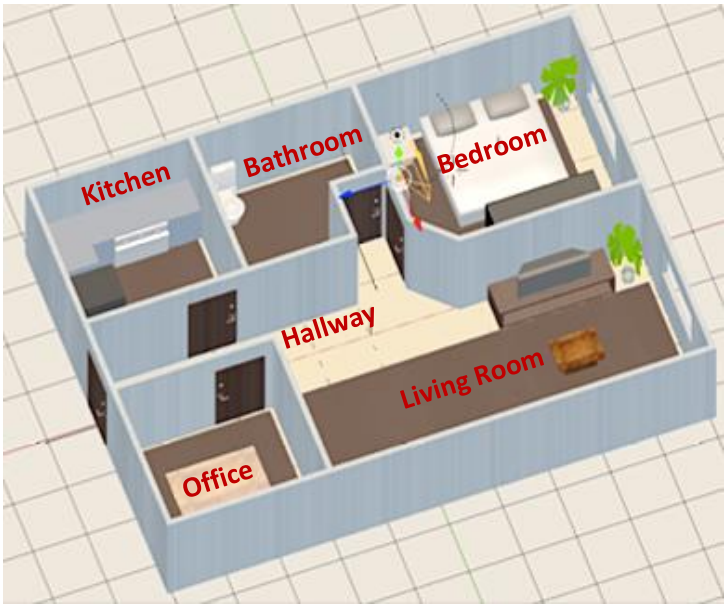


Figure 4.1 *Smart-Home Design*

Figure 4.2 *Dialogue Box to select activities*

The smart home is equipped with 29 binary sensors positioned at various objects such as doors, carpets, TV, lights, cupboards, carpets etc. which tells the state of the object on which is installed. For example, a carpet sensor in the living with state (1): 'active' means that someone is walking/standing on the carpet of the living room, or the light sensor in the kitchen with state (0): 'inactive' means that the light is turned off in the kitchen at that moment. In addition to this, the dataset has a *timestamp* attribute which tells the precise date-time of the moment, where the sensor reading are recorded at a time interval of one second in the virtual environment, when the simulation is activated. The dataset also has an annotated *Activity* column which determines the classifies the activity of the participant into

sleep, work, personal, eat, leisure, other and an additional label *anomaly* which annotates difficult to classify state of the participant. The list of all sensors and other features is shown in *Table 4.1*

#	Name	Type	Description
1	bathroomCarp	binary	Bathroom carpet sensor
2	bathroomDoor	binary	Bathroom door sensor
3	bathroomDoorLock	binary	Bathroom door lock sensor
4	bathroomLight	binary	Bathroom ceiling light
5	bed	binary	Bed contact sensor
6	bedTableLamp	binary	Bedroom table lamp
7	bedroomCarp	binary	Bedroom carpet sensor
8	bedroomDoor	binary	Bedroom door sensor
9	bedroomDoorLock	binary	Bedroom door lock sensor
10	bedroomLight	binary	Bedroom ceiling light
11	couch	binary	Living room couch
12	fridge	binary	Kitchen fridge
13	hallwayLight	binary	Hallway ceiling light
14	kitchenCarp	binary	Kitchen carpet sensor
15	kitchenDoor	binary	Kitchen door sensor
16	kitchenDoorLock	binary	Kitchen door lock sensor
17	kitchenLight	binary	Kitchen ceiling light
18	livingCarp	binary	Living room carpet sensor
19	livingLight	binary	Living room ceiling light
20	mainDoor	binary	Main door sensor
21	mainDoorLock	binary	Main door lock sensor
22	office	binary	Office room desk sensor
23	officeCarp	binary	Office room carpet sensor
24	officeDoor	binary	Office door sensor
25	officeDoorLock	binary	Office door lock sensor
26	officeLight	binary	Office ceiling light
27	oven	binary	Kitchen oven sensor
28	tv	binary	Living room TV sensor
29	wardrobe	binary	Bedroom wardrobe sensor
30	Activity	String	The current participant activity
31	timestamp	String	The timestamp every second

Table 4.1 *List of sensors and other attributes in the raw dataset*

The raw data when loaded in python appears as shown *Table 4.2* with a total record of 100974 rows in 31 columns.

	wardrobe	tv	oven	officeLight	officeDoorLock	officeDoor	officeCarp	office	mainDoorLock	mainDoor	...	bed	bathroomLight	bathroomDoorLock	bathroomDoor	bathroomCarp	Activity	timestamp
0	0	0	0	0	0	0	0	0	1	0	...	1	0	0	0	0	sleep	2016-02-01 08:04:10
1	0	0	0	0	0	0	0	0	1	0	...	1	0	0	0	0	sleep	2016-02-01 08:04:11
2	0	0	0	0	0	0	0	0	1	0	...	0	0	0	0	0	sleep	2016-02-01 08:04:12
3	0	0	0	0	0	0	0	0	1	0	...	0	0	0	0	0	sleep	2016-02-01 08:04:13
4	0	0	0	0	0	0	0	0	1	0	...	0	0	0	0	0	sleep	2016-02-01 08:04:14
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
100969	0	0	0	0	0	0	0	0	1	0	...	1	0	0	0	0	sleep	2016-03-03 18:18:14
100970	0	0	0	0	0	0	0	0	1	0	...	1	0	0	0	0	sleep	2016-03-03 18:18:15
100971	0	0	0	0	0	0	0	0	1	0	...	1	0	0	0	0	sleep	2016-03-03 18:18:16
100972	0	0	0	0	0	0	0	0	1	0	...	1	0	0	0	0	sleep	2016-03-03 18:18:17
100973	0	0	0	0	0	0	0	0	1	0	...	1	0	0	0	0	sleep	2016-03-03 18:18:18

Figure 4.3 *Raw dataset as seen in Python*

Exploratory data analysis is a key part of understanding the data and distribution of variables and attributes. An exploratory visualization analysis of few of the attributes, to see its distribution is shown in Fig 4.4 for *Activity* attribute, which shows the highest count for 'personal' activity and lowest for 'anomaly'.

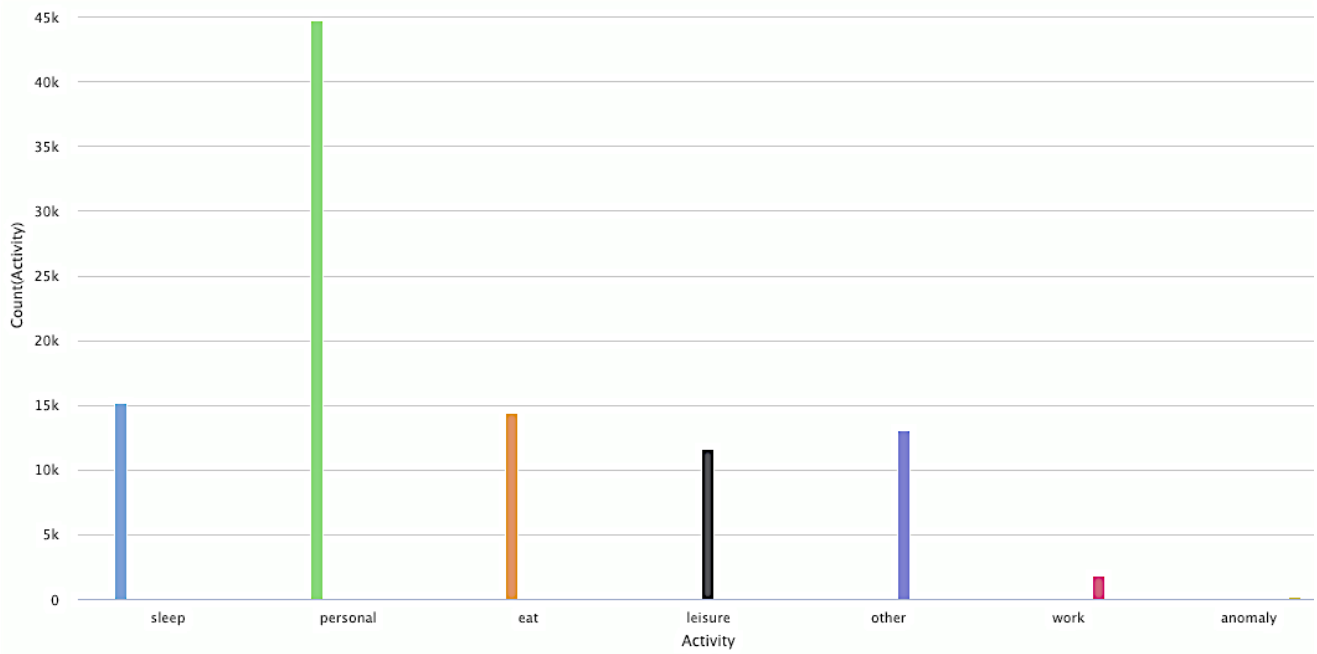


Figure 4.4 *Visualization for count of 'Activity' labels*

Similarly, visualization for distribution of *Timestamp* feature is also visualised and it's noticed that the activities recorded at the first week of the second month has the highest frequency.

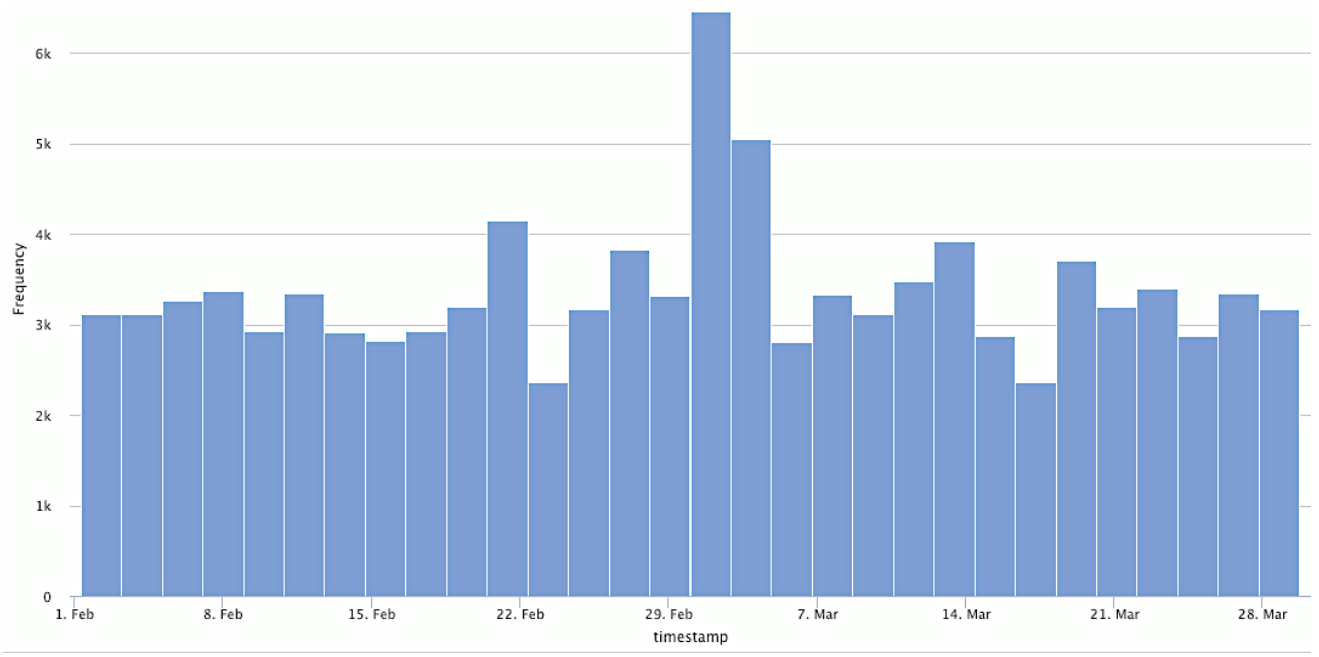


Figure 4.5 *Visualization for distribution of 'Timestamp' attribute*

As the other features have binary outputs, therefore such distribution is not relevant. Inter-relation between variables can be seen in a correlation heatmap, but in this dataset, correlation is not relevant as in some cases, the relation between sensors are obvious. For example, when the office desk sensor is in 'active' state, then office carpet sensor would also be in 'active' state due to the presence of participant over the carpet which covers the entire office. The reason for such exploratory exercise is to find out any interesting relations, irregularity, or patterns within the data before further processing. As the data appeared fine through this first look, the study was proceeded further to the next phase, where the data pre-processing and modelling is done back and forth into two phases. Phase-1 for learning the frequent events and association rules of those events, and Phase-2 for extracting the contextual information and adding them to rules for recommending services.

4.3 Data Preparation & Modelling: Phase-1 (Learning Phase)

In this section, the development and findings of the Phase-1, i.e. learning of frequent events and association rules, are discussed in detail. The necessary pre-processing required and the model applied will be explored gradually. The tool used in this section for the mentioned steps was *Python* and the output code is available in *Jupyter Notebook* (.ipynb) format.

4.3.1 DATA PRE-PROCESSING: Phase-1

Once the data is loaded into the *Python* and a preliminary outlook of raw data through exploratory analysis is done, the data needs to be formatted for consistency and relevance, and an appropriate subset is to be prepared as per the configuration required by the model to be applied. The steps conducted are as follows:

- Convert the raw data, which was in *CSV* format, into a *pandas DataFrame* format as it's the recommended one because of its ease of data manipulation methods available, processing speed, memory consumption.

```
import pandas as pd
rawdata = pd.read_csv("smart_home_data.csv")
```

Code Snippet: *Converting CSV to DataFrame format in Python*

- Convert the temporal data in the *Timestamp* column to date-time format

```
#Coverting datatype of 'timestamp' column into datetime format
rawdata['timestamp'] = pd.to_datetime(rawdata['timestamp'],infer_datetime_format=True)
timestamp           datetime64[ns]
```

Code Snippet: *Converting timestamp feature to date-time format in Python*

- As the *Apriori* algorithm requires the items, or events in this case, to be *One Hot Encoded*[5] transaction format i.e. converting nominal categorical data into features with binary value where 1 represents the presence of an item and 0 represents absence of the item as shown in *Fig 4.6*. This format is often used in ML to indicate the state of a machine or device.

Note: *In this study, the presence of an event suggests that a sensor is 'active or state 1' and absence suggests that a sensor is 'inactive or state 0' during a given time*

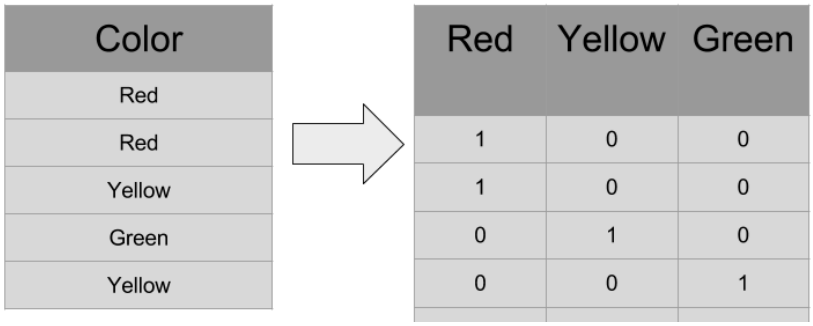


Figure 4.6 *One-Hot Encoding*

The dataset used here was already in this format, so it did not require any transformation, but is an essential step for all Apriori algorithm implementations.

- Create a subset of the dataset on which the model is to be applied. Here, we need all the sensor data columns, which exists in the first 29 columns. Therefore, the *timestamp* and the *activity* columns can be dropped and rest can selected to create a subset for mining the frequent events in the next step.

```
newdata=rawdata.iloc[:,:29]
newdata
```

Code Snippet: *Creating a subset of data with all sensor data columns, excluding the activity and timestamp columns*

[5] https://stackabuse.com/one-hot-encoding/

4.3.2 MODELLING: Phase-1

Once the required format of the data is achieved through the pre-processing steps, the modelling phase can be executed. In this learning phase, the frequent events or preferences of the user are discovered, and the association rules are learned using an association analysis algorithm. This section covers the rationale for selection of algorithm including its theory, and its implementation in *Python* for associating the co-occurrence of events and deriving rules from it for the recommendation engine.

APRIORI Algorithm

The Apriori is one of the algorithms of association rule analysis. The association rules quantifies the strength of co-occurrence of an item or an event with another item or an event (Kotu & Deshpande, 2014). The input for association rules is a list of past transactions (here user interactions or events) and the output is a rule, configured as tuple of items that co-occur frequently, that says that 'There's an increased likelihood of occurrence of an event B, if the event A had occurred'. All association rule algorithms should be able to find frequently occurring events from the entire set of all possible events (Kotu & Deshpande, 2014). The Apriori algorithm is based on the principle "If an itemset is frequent, then all its subset items will be frequent" as quoted by (Tan, et al., 2005).

Example of Association Rules of activated sensors in a smart-home environment:

{Bedroom Carpet} → {Bedroom Light}
{Oven, Kitchen Cupboard} → {Radio}
{Couch, TV} → {Lights Dim}

Note: Implication only means co-occurrence, not causality!

From the examples above shown above are some frequently co-occurring events. It can be seen that when the bedroom carpet activates, usually the bedroom light is found to be on. Or when the oven is turned on and kitchen cupboards are open, the radio is also found to be on, which implies that the inhabitant prefers to listen to radio while cooking. Or prefers to dim the light when watches TV from the couch. It can be inferred from the frequency of these events that it's the preference of the inhabitant in given context.

Some of the key concepts related with association rule analysis is mentioned below:

- **Itemset:** A collection or list of one or more items or events
  Example- {Oven, Kitchen Cupboard}, {Living Lights}, {Couch, TV}

- **Support:** The ratio of an itemset in total number of transactions. **{**range: [0,1]**}**
  Example- Support ({Couch, TV}) = 2/7, which means couch and TV were found active together in two instances/transactions where total instances were 7

- **Frequent Itemset:** Any itemset which has a *Support* equal to or greater than a set *Minimum Threshold of Support*

- **Association Rule:** An implied expression in the form of **X** → **Y** where X and Y are itemsets
  Example- {Oven, Kitchen Cupboard} → {Radio}
  {Couch, TV} → {Lights Dim}

- **Antecedent & Consequent:** These are set of itemsets found in conjunction of each other in an association rule. In a rule **X** → **Y**, X is antecedent and Y is consequent
  Example- For the rule {Oven, Kitchen Cupboard} → {Radio}, here the itemset {Oven, Kitchen Cupboard} is antecedent and {Radio} is consequent.

- **Confidence:** The measure of likelihood of a how often does all the items in the consequent **Y** of a rule appears in transactions that contains all the itemsets in the antecedent **X.** Confidence gives the reliability of the rule. **{**range: [0,1]**}**

$$\text{Confidence } (X \rightarrow Y) = \frac{\text{Support } (X \cup Y)}{\text{Support } (X)}$$

- **Lift:** The interestingness measure. Confidence ignores frequency of consequent **Y**, this is accounted in lift. Value closer to 1 means that the antecedent and consequent are independent and rule not interesting. Lift values higher than 1 means the rule is interesting. Lift may find very strong associations for less frequent items **{**range:[0,∞]**}**

$$\text{Lift } (X \rightarrow Y) = \frac{\text{Support } (X \cup Y)}{\text{Support } (X) * \text{Support } (Y)}$$

- **Conviction:** Conviction is the ration of the expected frequency of antecedent **X** occurring in spite of the consequent **Y** and the observed frequency of mistaken predictions. A conviction of 1.40 means that the rule **X** → **Y** would be incorrect 40% more often if the relationship among X and Y is purely random. **{**range:[0,∞]**}**

$$\text{Conviction } (X \rightarrow Y) = \frac{1 - \text{Support } (Y)}{1 - \text{Confidence } (X \cup Y)}$$

- **Leverage:** Similar to lift but leverage computes the difference, and tends to prioritize items with higher frequencies/support in the dataset.

$$\text{Lift } (X \rightarrow Y) = \textit{Support } (X \cup Y) - \text{Support } (X) * \text{Support } (Y)$$

Apriori Principle: If any itemset occurs frequently, then all of its subsets must also be frequent and conversely, if any item is infrequent, then all of its supersets must also be in frequent

$\forall A,B : (A \subseteq B) \Rightarrow s(A) \geq s(B)$ - **Support of any itemset shall never be more than the support of its subset**

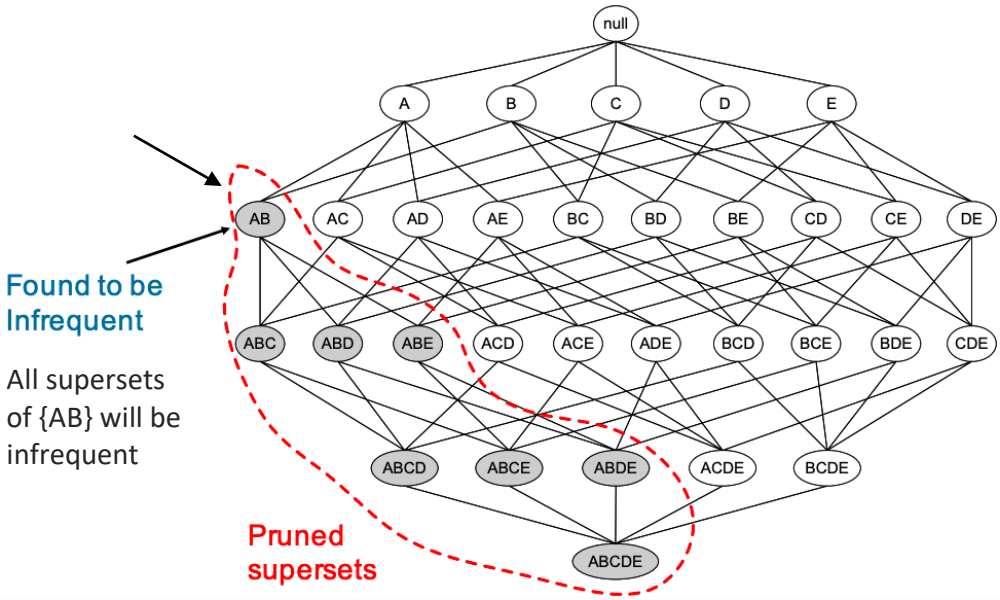


Figure 4.7 Itemset lattice depicting APRIORI principle

Apriori Pseudocode:

– Let k=1

– Generate F1 = {frequent 1-itemsets}

– Repeat until F_k is empty

Candidate Generation: Generate L_k+1 from F_k

Candidate Pruning: Prune candidate itemsets in L_k+1 containing subsets of length k that are infrequent

Support Counting: Count the support of each candidate in L_k+1 by scanning the database

Candidate Elimination: Eliminate candidates in L_k+1 that are infrequent, leaving only those that are frequent => F_k+1

Where- F_k: frequent k-itemsets
L_k: candidate k-itemsets

IMPLEMENTATION: Discovering Frequent Events

After applying all the required pre-processing steps, the first step applied in the *Learning Phase* was to discover the frequent events, or to discover which sensors were activated simultaneously most frequently. This is the rudimental step to gain insights about the user preferences which would help to build the recommender system. The *Apriori* algorithm discussed before was used to carry this step out in *Python* using *MLxtend*[6] library for frequent patterns.

The result-altering key parameter for generating the frequent event set, which can be set manually as required, is the *Minimum Support Threshold*, discussed earlier in the theory of *Apriori*. The default value of this support threshold is set at 0.5. For the initial trial run it was set at 0.005, to observe what number of frequent sets could be discovered. A further implementation to get the length of the itemset was also executed.

```
frequent_itemsets = apriori(newdata, min_support=0.005,use_colnames=True)

frequent_itemsets['length'] = frequent_itemsets['itemsets'].apply(lambda x: len(x))
frequent_itemsets
```

	support	itemsets	length
0	0.251055	(wardrobe)	1
1	0.077574	(tv)	1
2	0.162378	(oven)	1
3	0.016579	(officeLight)	1
4	0.017391	(officeDoor)	1
...	...	...	...
1113	0.008458	(bedroomDoor, kitchenDoor, bathroomDoor, fridg...	8
1114	0.008458	(bedroomDoor, kitchenDoor, bathroomDoor, fridg...	8
1115	0.040525	(oven, bedroomDoor, kitchenDoor, fridge, kitch...	8
1116	0.008458	(bedroomDoor, kitchenDoor, bathroomDoor, fridg...	8
1117	0.008458	(bedroomDoor, kitchenDoor, bathroomDoor, fridg...	9

1118 rows × 3 columns

Code Snippet: *Discovering frequent items sets from the processed data along with the length of the itemset.*

The result above shows 1118 frequent events were generated. This could be filtered further by increasing the minimum support threshold.

[6] http://rasbt.github.io/mlxtend/api_subpackages/mlxtend.frequent_patterns/

As the number of frequent sets is too large and would result in even more number of rules, probably exponentially large, most of which would be uninteresting and unimplementable in the recommender system. One way to reduce this dataset would be to increase the minimum support threshold but there's a risk of missing out interesting sets. An alternative solution to this has been shown in (Verma, 2009) in what is called *Maximal Frequent* and *Closed Frequent* itemsets.

Maximal Frequent Itemset

A frequent itemset whose none of the immediate supersets are frequent can be called as a Maximal Frequent Itemset. To find these, examine the itemsets that appear on the borders or edge of frequent and infrequent itemsets. If all of the immediate supersets of the itemset is infrequent, it can denoted as maximal frequent. This is depicted in *Fig 4.8*, where the frequent sets *ad* and *abc* have all immediate supersets (*abd, acd*, and *abcd*) infrequent. Therefore, *ad* and *abc* are maximal frequent.

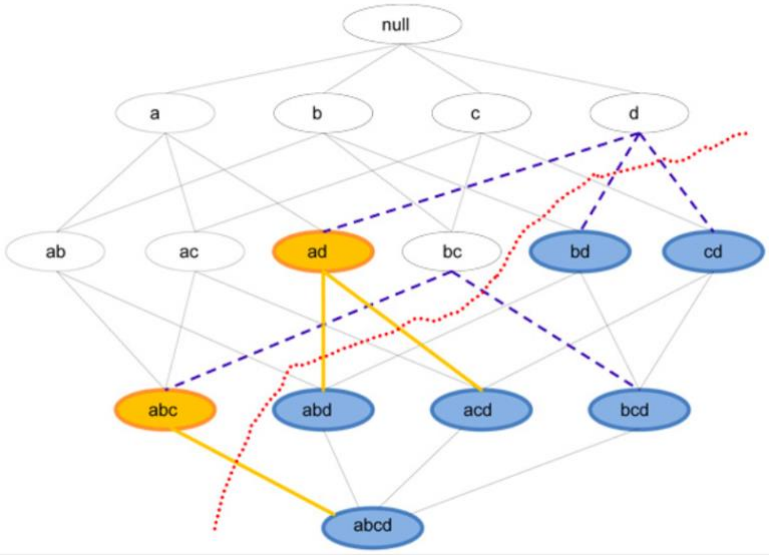


Figure 4.8 *Examining edges of frequent and infrequent itemsets for Maximal Frequent*

This extremely compact representation of frequent itemsets is helpful when the dataset is very large and itemsets required is small. This has been implemented as shown below.

```
maximal_frequent_itemsets = frequent_itemsets.copy()

for index, row in frequent_itemsets.iterrows():
    itemset = row['itemsets']
    itemsets_to_compare_with = frequent_itemsets.drop(index)
    for i, r in itemsets_to_compare_with.iterrows():
        if itemset.issubset(r['itemsets']):
            maximal_frequent_itemsets = maximal_frequent_itemsets.drop(index)
            break;
```

```
      support                                          itemsets  length
8    0.006081                                         (mainDoor)       1
177  0.005021              (wardrobe, bedroomLight, bedroomDoor)       3
295  0.006338         (bathroomCarp, mainDoorLock, bathroomDoor)       3
712  0.007903  (bedroomDoor, kitchenDoor, wardrobe, mainDoorL...       5
970  0.028700  (bedroomDoor, bathroomDoor, wardrobe, mainDoor...       6
971  0.012964  (bathroomCarp, bedroomDoor, bathroomLight, war...       6
972  0.020867  (bedroomDoor, bathroomDoor, bed, wardrobe, mai...       6
995  0.016915  (bedroomDoor, kitchenDoor, tv, couch, mainDoor...       6
1008 0.050538  (oven, bedroomDoor, kitchenDoor, kitchenLight,...       6
1024 0.016400  (officeDoor, bedroomDoor, officeLight, officeC...       6
1047 0.068790  (bedroomDoor, bathroomDoor, bed, mainDoorLock,...       6
1106 0.020936  (bedroomDoor, bathroomDoor, tv, couch, wardrob...       8
1115 0.040525  (oven, bedroomDoor, kitchenDoor, fridge, kitch...       8
1117 0.008458  (bedroomDoor, kitchenDoor, bathroomDoor, fridg...       9
(14, 3)
```

Code Snippet: *Implementing maximal frequent itemsets on already derived frequent itemsets*

The number of frequent sets has been reduced to 14 as shown in the snippet above. Another useful representation, which is not this dense and filtered, is closed frequent itemsets.

Closed Frequent Itemset

A frequent itemset which does not have any superset that has the same support count as the original itemset is called closed frequent itemset. After identifying all the frequent itemsets, check if the superset of each has support count as same as the original itemset. If yes then disqualify those, and the ones with none are closed frequent. In Fig 4.9, there are 9 frequent itemsets, out of which 4 (*a, c, cd, abd*) are closed frequent and 2 (cd, abd) are maximal frequent.

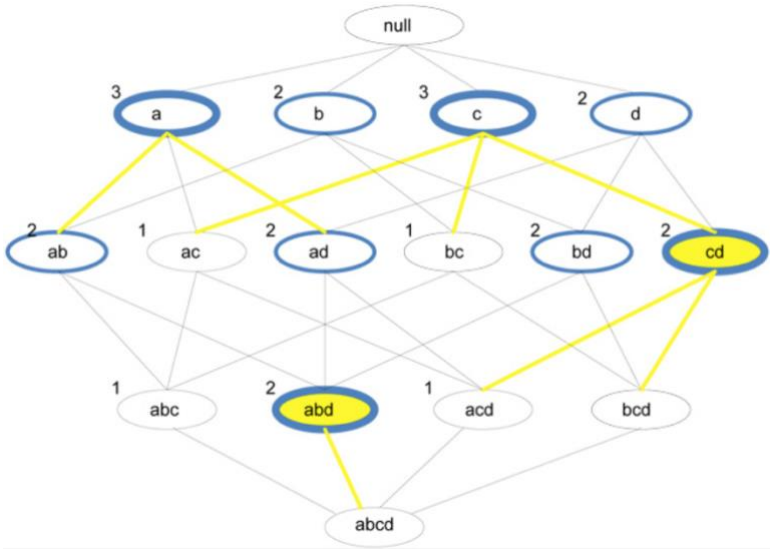


Figure 4.9 *Examining for Closed Frequent Itemsets*

Closed frequent is less sparse than maximal frequent, and its implementation and result is shown in the snippet below.

```
closed_frequent_itemsets = frequent_itemsets.copy()

for index, row in frequent_itemsets.iterrows():
    itemset = row['itemsets']
    itemsets_to_compare_with = frequent_itemsets.drop(index)
    for i, r in itemsets_to_compare_with.iterrows():
        if itemset.issuperset(r['itemsets']):
            if row['support'] == r['support']:
                closed_frequent_itemsets = closed_frequent_itemsets.drop(index)
                break;
```

```
      support                                           itemsets  length
0    0.251055                                          (wardrobe)       1
1    0.077574                                                (tv)       1
2    0.162378                                              (oven)       1
3    0.016579                                         (officeLight)       1
4    0.017391                                          (officeDoor)       1
..        ...                                                 ...     ...
347  0.055499          (bathroomCarp, hallwayLight, bedroomDoor)       3
349  0.099540          (bedroomCarp, hallwayLight, bathroomDoor)       3
354  0.116753                    (bed, bedroomCarp, bedroomDoor)       3
460  0.028700  (wardrobe, bedroomCarp, hallwayLight, bathroom...       4
609  0.089657      (bed, mainDoorLock, bedroomCarp, bedroomDoor)       4

[114 rows x 3 columns]
```

Code Snippet: *Implementing closed frequent itemsets on already derived frequent itemsets*

The number of frequent sets is 114 as shown above, which is less than the maximal frequent itemset of 14 and the original frequent itemset as 1118. A comparative relation between these three types of frequent item sets is depicted in *Fig 4.10.*

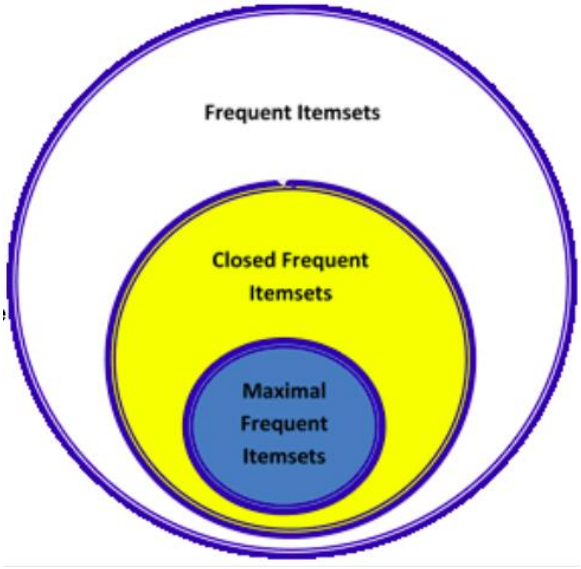


Figure 4.10 *Relation between the three types of itemsets*

Maximal and closed itemset both are a subset of frequent itemset, but maximal is more compact. “Closed frequent itemsets are more widely used than maximal frequent itemset because when efficiency is more important than space, they

provide us with the support of the subsets so no additional pass is needed to find this information", says (Verma, 2009). And this has been found suitable in this study as well, and therefore closed frequent itemset was utilised for generating association rules ahead.

Another way of filtering the frequent itemset would be through gauzing parameters like the support and length of the itemset as shown in snippet below.

```
frequent_itemsets[ (frequent_itemsets['length'] >= 3) &
                   (frequent_itemsets['support'] >= 0.1) ]
```

	support	itemsets	length
137	0.109791	(wardrobe, mainDoorLock, hallwayLight)	3
140	0.210767	(wardrobe, mainDoorLock, bedroomDoor)	3
141	0.130766	(wardrobe, mainDoorLock, bedroomCarp)	3
168	0.109791	(wardrobe, hallwayLight, bedroomDoor)	3
178	0.169984	(wardrobe, bedroomCarp, bedroomDoor)	3
...	...	...	...
1011	0.102620	(oven, bedroomDoor, fridge, kitchenCarp, kitch...	6
1015	0.102620	(oven, bedroomDoor, kitchenDoor, fridge, kitch...	6
1020	0.102620	(oven, bedroomDoor, kitchenDoor, fridge, kitch...	6
1029	0.112049	(bedroomDoor, kitchenDoor, fridge, kitchenCarp...	6
1093	0.102620	(oven, bedroomDoor, kitchenDoor, fridge, kitch...	7

130 rows × 3 columns

Code Snippet: *Filtering the frequent itemsets manually by tuning support and length*

This is more mechanical and nescient way, but may be suitable in many cases. In this study, the theoretical approach of closed frequent itemset works better because of its innate and conscious methodology.

IMPLEMENTATION: Learning Association Rules

After discovering the frequent itemsets and narrowing the list down by using the Closed Frequent Itemset approach, the next step in the *Learning Phase* is to learn the association rules out of those frequent itemsets. These rules are the user preferences in form of antecedents and consequents, as defined earlier. These rules can be considered as the *if-then* statements that shows the probabilistic relationship between the occurrence of consequents in presence of the antecedent. For example in the context here, if an antecedent {Oven, Kitchen Cupboard} is associated with a consequent {Radio}, this could be interpreted

as when the inhabitant uses the oven and opens kitchen cupboard, i.e. he/she is cooking, the radio was also often found to be active at the same time. This is learning the user preferences of 'listening to radio while cooking'.

A frequent itemset containing n number of items can create $3^n - 2^{n+1} + 1$ rules ((Tan, et al., 2005). This shows that even frequent itemset containing dozens of items may create hundreds of rules, most of which will be not relevant and not suitable for our recommendation system. Too frequent suggestions may not only annoy the inhabitant, but also belittle the value of the important and interesting rules. An example of thousands of rules learned by using the large frequent itemset, even with high confidence threshold, is shown below.

```
rule = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.9)
rule
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(bedroomLight)	(wardrobe)	0.005051	0.251055	0.005021	0.994118	3.959765	0.003753	127.320696
1	(wardrobe)	(bedroomDoor)	0.251055	0.918434	0.251055	1.000000	1.088809	0.020477	inf
2	(tv)	(mainDoorLock)	0.077574	0.897429	0.077574	1.000000	1.114294	0.007957	inf
3	(tv)	(livingCarp)	0.077574	0.118327	0.077574	1.000000	8.451122	0.068395	inf
4	(tv)	(couch)	0.077574	0.111742	0.077505	0.999106	8.941218	0.068837	993.961109
...	...	...	...	...	...	...	...	...	...
14626	(kitchenCarp, wardrobe)	(kitchenLight, bedroomDoor, hallwayLight, frid...	0.009082	0.008458	0.008458	0.931298	110.113413	0.008381	14.432450
14627	(kitchenCarp, bathroomDoor)	(kitchenLight, bedroomDoor, hallwayLight, frid...	0.009082	0.008458	0.008458	0.931298	110.113413	0.008381	14.432450
14628	(fridge, wardrobe)	(kitchenLight, bedroomDoor, hallwayLight, kitc...	0.008458	0.008804	0.008458	1.000000	113.581552	0.008383	inf
14629	(fridge, bathroomDoor)	(kitchenLight, bedroomDoor, hallwayLight, kitc...	0.008458	0.008804	0.008458	1.000000	113.581552	0.008383	inf
14630	(kitchenDoor, bathroomDoor)	(kitchenLight, bedroomDoor, hallwayLight, kitc...	0.009151	0.008458	0.008458	0.924242	109.279221	0.008380	13.088359

14631 rows × 9 columns

Code Snippet: *Learning thousands of uninteresting and irrelevant rules by using large frequent itemsets*

Therefore, it is important to filter out unimportant rule by setting reasonable threshold for metrics like confidence, support, lift, etc. (Kotu & Deshpande, 2014), and by using more compact itemsets like closed frequent itemsets. Therefore, thorough consideration of setting threshold of relevant metrics to filter out irrelevant and unimportant is the key to learn the right rules. The association rules in *MLxtend* provides all the five parameters namely *support, confidence, lift, leverage* and *conviction* to choose from and set the thresholds. This may be chose as per the requirement of the application. Here, the *Lift* metric was chosen and implemented as it brings out the most interesting rules even from the less frequent events (Azevedo & Jorge, 2007). The list of rules was further filtered down by setting a threshold for

Confidence, which assures the reliability factor of the occurrence of consequent in presence of antecedent. Therefore, the number of rules generated to learn the user preference was reduced to 6 as shown below.

```
rules= association_rules(closed_frequent_itemsets, metric="lift", min_threshold=8)
rules
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(couch)	(tv)	0.111742	0.077574	0.077505	0.693610	8.941218	0.068837	3.010624
1	(tv)	(couch)	0.077574	0.111742	0.077505	0.999106	8.941218	0.068837	993.961109
2	(officeLight)	(office)	0.016579	0.016757	0.016400	0.989247	59.035614	0.016122	91.441619
3	(office)	(officeLight)	0.016757	0.016579	0.016400	0.978723	59.035614	0.016122	46.220809
4	(couch, kitchenDoor)	(tv)	0.016985	0.077574	0.016915	0.995918	12.838231	0.015598	225.994266
5	(kitchenDoor, tv)	(couch)	0.016985	0.111742	0.016915	0.995918	8.912688	0.015017	217.623299
6	(couch)	(kitchenDoor, tv)	0.111742	0.016985	0.016915	0.151378	8.912688	0.015017	1.158367
7	(tv)	(couch, kitchenDoor)	0.077574	0.016985	0.016915	0.218052	12.838231	0.015598	1.257136
8	(oven, hallwayLight)	(fridge)	0.040961	0.112049	0.040525	0.989362	8.829752	0.035936	83.467427
9	(fridge)	(oven, hallwayLight)	0.112049	0.040961	0.040525	0.361676	8.829752	0.035936	1.502432

```
rules= rules[ (rules['confidence'] >0.8)]
rules
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
1	(tv)	(couch)	0.077574	0.111742	0.077505	0.999106	8.941218	0.068837	993.961109
2	(officeLight)	(office)	0.016579	0.016757	0.016400	0.989247	59.035614	0.016122	91.441619
3	(office)	(officeLight)	0.016757	0.016579	0.016400	0.978723	59.035614	0.016122	46.220809
4	(couch, kitchenDoor)	(tv)	0.016985	0.077574	0.016915	0.995918	12.838231	0.015598	225.994266
5	(kitchenDoor, tv)	(couch)	0.016985	0.111742	0.016915	0.995918	8.912688	0.015017	217.623299
8	(oven, hallwayLight)	(fridge)	0.040961	0.112049	0.040525	0.989362	8.829752	0.035936	83.467427

Code Snippet: *Filtering rules by setting metric thresholds to learn the interesting rules only*

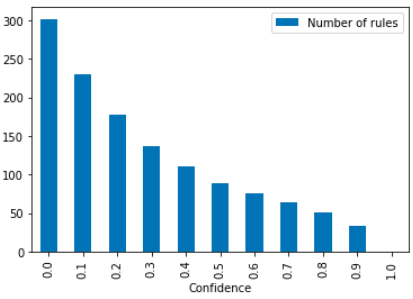
The number of rules generated by varying the thresholds can also be graphically analysed to find the elbow points or points of convergence as shown below.

```
confidence = [0, .1, .2, .3, .4, .5, .6, .7, .8, .9, 1]
number_of_rules = []

for i in confidence:
    a = association_rules(closed_frequent_itemsets, metric="confidence", min_threshold=i)
    number_of_rules.append(len(a))

data = {'Confidence': confidence, 'Number of rules': number_of_rules}
df2 = pd.DataFrame(data=data)
print(df2)
plot = df2.plot.bar(x='Confidence', y='Number of rules')
```

```
    Confidence  Number of rules
0          0.0              302
1          0.1              230
2          0.2              178
3          0.3              137
4          0.4              110
5          0.5               89
6          0.6               76
7          0.7               64
8          0.8               51
9          0.9               33
10         1.0                0
```



Code Snippet: *Visualising the number of rules generate by varying the threshold*

Another way of filtering rules from a larger set of rules would be to filter out by multiple metrics, as well as length of antecedents and consequents as shown in the snippet ahead.

```
rules['ant_leng'] = rules['antecedents'].apply(lambda x: len(x))
rules['con_len'] = rules['consequents'].apply(lambda x: len(x))

rules[ (rules['lift'] > 10) &
       (rules['confidence'] >= 0.95) &
       (rules['support'] > 0.02) &
```

Code Snippet: *Filtering by multiple metrics and length of antecedents and consequents*

Any of the filtering methods shown above could be applied as suited by the application. Visualization is can also be an intuitive method to find out interesting rules and have a comparative analysis of the rules with key metrics. For example, the dot on the top at most

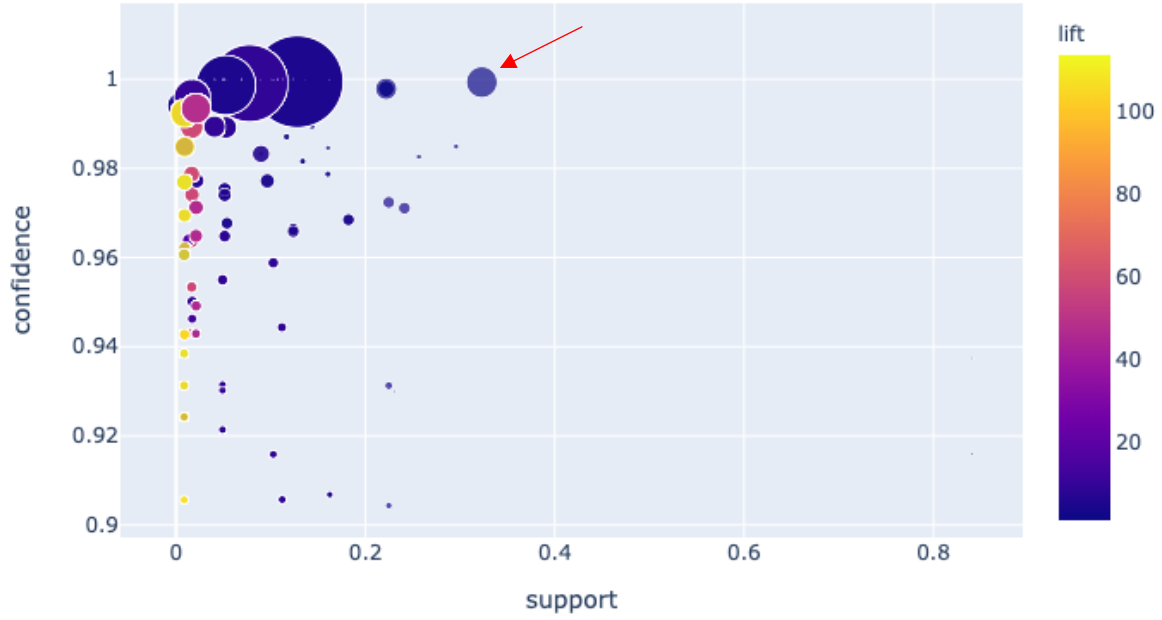


Figure 4.11 *Visualizing rules across three metrics- confidence, support & lift*

right of *Fig. 4.11* can be shown to have a good support and confidence, but with low lift measure. This tells that this rule will stand true in almost every instances or is causally related, but is less interesting.

The complex rules containing many antecedents and consequents can also be effectively visualised by *Directed Graphs* using NetworkX[7] library in python. The directed graphs uses nodes, edges and arrows to show relation between entities. In *Fig. 4.12* the yellow nodes are the rules, the green nodes are the antecedents and consequents, and the arrows coming

[7] https://networkx.github.io/documentation/stable/index.html

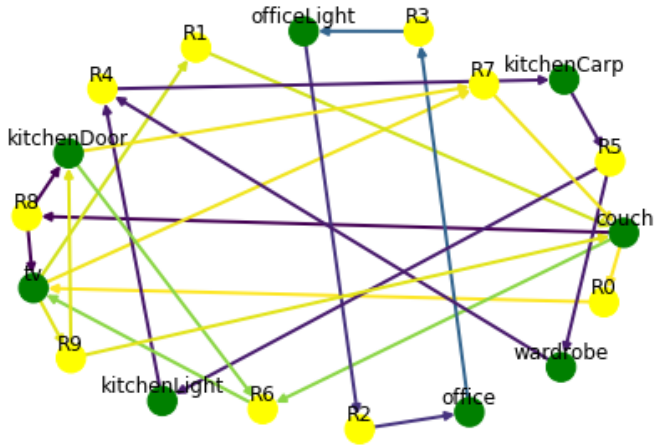


Figure 4.12 *Directed network graph analysis from NetworkX library in Python*

towards the rules depict the relation of antecedent to the rule and the arrow going out from the rule shows the relation of consequent to the rule. Altogether, these show a network of interrelated rules sharing common elements. These graph can be utilised to predict future links, or activities of inhabitants ahead of these rules. Further study is recommended on the directed graph analysis and link prediction for implementation of recommender systems.

4.4 Data Preparation & Modelling: Phase-2 (Recommendation Phase)

After the completion of Phase-1, the Phase 2 i.e. the *Recommendation Phase* begins. During the preparation stage, the contexts are extracted from the raw data, and then these contexts and the rules are merged with the original data to create a dataset for the modelling stage. In the modelling stage, a supervised learning problem is created and solved to predict the user action in a given context using the learned rules. The developments and finding are discussed in detail in the following section.

4.4.1 DATA PRE-PROCESSING: Phase-2

The *Recommendation Phase* begins with processing the raw data to extract the key contextual information. As mentioned earlier and emphasised by (Verkasalo, 2015) and

(Rasch, 2014), the contextual information is an additional important knowledge that augments sense to the whole data. Therefore, it's important to incorporate such knowledge to gain advantage in decision making, i.e. recommending more accurate automation services to the inhabitants.

Along with the *activity* attributes already present in the data, three other contexts were derived . They are *time of the day*, *type of the day* and user *location* within the house. These are conducted as shown below:

- *timeofday:* The hours of the day are discretised into six parts of the day, adding a context in regards to set of activities done during these group of hours, as shown below.

```
timeofday=[]
for i in range(0,len(rawdata)):
    a=rawdata.timestamp[i].hour

    if a>=0 and a<4:
        tod= 'Post Midnight'
    elif a>=4 and a<8:
        tod= 'Early Morning'
    elif a>=8 and a<12:
        tod= 'Morning'
    elif a>=12 and a<16:
        tod= 'Afternoon'
    elif a>=16 and a<20:
        tod= 'Evening'
    elif a>=20 and a<24:
        tod= 'Night'

    #print(tod, rawdata.timestamp[i])

    timeofday.append(tod)
```

Code Snippet: *Extracting the context timeofday from the timestamp*

- *typeofday:* The days of the week are further discretised into a *weekday* or *weekend,* as the activities of the inhabitants might change significantly during these two types of day off the week.

```
typeofday=[]
for i in range(0,len(rawdata)):
    a=rawdata.timestamp[i].weekday()

    if a>4:
        daytype= 'Weekend'
    else:
        daytype= 'Weekday'

    #print(daytype, calendar.day_name[a], rawdata.timestamp[i])

    typeofday.append(daytype)
```

Code Snippet: *Extracting the context typeofday from the timestamp*

- *location:* This is another important context which indicates the location of the inhabitant with respect to the part of house he/she currently is in. This derived from the sensors states located in the carpets of every part of the house, representing presence of the inhabitant in that part.

```
location=[]
for i in range(0,len(rawdata)):

    if rawdata.officeCarp[i]==1:
        loca= 'Office'
    elif rawdata.livingCarp[i]==1:
        loca= 'Living'
    elif rawdata.kitchenCarp[i]==1:
        loca= 'Kitchen'
    elif rawdata.bedroomCarp[i]==1:
        loca= 'Bedroom'
    elif rawdata.bathroomCarp[i]==1:
        loca= 'Bathroom'
    else:
        loca= 'Hallway/Others'

    location.append(loca)
```

Code Snippet: *Extracting the context location from the carpet sensors*

- These *contexts* are added as attributes to the individual rows of the raw data as shown

```
rawdata['timeofday'] = timeofday
rawdata['typeofday'] = typeofday
rawdata['location'] = location
rawdata.head()
```

...	bathroomLight	bathroomDoorLock	bathroomDoor	bathroomCarp	Activity	timestamp	timeofday	typeofday	location	None
...	0	0	0	0	sleep	2016-02-01 08:04:10	Morning	Weekday	Bedroom	1
...	0	0	0	0	sleep	2016-02-01 08:04:11	Morning	Weekday	Bedroom	1
...	0	0	0	0	sleep	2016-02-01 08:04:12	Morning	Weekday	Bedroom	1
...	0	0	0	0	sleep	2016-02-01 08:04:13	Morning	Weekday	Bedroom	1
...	0	0	0	0	sleep	2016-02-01 08:04:14	Morning	Weekday	Bedroom	1

Code Snippet: *Adding the extracted contexts as attributes to the each row of raw data*

The next step in preparation of data is to extract the antecedents and the consequents of the rules learned in Phase-1 and transforming it in useable form. The items of the antecedents and consequents are extracted and separated in a form to match with each of the attributes of the raw data as shown in the snippets below.

```
#Converting frozenset items to strings from the Rules DataFrame
temp= pd.DataFrame()
temp['antecedent'] = rules['antecedents'].apply(lambda x: ', '.join(list(x))).astype("unicode")
temp['consequent'] = rules['consequents'].apply(lambda x: ', '.join(list(x))).astype("unicode")
```

Code Snippet: *Converting antecedents and consequents from the rules in useable format*

```
df1=temp.antecedent.str.split(", ", expand=True,)
df1.columns = [f'antecedent_{i}' for i in df1.columns[0:]]

df2=temp.consequent

result = pd.concat([df1, df2], axis=1, sort=False)
result.replace(to_replace=[None], value='None', inplace=True)
result =result.reset_index(drop = True)
```

	antecedent_0	antecedent_1	consequent
0	tv	None	couch
1	officeLight	None	office
2	office	None	officeLight
3	couch	kitchenDoor	tv
4	kitchenDoor	tv	couch
5	oven	hallwayLight	fridge

Code Snippet: *Extracting the items from the antecedents and consequents into separate columns to match with the raw data*

In the next step, which is an extremely important one, the rules are merged with the raw data. Here the items of antecedents and the consequents in each rule are matched with their respective attributes for an 'active' state(1), and each row is returned and appended to form a new data frame that has the rules and contexts logically embedded in them as shown below. The duplicates are also dropped to avoid redundancy.

```
rawdata['None']=1
context= pd.DataFrame()
for i in range(0,len(result)):
    con = rawdata[(rawdata[result.antecedent_0[i]] == 1)  &
                  (rawdata[result.antecedent_1[i]] == 1) &
                  (rawdata[result.consequent[i]] == 1)]
    con['consequent']= result.consequent[i]
    context= context.append(con)
```

Code Snippet: *Logically embedding the rules and context into the raw data, and dropping the duplicates*

In the next step, a final dataframe is created including required attributes only which will be used in the modelling. Here the consequents are merged into an attribute called *action* which indicated the next action the inhabitant would prefer to execute in a given contextual setup.

```
newdf= context
newdf= newdf.drop(['timestamp','None'], axis=1)
newdf
```

wardrobe	tv	oven	officeLight	officeDoorLock	officeDoor	officeCarp	office	mainDoorLock	mainDoor	...	Activity	timeofday	typeofday	location	consequent
0	1	0	0	0	0	0	0	1	0	...	eat	Evening	Weekday	Living	couch
0	1	0	0	0	0	0	0	1	0	...	leisure	Evening	Weekday	Living	couch
0	1	0	0	0	0	0	0	1	0	...	leisure	Evening	Weekday	Living	couch
0	1	0	0	0	0	0	0	1	0	...	leisure	Evening	Weekday	Living	couch
0	1	0	0	0	0	0	0	1	0	...	leisure	Evening	Weekday	Living	couch
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
0	0	1	0	0	0	0	0	1	0	...	eat	Evening	Weekday	Kitchen	fridge
0	0	1	0	0	0	0	0	1	0	...	eat	Evening	Weekday	Kitchen	fridge
0	0	1	0	0	0	0	0	1	0	...	eat	Evening	Weekday	Kitchen	fridge
0	0	1	0	0	0	0	0	1	0	...	eat	Evening	Weekday	Kitchen	fridge
0	0	1	0	0	0	0	0	1	0	...	eat	Evening	Weekday	Kitchen	fridge

16938 rows × 34 columns

Code Snippet: *The final data frame with required attributes and the label 'action'*

This is the final data with 34 columns including the sensor data, contextual information, and the consequents or the user preferences. This dataset is ready to export for modelling where the label will be the *consequent* attribute.

4.4.2 MODELLING: Phase-2

In this modelling phase, supervised learning algorithms were used to train a model which predicts and recommends the action of the inhabitant in a given contextual setup, based on the rules learned. Here, various algorithms were compared using the rapid prototyping in *Rapid Miner.* A brief description of the algorithms used, and the development and findings of the modelling will be discussed in details in the section ahead.

The dataset created from the previous step which contains the desired user action or preferences stored in the *consequent* column will be treated as a label for a supervised learning classification problem. Various models will be trained, tested and compared using this data in *Rapid Miner*. Based on the nature of discrete information stored in the dataset and on the previous studies conducted in similar field , the following models were chosen to build the predictive model:

I. Generalized Linear Model (GLM)
II. Deep Learning

III. Gradient Boosted Tree

I. Generalized Linear Model (GLM)

The Generalized Linear Model or GLM is an adaptable generalization of the conventional Linear Regression that takes into consideration reaction factors that errors distributions that deviate from the normal distribution. In other words, GLM is a generalized application of linear Regression where the linear model is associated to a response variable by means of *Link Function* that permits the variance's magnitude of every measurement to be a function of the estimated value (Agresti, 2007).

The Generalized Linear Model id built-up of a Linear predictor:

$$\eta_i = \beta_0 + \beta_1 x_{1i} + \ldots + \beta_p x_{pi}$$

and two other functions:

a) *Link Function*: A function that describes how the predictor is a factor of the mean μ_i :

$$g(\mu_i) = \eta_i$$
where: $E(Y_i) = \mu_i$

The link function specifies the link between random and systematic components. It explains how the anticipated value of the response relates to the linear predictor of explanatory variables; E.g. , $\eta = g(E(Yi)) = E(Yi)$ for linear regression, or $\eta = \text{logit}(\pi)$ for logistic regression.

b) *Variance Function*: A function that describes how the variance *var*(Y_i) is dependent on the mean μ_i :

$$var(Y_i) = \phi V(\mu)$$
where: ϕ dispersion parameter is a constant

This refers to the probability distribution of the model, which may be normal distribution in case of linear regression or binomial distribution in case of logistic regression.

Advantages of GLM:

- No need to transform the response *Y* into a normal distribution
- More flexibility, as the choice of link is different from the choice of random component or variance function.
- Maximum likelihood estimation is used for model fitting, therefore optimal properties if estimators (Zheng & Agresti, 2000).
- GLM does not assumes a linear relation between the independent and the dependent variables, hence adaptable to variety data via link function

II. Deep Learning

Deep Learning is a subset of Machine Learning that is inspired by the structure of a human brain, called as artificial neural network as shown in *Fig. 4.13*. The difference between ML and DL is that the features on which the model operates is picked out by neural network without the human intervention, unlike ML where the features are manually annotated.

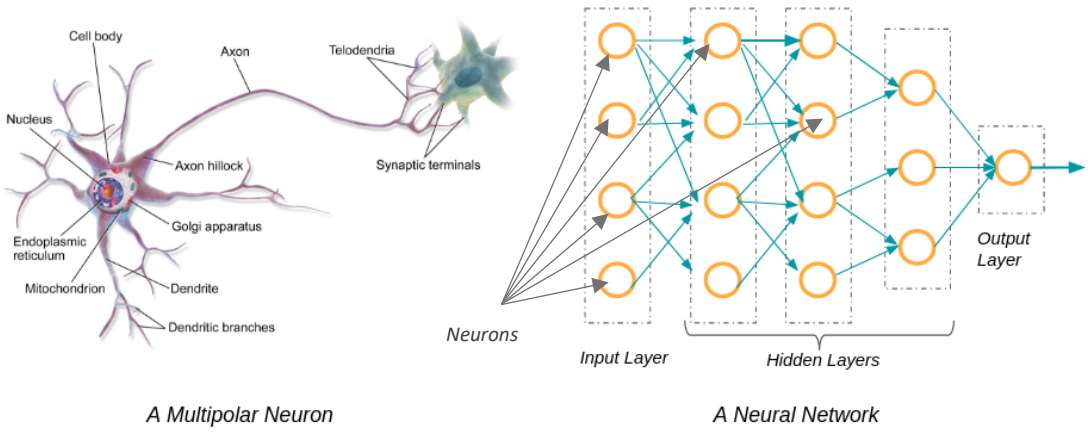
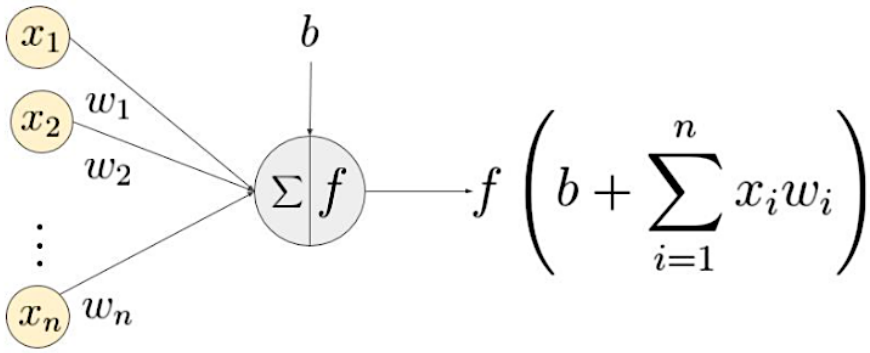


Figure 4.13 *A human brain structure compared to that of an Artificial Neural Network used in Deep Learning*

The *neuron*, the core entity of a neural network, is where all information processing takes place (Nielsen, 2019). Each of the features are fed to the neurons in the first layer of the neural network called as the input layer as shown in *Fig. 4.12.* On the other end, there's the output layer with each neuron representing the result or the output, with hidden layers existing between the input and the output. The information is transferred from one layer to another over connecting channels, each of these has a value attached to it and hence is called a weighted channels. All neurons has a unique number associated with them called bias. This bias is called added to the weighted sum of the inputs, reaching the neurons which is then applied to a function known as the *Activation Function Y*.

$$\text{Y = Activation}(\Sigma(weight * input) + bias)$$



An example of a neuron showing the input (x_1 - x_n), their corresponding weights (w_1 - w_n), a bias (b) and the activation function f applied to the weighted sum of the inputs.

Figure 4.14 *Deep Learning functioning explained*

The result of this function determines if the neuron gets activated as shown in *Fig. 4.14*. Every activated neuron passes on information to the following layers, this continues to the second last layer. The one neuron activated in the output layer corresponds to the output or a label class. The weights and bias are continuously adjusted to produce a well-trained network. Some activation functions used commonly in deep learning algorithms is shown in *Fig 4.15.*

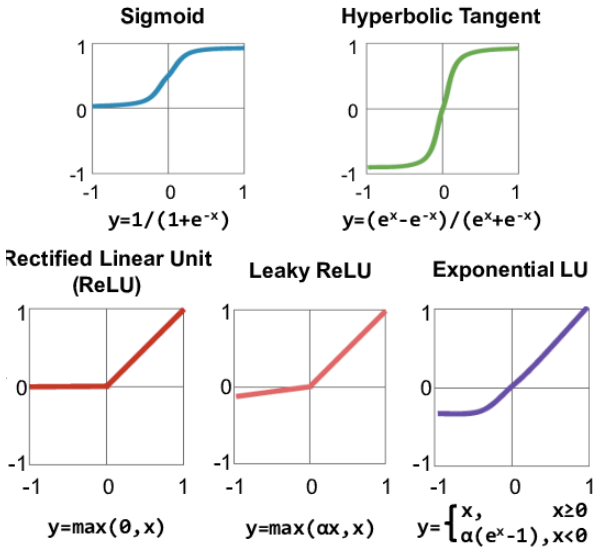


Figure 4.15 Different types of Activation Functions

III. Gradient Boosted Tree (GBT)

Introduced first in 1999 by Jerome Friedman, Gradient Boosted Tree algorithm uses *Decision Trees* with *Boosting* method to combine many weak classifier into strong one. Boosting is also called as 'Stagewise Additive Modelling', where one model after the other is trained based on the outputs from previous model. GBT sequentially ensembles weak decision trees models in a way that the new models fits in the residual from the previous step in order improve the model as shown in the *Fig 4.16.*

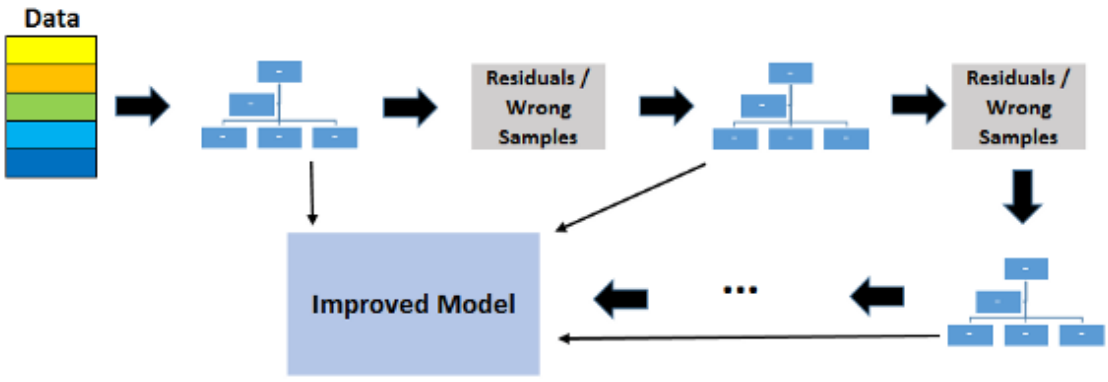


Figure 4.16 Gradient Boosted Trees

It's an additive regression model over an ensemble of trees, fitted to current residuals, gradients of the loss function, in a forward step-wise manner. This algorithm favours multiple trees with shallow depth.

The algorithm of gradient boosting as depicted in (Jiao, et al., 2006) is shown *Fig 4.17*

- Initialize $F_0(x) = \arg\min_\rho \sum_{i-1}^{N} L(y_i, \rho)$
- For $m = 1$ to M do:
  - Step 1. Compute the negative gradient

$$\tilde{y}_i = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F_{x_i}}\right]$$

  - Step 2. Fit a model

$$\alpha_m = \arg\min_{\alpha,\beta} \sum_{i=1}^{N} [\tilde{y} - \beta h(x_i; \alpha_m)]^2$$

  - Step 3. Choose a gradient descent step size as

$$\rho_m = \arg\min_{\rho} \sum_{i-1}^{N} L(y_i, Fm - 1(x_i) + \rho h(x_i; \alpha))$$

  - Step 4. Update the estimation of $F(x)$

$$F_m(x) = F_{m-1}(x) + \rho_m h(x; \alpha_m)$$

- end for
- Output the final regression function $F_m(x)$

Figure 4.1 *Gradient Boosting Algorithm*

$$F_m(x) = F_{m-1}(x) + \nu \cdot \sum_{l=1}^{L} \gamma_{lm} 1(x \in R_{lm})$$

$$\text{gradient}_{m+1}(x) = \text{label}(x) - F_m(x)$$

Hyper parameters, learning rate and n-estimators, are the two critical parts that affects the performance & accuracy of the model. The GBT are more accurate as compared to the random forest, can handle mixed type of data with almost no processing, but requires critical tuning of the hyperparameters as the tree may overfit if number of trees (n-estimators) is large.

These output of the three models -Generalized Linear Model, Deep Learning and Gradient Boosted Trees- are evaluated and discussed in detail in the next section.

Chapter 5 – Evaluation and Discussion

The results of the modelling done in previous section will be analysed and their implications with regards to this study will discussed in detail in this section.

5.1 Result Evaluation

During the Phase-1, i.e. *Learning Phase*, it was observed that the data collected from the sensors installed in a smart home environment, when carefully processed and treated, were capable of learning the habits/preferences of the inhabitants. Using association analysis algorithms like *Apriori*, association rules were generated which could be seen analogous to the user preferences in real life. Meaningful rules or preferences were discovered. For example from a rule in Fig.5.1 it can be inferred that when the inhabitant sits on the couch

antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
(tv)	(couch)	0.077574	0.111742	0.077505	0.999106	8.941218	0.068837	993.961109
(officeLight)	(office)	0.016579	0.016757	0.016400	0.989247	59.035614	0.016122	91.441619
(office)	(officeLight)	0.016757	0.016579	0.016400	0.978723	59.035614	0.016122	46.220809
(couch, kitchenDoor)	(tv)	0.016985	0.077574	0.016915	0.995918	12.838231	0.015598	225.994266
(kitchenDoor, tv)	(couch)	0.016985	0.111742	0.016915	0.995918	8.912688	0.015017	217.623299
(oven, hallwayLight)	(fridge)	0.040961	0.112049	0.040525	0.989362	8.829752	0.035936	83.467427

Figure 5.1 *Rules discovered in Learning Phase*

while the kitchen door was open, probably after making a snack in the kitchen, he/she prefers to watch TV. Or the office light was found turned on at times when the occupant was working in the office, or the fridge was found to be open when the oven was turned on for cooking probably. These user preferences or rules are also marked by high values of *confidence* meaning high co-occurrence of the antecedent and the consequent, high values of *lift* meaning interesting rules, high *conviction* and low *leverage* meaning high dependence on the antecedent. These rules are logically founded and makes sense most in a real home environment. Therefore, learning preferences from the interaction of inhabitants from the smart devices in home can be done by association analysis algorithm.

In the Phase-2, the user contexts such as location and time of the day were extracted from the given dataset and merged with the rules generated in Phase-1. After the required processing, the data was modelled using Rapid Miner and the results were generated.

Model	Accuracy	Classification Error	Average Class Precision	Average Class Recall	Training Time (1,000 Rows)	Scoring Time (1,000 Rows)
Generalized Linear Model	80.28 ± 0.3%	19.72 ± 0.3%	78.90%	60.64%	358 ms	588 ms
Deep Learning	80.14 ± 0.1%	19.86± 0.1%	46.42%	60.00%	816 ms	831 ms
Gradient Boosted Trees	80.22 ± 0.1%	19.78± 0.1%	60.03%	75.67%	117 ms	650 ms

Table 5.1 *Results from Phase-2 Modelling*

From the overview of performance of the models shown in *Table 5.1,* it can be observed that all the models have almost same classification accuracy, but differ significantly in class precision and class recall measures. Also, the train and score time varied notably. The results and performance of individual models is shown ahead.

5.1.1 Generalized Linear Model

	TRUE COUCH	TRUE OFFICE	TRUE OFFICELIGHT	TRUE TV	TRUE FRIDGE	CLASS PRECISION
PRED. COUCH	2224	2	0	469	0	**82.52%**
PRED. OFFICE	0	1	0	0	0	**100.00%**
PRED. OFFICELIGHT	0	469	472	0	0	**50.16%**
PRED. TV	11	0	0	18	0	**62.07%**
PRED. FRIDGE	0	1	1	1	1169	**99.74%**
CLASS RECALL	**99.51%**	**0.21%**	**99.79%**	**3.69%**	**100.00%**	

Table 5.2 *GLM Confusion Matrix with Class Precision and Recall*

The confusion matrix for GLM classification is shown in *Table 5.2* and is observed that the precision is fairly distributed for each class and is not biased towards one class, while the recall measure shows a poor score for the two classes i.e. class *office* and class *TV.*

The weights given to the attributes to build the predictive model through GLM is shown in *Fig 5.2.* It is observed that along with the sensor attributes, contextual attributes like *location*, *timeofday,* and *activity* has also been given comparatively large weights.

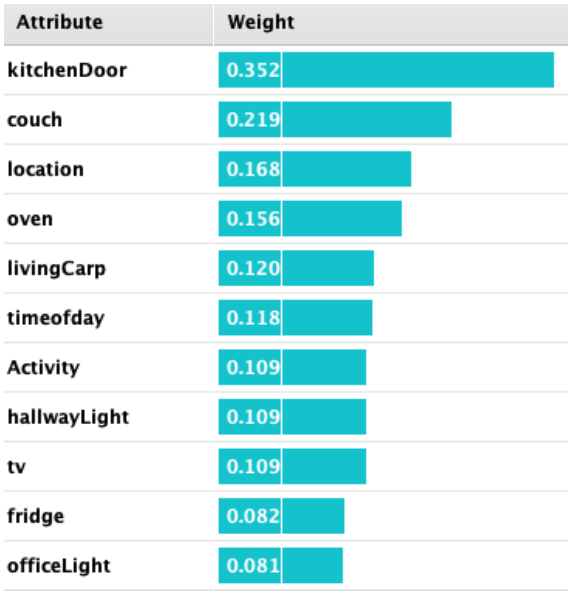


Figure 5.2 *Weights applied to various attributes in GLM*

5.1.1 Deep Learning

	TRUE COUCH	TRUE OFFICE	TRUE OFFICELIGHT	TRUE TV	TRUE FRIDGE	CLASS PRECISION
PRED. COUCH	2235	0	0	488	0	**82.08%**
PRED. OFFICE	0	0	0	0	0	**0.00%**
PRED. OFFICELIGHT	0	473	473	0	0	**50.00%**
PRED. TV	0	0	0	0	0	**0.00%**
PRED. FRIDGE	0	0	0	0	1169	**100.00%**
CLASS RECALL	**100.00%**	**0.00%**	**100.00%**	**0.00%**	**100.00%**	

Table 5.3 *Deep Learning Confusion Matrix with Class Precision and Recall*

The confusion matrix for Deep Learning classification is shown in *Table 5.3* and is observed that the precision as well as the recall are biased towards few of the classes. Especially, class *office* and class *TV* were not predicted at all giving them a class precision and class recall score of 0%.

The production model generated for this algorithm is shown in *Fig 5.3.* It is observed that a ten iteration were applied using four layer learning model with varying units in each of them. The output layer is the five classes to be predicted and uses a *SoftMax* function.

```
MSE: 0.09984413
RMSE: 0.3159812
R^2: 0.9646692
logloss: 0.27668974
mean_per_class_error: 0.4

Layer Units      Type Dropout       L1       L2 Mean Rate Rate RMS Momentum Mean Weight Weight RMS Mean Bias Bias RMS
    1   121     Input  0.00 %
    2    50 Rectifier       0 0.000010 0.000000  0.766729 0.407025 0.000000   -0.002754   0.102964  0.364558 0.102270
    3    50 Rectifier       0 0.000010 0.000000  0.569701 0.440843 0.000000   -0.022829   0.134497  0.526024 0.260309
    4     5   Softmax         0.000010 0.000000  0.357347 0.443218 0.000000   -0.138953   0.421237 -0.072341 0.136549
```

Figure 5.3 *Production Model of Deep Learning model applied*

The Fig. 5.4 shows the weights applied to each of the attributes for this model. Contextual attribute *location* has been given the highest weigh followed by sensor attributes and other contextual attributes.

Attribute	Weight
location	0.312
couch	0.224
kitchenDoor	0.195
kitchenCarp	0.173
livingCarp	0.161
Activity	0.156
hallwayLight	0.135
officeCarp	0.127
typeofday	0.123
oven	0.114
officeDoor	0.114
timeofday	0.108
office	0.105

Figure 5.4 *Weights applied to various attributes in Deep Learning Model*

5.1.3 Gradient Boosted Tree

	TRUE COUCH	TRUE OFFICE	TRUE OFFICELIGHT	TRUE TV	TRUE FRIDGE	CLASS PRECISION
PRED. COUCH	1752	0	0	0	0	**100.00%**
PRED. OFFICE	0	0	0	0	0	**0.00%**
PRED. OFFICELIGHT	0	473	473	0	0	**50.00%**
PRED. TV	484	0	0	487	0	**50.15%**
PRED. FRIDGE	0	0	0	0	1170	**100.00%**
CLASS RECALL	**78.35%**	**0.00%**	**100.00%**	**100.00%**	**100.00%**	

Table 5.4 *Gradient Boosted Tree Confusion Matrix with Class Precision and Recall*

It can be observed from the confusion matrix in *Table 5.4* that the class precision and recall for all classes except class *office* is fair and none of the instances in this class was predicted correctly by this model.

The model applies 30 iterations, with a tree for each class in every iteration. The tree in first iteration for class *couch* and the tree in last iteration is for the same as shown in *Fig. 5.5,* and is clearly visible how the tree transform through iterations by the gradient boosted method.

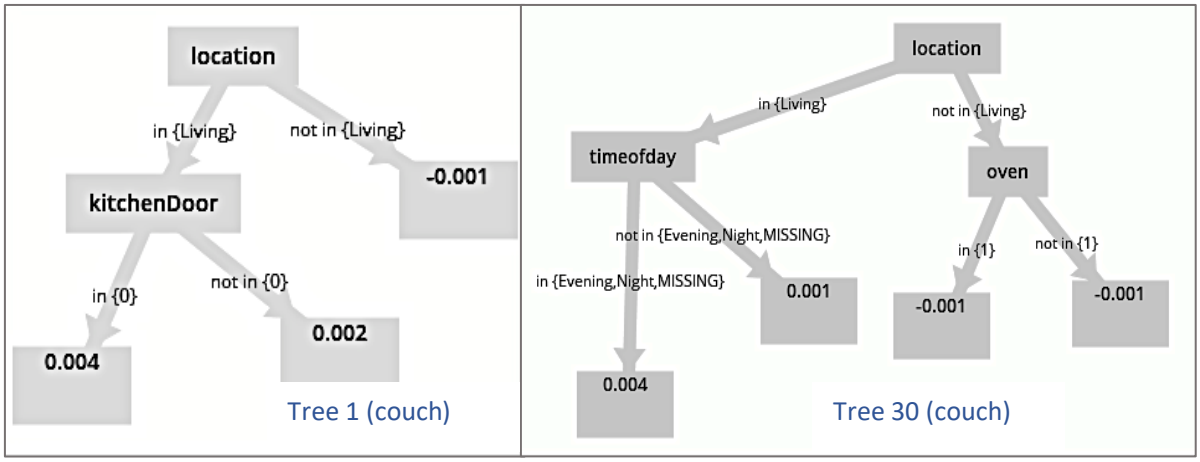


Figure 5.5 *Tree for class couch in first iteration and the last iteration*

Fig. 5.6 shows the weights applied to the attributes in the gradient boosted model. Here again, the contextual attributes like location, timeofday and activity has been given priority by assigning higher weights as compared to most other attributes.

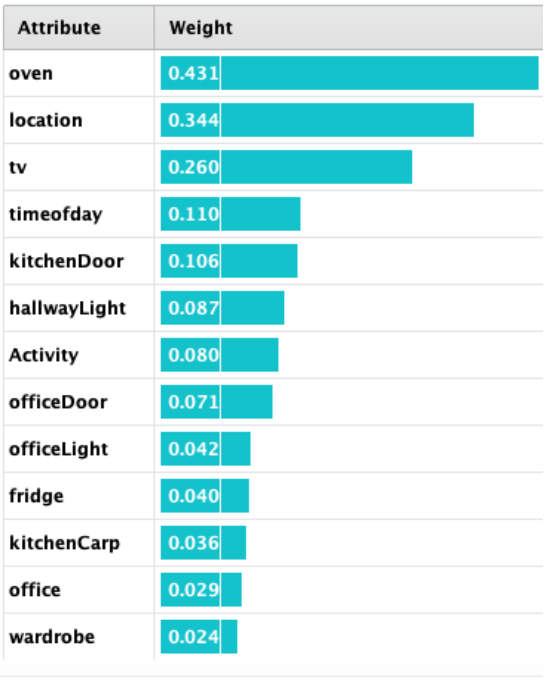


Figure 5.6 *Weights applied to various attributes in Gradient Boosted Trees Model*

5.2 Discussion

The proceedings conducted through this entire study focusses on the novel research problem of building a recommender system for a smart home environment that learns the user preferences from past interactions with IoT devices, and applying those preferences to predict future user actions and recommending automation services while augmenting contextual information available. The user preferences were discovered during the *Learning Phase* where unsupervised association analysis algorithm, *Apriori*, was applied to mine the frequent events and the user preferences in the form of rules. In the next *Recommendation Phase*, these rules were combined logically with the contextual information extracted from the already available data and fed to supervised learning algorithms to build a predictive model for recommending actions to the user in a given situation.

The premise of using Apriori algorithm and the theory of frequent itemsets and rules was that the habits (or the user preferences here) can be considered as routine of behaviours repeated regularly that tends to happen subconsciously (Gardner, 2015). During the *Learning Phase* it

was found that the size of frequent itemsets and the resulting rules from it could be inappropriately large, if not treated properly. Various methods for reducing these numbers, such as mechanical methods where the association metrics are gauzed to reduce the size, and conceptual methods where the size of the sets is reduced consciously by dealing with the innate characteristics of these frequent itemsets. It was found that reducing the itemset size by making use of *Closed Frequent Itemset* method worked well in this study as the rules generated were compact in size and logically founded. However, other methods mentioned may also be relevant, depending on the size of the data employed and the nature of application involved.

The second phase i.e. *Recommendation Phase* involved a series of data pre-processing steps followed by modelling in Rapid Miner. Collating the extracted contextual information with the rules learned entails considerable effort and was facilitated by data manipulation library *pandas* in *Python.* The supervised learning models utilised in this phase to predict the label *consequents,* or the actions preferred by the user, shows an almost similar accuracy score across the three models. However, a more appropriate measure of success for prediction when the classes are imbalanced, as in this case, would be precision and recall. While precision measure the relevancy of the result, recall measures how many truly relevant results are returned. On these two parameters, it was observed that the Generalized Linear Model scored better and was not biased to any of the classes in general. Also the training time was better than that of Deep Learning model, and the scoring time was slightly more than that of the Gradient Boosted Tree. Recommended in (Belghini, et al., May 2016), Artificial Neural Network, which is the shallow version (fewer hidden layer) of the Deep Learning utilised in this study and it seems to work just at par with GLM and GBT in terms of accuracy and no better in terms of precision-recall measures. A probable reason of GLM working better in study would be the error distribution of the response variable. The application of the other two models shall be compared with GLM in further studies involving a larger dataset with complex rules having multiple antecedents and consequents.

An interesting point observed in building this context-aware recommender system was that the supervised models that performed comparatively better on precision-recall measure used higher weights for the contextual attributes. This means that the systems performing better

on these measures actually accounted for contextual intelligence as emphasised in previous works of (Verkasalo, 2015) and (Rasch, 2014). The GBT tree had 3 contextual attributes in top 7 weights, GLM again had 3 in top 7 while Deep Learning had only 2 in top 7 weights. Hereafter, a comparative study to see the effect of use of contextual information in the performance of recommender systems for smart home environment is suggested.

As also suggested in (Gohari, et al., 2018), an interesting adaptation from the results obtained by the supervised learning models would be to use the confidence levels of the class predicted to form a ranked list of recommendation rather than a single recommendation. As shown in *Fig. 5.7*, it can be observed that the classes were predicted with certain class confidence, which could be utilised to form a ranked list of recommendation. This will not only give the user a broad range of actions to choose from providing flexibility, but also will reduce the chances of incorrect recommendation by offering an educated ranked list.

consequent ↑	prediction(consequent)	confidence(couch)	..	..	confidence(tv)	confidence(fridge)
couch	tv	0.407	.	.	0.427	0.165
couch	couch	0.802	.	.	0.032	0.165

Figure 5.7 *Class confidence provided by the predictive supervised learning model*

An example of recommendation for the two instances shown in *Fig. 5.7* is depicted in the *Table. 5.5*, which shows a ranked list of preferred user actions

Instances	Recommendation
Instance # 1	1.) *Would you like to turn on the TV?*
	2.) Would you like to turn on the heaters on the couch?
	3.) *Would you like to open the refrigerator?*
Instance # 2	1.) *Would you like to turn the on heaters on the couch?*
	2.) *Would you like to open the refrigerator?*
	3.) *Would you like turn on the TV?*

Table 5.5 *Ranked list of recommendation based on Class Confidence*

Chapter 6 – Conclusion

This research work presented an embodiment of a context-aware recommender system for smart-home automation that utilises unsupervised algorithms to discover user-preferences and supervised algorithms to predict user actions. The contextual features were extracted from the given dataset using the temporal attributes and sensor outputs, and merged coherently with the preferences. Based on the inhabitant's current state of affair or the context, the proposed system recommends services to automate the actions that user may intend to perform next. The model was implemented and evaluated using a publicly available dataset. It was found that association analysis algorithms like *Apriori* are competent to discover recurring user actions and establish user-preference in form of association-rules. Conscious filtering of these rules resulted in meaningful user-preferences. The three supervised learning algorithms -*Deep Learning, GLM* and *GBT*- implemented to recommend services by predicting the user action were found adequate and achieved similar accuracy scores. However, they varied significantly in precision-recall scores, and it was noticed that the algorithms assigning higher weights to the contextual information seemed to score higher on these metrics. Also, utilization of class confidence to curate a ranked list of recommendation is also suggested for better user-experience by increasing flexibility.

Future Work

The foundational work laid in this study opens various avenues of future works extending the research in this field. One avenue to pursue in future is to compare the efficiency of the supervised learning algorithms for a realistically large dataset with complex association-rules, and examine the variance of outputs to find the most suitable model for this application. Other avenue of future work would be to enable learning of preferences in a multi-occupant home. A glance of directed-network graphs for rules generated has been seen in this study and found potential, therefore a further research in employing graph analytics for link prediction in this application would be suggested because of the associative and discrete nature of the data used in this application.

REFERENCES

Agresti, A., 2007. Generalized Linear Models. In: *An Introduction to Categorical Data Analysis.* Hoboken, New Jersey: John Wiley & Sons, pp. 65-98.

Alshammari, T., Alshammari, N., Sedky, M. & Howard, C., 2018. *SIMADL: Simulated Activities of Daily Living Dataset.* Basel, Switzerland: MDPI AG.

Azevedo, P. J. & Jorge, A. M., 2007. *Comparing Rule Measures for Predictive Association Rules.* s.l., Springer, Berlin, Heidelberg.

Belghini, N., Gouttaya, N., Bouab, W. & Sayouti, A., May 2016. Pervasive Recommender System for Smart Home Environment. *International Journal of Applied Information Systems (IJAIS),* 10(9).

Brush, A. B. et al., 2011. *Home Automation in the Wild: Challenges and Opportunities.* s.l., Microsoft Research.

Davidoff, S. et al., 2006. Principles of Smart Home Control. *UbiComp 2006: Ubiquitous Computing,* pp. 19-34.

Eggen, B., Hollemans, G. & Sluis, R. v. d., 2003. Exploring and Enhancing the Home Experience. *Cognition, Technology and Work,* 5(1).

Engelmann, K. F., Holthaus, P., Wrede, B. & Wrede, S., 2016. *An Interaction-Centric Dataset for Learning Automation Rules in Smart Homes.* Portorož, Slovenia, Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16).

Fernandes, E., Riva, O. & Nath, S., 2015. *My OS ought to know me better: In-app behavioural analytics as an OS service.* s.l., Workshop on Hot Topics in Operating Systems (HotOS).

Forbes, 2013. *5 Time-Tested Success Tips From Amazon Founder Jeff Bezos.* [Online] Available at: https://www.forbes.com/sites/johngreathouse/2013/04/30/5-time-tested-success-tips-from-amazon-founder-jeff-bezos/#1f994b28370c
[Accessed June 2020].

Gardner, B., 2015. A review and analysis of the use of 'habit' in understanding, predicting and influencing health-related behaviour. *Health Psychology Review,* 9(3), pp. 277-295.

Gohari, F. S., Aliee, F. S. & Haghighi, H., 2018. A new confidence-based recommendation approach: Combining trust and certainty. *Information Sciences,* Volume 422, pp. 21-50.

Jiao, F., Xu, J., Yu, L. & Schuurmans, D., 2006. *Protein Fold Recognition Using The Gradient Boost Algorithm.* s.l., Computer System Bioinformatics Conference.

Kotu, V. & Deshpande, B., 2014. *Predictive Analytics and Data Mining.* 1st Editionv ed. s.l.:Morgan Kaufmann.

Louis, J. & Rashid, K., 2018. *Utilizing Building Information Models as Operating Systems for Smart Homes.* s.l., Oregon State University.

Lueth, K. L., 2015. *The 10 most popular Internet of Things applications right now.* [Online] Available at: https://iot-analytics.com/10-internet-of-things-applications/ [Accessed May 2020].

Marr, B., 2015. *17 'Internet Of Things' Facts Everyone Should Read.* [Online] Available at: https://www.forbes.com/sites/bernardmarr/2015/10/27/17-mind-blowing-internet-of-things-facts-everyone-should-read/
[Accessed 23 July 2020].

Netflix, 2020. *Netflix- Personalization & Search.* [Online] Available at: https://research.netflix.com/business-area/personalization-and-search [Accessed 2 August 2020].

Nielsen, M., 2019. *Neural Networks and Deep Learning,* s.l.: http://neuralnetworksanddeeplearning.com/.

Rasch, K., 2014. An unsupervised recommender system for smart homes. *Journal of Ambient Intelligence and Smart Environments,* Issue 6, pp. 21-37.

Tan, P.-N., Steinbach, M. & Kumar, V., 2005. *Introduction to Data Mining.* 1st Edition ed. s.l.:Addison-Wisley.

Tong, C., Wong, S. K.-S. & Lui, K. P.-H., 2012. The Influences of Service Personalization, Customer Satisfaction and Switching Costs on E-Loyalty. *International Journal of Economics and Finance,* 4(3).

Verkasalo, H., 2015. *System and Method for Behavioural and Contextual Data Analytics.* United States of America, Patent No. US9,075,883B2.

Verma, R., 2009. *Data Mining Hypertextbook,* s.l.: s.n.
Zheng, B. & Agresti, A., 2000. Summarizing the predictive power of a generalized linear model. *Statistics in Medicine,* Volume 19.

APPENDICES

This section guides through the list of files present in the 'Artefacts' folder and the steps required to implement the code in *Python* and the in processes in *Rapid Miner* developed for the dissertation project titled "*A Context-Aware Personalized Recommender System for Automation in IoT based Smart Home Environment*".

Contents of the Artefacts:

[1] Datasets

- **SmartHome_Data_Raw.csv** : The raw dataset obtained from the publisher of the dataset. This dataset is to be used in *Python* for implementing all the required pre-processing and unsupervised learning model of Phase-1 (Learning Phase)

- **SmartHome_Data_Processed.csv** : The processed dataset created by implementing all the necessary step in *Python.* This data set is to be used in *Rapid Miner* for supervised learning model of Phase-2 (Recommendation Phase)

[2] Python Contents

- **SmartHome_Recommender_System.ipynb** : This is the source file for the code developed in *Python Jupyter Notebooks* to perform the required pre-processing steps and to implement unsupervised learning for rule generation.

- **SmartHome_Recommender_System.html** : This same file as above but in HTML formal to view the outputs of the code without the need to run the program in python.

- **Datasets** : The input raw dataset and the output processed dataset could also be found here, as in the previous folder.

[3] Rapid Miner Contents

- **Comparison_Overview.csv** : This sheet contains the a comparative performance overview of all the three supervised learning algorithms implemented in Rapid Miner
- **Deep Learning**

- **DL_Process.rmp** : This is the Rapid Miner process file which includes all the steps implemented for predicting the label i.e. the user action.
- **DL_Weights.csv** : This sheet contains the weights applied to each attribute for the modelling.
- **DL_Predictions.csv** : This sheet contains the test dataset with the predictions done by the deep learning model.

- **Generalised Linear Model**
  - **GLM_Process.rmp** : This is the Rapid Miner process file which includes all the steps implemented for predicting the label i.e. the user action.
  - **GLM_Weights.csv** : This sheet contains the weights applied to each attribute for the modelling.
  - **GLM_Predictions.csv** : This sheet contains the test dataset with the predictions done by the GLM model.
- **Gradient Boosted Trees**
  - **GBT_Process.rmp** : This is the Rapid Miner process file which includes all the steps implemented for predicting the label i.e. the user action.
  - **GBT_Weights.csv** : This sheet contains the weights applied to each attribute for the modelling.
  - **GBT_Predictions.csv** : This sheet contains the test dataset with the predictions done by the GBT model.

[4] Readme

- **Readme.txt** : This is a text file that provides all the necessary instructions that helps to understand how to implement the codes in Python and to run the processes in Rapid Miner.

Note: Refer the readme.txt file to know the versions and necessary installations required for easy implementation of the artefacts.