# Electricity usage recommender system with limited input data


**Yelyzaveta Al-Dara**

| | |
|---|---|
| Main academic Supervisor: | Prof. Christos Tjortjis, International Hellenic University (IHU). |
| Academic co-supervizor: | Prof. Zacharie De Grève, University of Mons (UMONS). |
| Partner supervisor: | Thibaut Piraux, WeSmart. |


**A Master Thesis submitted for the Erasmus Mundus Joint Master Degree on Smart Cities and Communities (SMACCs)**

June 2021


University of Mons, Heriot Watt University, International Hellenic University, University of the Basque Country

# Abstract

Recommender Systems (RS) in the field of smart grids are gaining popularity. They help the consumers to make corrections to their behavior in order to assist them with achieving their goals and interests. However, the creation of a RS requires the availability of some previously existing data that can describe the interests of the users. In real life problems may arise when creating the RS due to the limited data available. In the literature these problems are often referred to as “cold start” problems.

In this work, the problem of creating an electricity usage RS, when limited data are available to be used as an input, is addressed. The input data was limited to the values of the whole-house aggregated electrical power consumption measured with a constant frequency. Data that describes the general characteristics of the building was also utilized. This is namely the type, location and the climate of the area where the building is situated.

The usage of a disaggregation algorithm was chosen as a first step included at the beginning of the RS. The suggestion is to utilize one of the training-less disaggregation algorithms based on the graph signal processing. Nevertheless, other training-less disaggregation algorithms could also be embedded at this part of the RS.

The second part is the RS itself, which is based on the principle of Collaborative-Filtering (CF) RS using the descriptive data to identify the neighboring buildings.

The results of the validation of the RS showed that it can generate recommendations with high accuracy in most of the cases. However, the accuracy drops when each of the neighboring buildings was identified based on the match of different descriptive features with the target building. Thus, some suggestions to improve the accuracy are also introduced.

**Keywords:** Recommender System (RS), Collaborative-Filtering (CF), Appliance disaggregation, Python.

Yelyzaveta Al-Dara

21/06/2021

# Table of Contents

**ABSTRACT** ..... **II**

**TABLE OF CONTENTS** ..... **III**

**1 INTRODUCTION** ..... **1**

**2 THEORETICAL BACKGROUND** ..... **5**

2.1 DATA ANALYTICS ..... 5

*2.1.1 Data preprocessing* ..... *5*

*2.1.2 Data analysis* ..... *6*

*2.1.3 Data postprocessing* ..... *7*

2.2 SMART GRIDS ..... 7

2.3 DATA ANALYTICS IN SMART GRIDS ..... 8

2.4 RECOMMENDER SYSTEMS (RS) ..... 9

2.5 RECOMMENDER SYSTEMS FOR SMART GRIDS ..... 10

**3 RELATED WORK** ..... **13**

3.1 RECOMMENDER SYSTEMS USING AGGREGATED DATA ..... 13

3.2 RECOMMENDER SYSTEMS USING DISAGGREGATED DATA ..... 14

3.3 COMPARATIVE SUMMARY OF RECOMMENDER SYSTEMS ..... 16

3.4 INTRUSIVE DISAGGREGATION ..... 17

3.5 NON-INTRUSIVE DISAGGREGATION ..... 18

**4 METHODOLOGY** ..... **19**

4.1 COLD-START PROBLEM ..... 19

4.2 FACTORS INFLUENCING RECOMMENDATION RATINGS ..... 20

4.3 DESCRIPTION OF THE SYSTEM ..... 21

4.4 NILM MODEL ..... 22

*4.4.1 Simplified explanation* ..... *23*

*4.4.2 Detailed description* ..... *28*

4.5 COLLABORATIVE-FILTERING MODEL ..... 33

**5 EXPERIMENTAL RESULTS** ..... **41**

5.1 ELECTRICITY CONSUMPTION DATASET ..... 41

5.2 REDD DATA ..... 42
5.3 RESULTS OF THE NILM MODEL ..... 46
5.4 RESULTS OF THE RECOMMENDER SYSTEM ..... 49
**6 CONCLUSIONS AND FUTURE WORK ..... 59**
**BIBLIOGRAPHY ..... 62**
**APPENDIX 1 ..... 69**
**APPENDIX 2 ..... 71**
**APPENDIX 3 ..... 81**
**APPENDIX 4 ..... 83**
**APPENDIX 5 ..... 87**
**APPENDIX 6 ..... 89**

# 1 Introduction

Recommender Systems (RS) are gaining popularity as such systems are created for different purposes. With the development of society and with the tremendous increase in the demand for products and the variety of available products on the market, the competition among manufacturers increases. RS are being developed to help consumers choose products [1] that suit their preferences and needs, out of the many options available on the market. This serves the interests of consumers by making it easier to find suitable products. RS also serve the interests of manufacturers by targeting consumers that are most likely to be interested in the product, which increases the advertisement effectiveness.

The most popular types of RS are the ones with commercial purposes advertising products like movies [2], cosmetics, food etc. However, RS can also be used for other purposes which are not meant to sell products.

One of the fields where RS can be used for non-commercial purposes is the field of smart grids. RS can suggest to the participants of a smart grid some actions that can improve the efficiency of their performance as smart grid players or of the system as whole. The RS can also play a vital role in assisting the users in reducing their energy bills and reduce their power consumption. In these terms RS can be very effective and helpful.

When there are already some existing data that can be used to develop the RS such as data about users, their preferences, ratings and feedbacks, data about the preferences of the target user, their feedbacks and ratings, many widely used algorithms can be applied to develop a RS. The challenge at this point arises when there is a need to create a RS for the first time, when there are no previously recorded data that can be used to train the recommender model.

This is the challenge that was faced by WeSmart in the attempt to develop a RS that would benefit the customers. The aim of the company was to create a RS that would give the customers useful recommendations on how to reduce their electricity consumption and their bills. However, the available data was limited to the values of electricity consumption of the building measured with constant frequency. There was no data about any ratings from the customers of any previously given recommendations that could be relied on when developing the RS.

Thus, this work was written in collaboration with WeSmart addressing this challenge that was encountered by the company and is relevant to many other cases when a new RS with limited input data is designed.

This situation is particularly relevant when developing a RS for less widely spread applications. In this case it can be challenging even to find some publicly available data to train the model on. This highlights the importance of finding possible solutions to this problem in order to develop the RS based on the limited amount of available data. The possibility of recycling the output of the system by adding these to the system database could also be an added feature to the system. This could make the system continuously improving with increasing accuracy.

So, this work addresses the problem of developing a RS in these conditions when only a limited amount of data is available. The application of the RS is in the field of smart grid favoring the interests of consumers to reduce their electricity bills.

The aim of this work is to meet the challenges that arise when designing an electricity usage RS suitable for limited input data.

Its main objectives are:

- To suggest a viable solution producing recommendations for electricity usage with limited available data.
- To evaluate the accuracy of the suggested solution and propose ways to improve the accuracy of the system in future.

In order to meet the objectives, an overview of the RS in general is provided in addition to RS in the context of smart grids. Information on the basic principles of how RS work and what are the necessary data for developing such system was researched. After that, a research of the state-of-the-art in this field was conducted, along with research on other issues pertinent to the creation of the system.

This thesis comprises the following chapters:

1) Theoretical background: consists of five sections in which the basic concepts used in this work are explained, including data analytics, smart grids, RS and the areas of intersection of these concepts.
2) Related work: gives an overview about existing RS related to electricity usage and smart grids, with a comparative summary of published work.

3) Methodology: details the proposed RS, explains each of its parts and provides suitable schematic representations.
4) Experimental results: provides an overview of the available datasets for electricity consumption. The data to be used for the evaluation of both the disaggregation part and the Collaborative Filtering (CF) part of the proposed system are described. The experimental results are presented along with their analysis, discussion and evaluation.
5) Conclusion and future work: includes the final discussion of the system, its advantages and disadvantages, in addition to pointers for improvement of system accuracy and quality in the future.

# 2 Theoretical background

Since this work is dedicated to RS in the context of electric grids and smart grids in particular, a theoretical background, based on which further research is developed, is provided here.

## 2.1 Data analytics

Data analytics is a popular term nowadays. The main distinction between data mining and data analytics is that the main purpose of data mining is to find some useful patterns in data while data analytics is an interdisciplinary term including the use of computer systems for analyzing datasets to make some decisions [3]. In other words, data mining is one of the key processes of data analysis [4]. Data analysis is conducted in almost every field, such as technology, healthcare, mobility, urban planning, smart grids etc.

When applying data analysis, the process normally includes a number of steps [3]. The first step is the assessment and selection of data, followed by cleaning and filtering. After that, a visual interpretation of the data is presented and analyzed. After the results of the analysis are ready, they should be adequately interpreted and evaluated. To summarize, four main phases of a data analysis project can be distinguished as shown in Figure 1 [5]**Error! Reference source not found.**.

Figure 1: Phases of data analysis projects [5]

### 2.1.1 Data preprocessing

According to [6] before data analysis is conducted, data should be collected, selected and preprocessed. It is necessary to preprocess the data received from measurements conducted in real life, because these data can contain errors and discrepancies, which result in data being incomplete, inconsistent, noisy. Furthermore, in many cases the amount of available data might be large and some of these data can be irrelevant for each task and

each specific result that is meant to be achieved analyzing these data. Thus, this step helps to select out of the multitude of data available the most relevant.

Duplicated records and anomalies are eliminated. Preparation produces data of higher quality by recovering the incomplete data, correcting errors and resolving any conflicts that may be present. Apart from that, [7] suggests that data should be cleaned by filling the missing values, smoothing out noisy data, correcting the inconsistencies of the data and resolving the redundancies. After that, data could be combined from multiple sources, if necessary, normalized and generalized. In case the amount of data seems to be large for the purpose of the work, data reduction can be applied. Discretization could be also applied in some cases. From this it can be said that there are no standard steps that should be applied to the dataset to be preprocessed. The literature suggests different steps to preprocess data. However, it can be concluded that data preprocessing is an important step of the data analysis process, which should not be skipped. Depending on the available data, it can be verified which preprocessing methods are relevant for each case.

### 2.1.2 Data analysis

Data can be analyzed using different techniques. Depending on their content, data can be quantitative or qualitative. Quantitative data represents measured quantities and normally represented by numbers. For example, 35kWh, 10MW, 15 minutes etc. Qualitative data gives more descriptive information that cannot be represented by numbers. For example, location, season, the type of building etc. In this work the tasks will be performed mainly by using the combination of available quantitative and qualitative data. The quantitative data represents the amount of electricity consumed and measured every 15 minutes in kWh. While the qualitative data represents the characteristics of the building such as the type, location and climate.

There are several existing techniques for data analysis. One of the common techniques for better understanding and analyzing quantitative data in particular, is visualization. It can be performed using any visual tool that makes it easier for people to see the structure of the data and understand the trends and patterns [8]. This technique has been used in many scientific papers and publications in various fields. Data can be visualized using pie charts [9], line graphs [10], bar charts [11] and other visual tools.

Correlation is a technique used to find the dependencies between features in data. It helps to observe the relationship between features and find possible causes of some occurring

effects. While regression focuses on estimating the function which defines the dependency between variables.

The difference between classification [12] and clustering is that classification is supervised learning with predefined classes, while clustering is unsupervised learning with no predefined classes.

So, in classification there are some previously labeled data that are fed into the model. These data are used to train the model to be able to identify which object belongs to which class. After the training stage is done, the model is run on testing dataset which is not labeled. The task of the classification model is to assign each item of the testing dataset to one of the classes that the model was trained on. While clustering does not have training dataset. It uses unlabeled data as input. After that, it forms groups, which are referred to as clusters. These clusters contain the items of the database that are similar to each other based on some predefined similarity measures.

### 2.1.3 Data postprocessing

After data analysis is conducted and the results are obtained, the results are interpreted, documented and evaluated. In other words, the results can be either directly in the system or as a base for it. The results can be also visualized, summarized, transformed to a different format and tested [13]. This work will also use data analysis to extract information from the data and support making decisions to provide electricity usage recommendations.

## 2.2 Smart grids

The definition of the term “smart grid” varies from one source to another. Smart grids were defined by the European Union (E.U.) as electrical networks with intellectual integration of the interaction between the users in the network, which enables bi-directional power flows. Smart grids must ensure security, cost-effectiveness and environmental sustainability [14].

Another definition for a smart grid was given by Paul et al. [15], as a combination of power generation and distribution system in one frame, which makes the system cleaner. This definition in some way is different and focused more on the ecological aspect that this term incorporates.

While conducting research on the improvement path of China's smart grid security control Teng defined a smart grid as the power grid that integrates modern computer, information, communication and advanced sensor technologies into its physical base [16]. The structure of a typical smart grid and the bi-directional information flow in it was clearly depicted in [17] as shown in Figure 2.

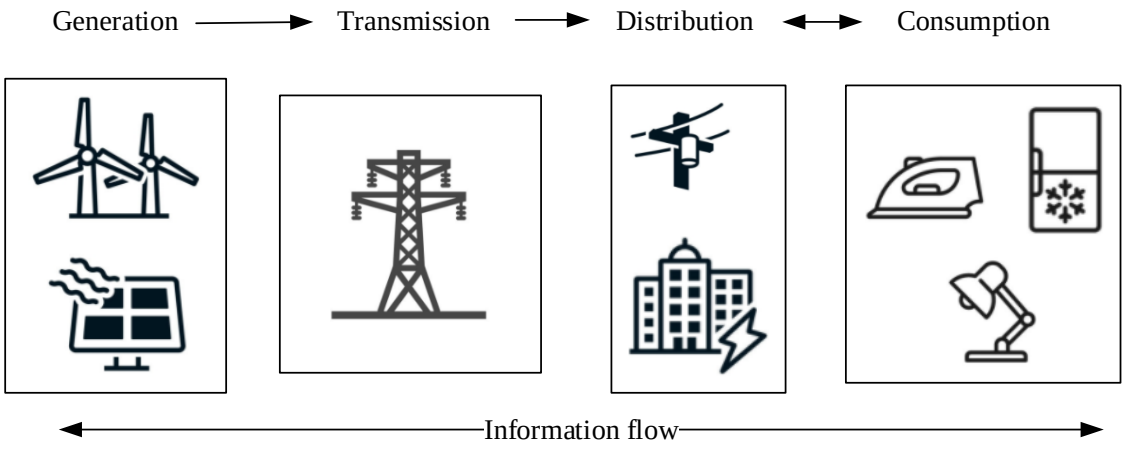



Figure 2: Smart grid structure [17]

The participants of a smart grid can be divided into three categories:

- Network operators: transmission and distribution network operators.
- Users: consumers, generators and storage owners.
- Other participants: suppliers, aggregators, applications and service providers [18].

The interaction of these participants with each other over the grid, the information and communication devices used, the information flow and the physical structure of the power network itself, form a smart grid.

## 2.3 Data analytics in smart grids

Data analytics is a powerful tool that is nowadays frequently used in the field of smart grids. There are a variety of applications of data analytics in smart grids. A brief summary of the uses of data analytics techniques and their applications in smart grids is depicted in Figure 3 [19].

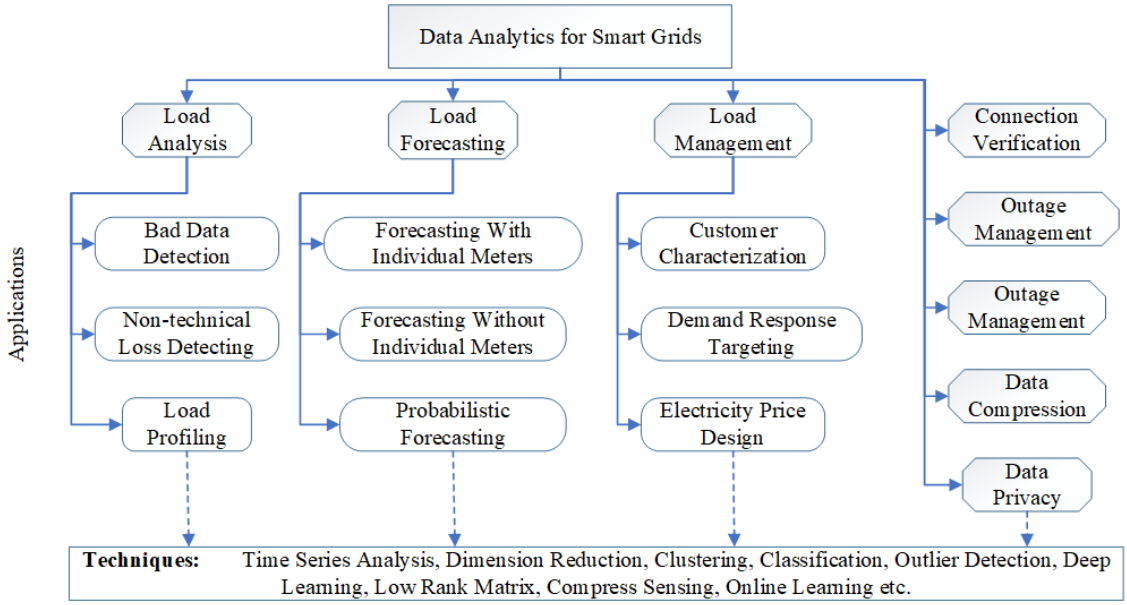



Figure 3: Taxonomy of smart meter data analytics [19]

Data analytics can favor energy market players on different levels. For consumers, individual load forecasting is performed in order to increase the efficiency of energy consumption [20]. However, most of the works in data analytics from the consumer side is focused mainly on achieving load forecasting to adjust the amount of generated energy.

It can be noted that data analytics can also be used as a tool to increase the efficiency of electrical energy usage from the consumer side by providing the users with recommendations contributing to reducing the electricity bills costs [21].

## 2.4 Recommender Systems (RS)

With the rapid development of technologies and the available products on the market, the popularity of RS is growing tremendously. A RS was defined [22] as a technique or a tool that provides the user with suggestions of items. It was mentioned by Massa in [23] that the RS is “a technique that is able to cope with Information Overload problem”. RS are indeed effective tools to tackle the challenges caused by the huge amount of information available. They automate the process of selecting relevant items for users and make it easier for the users to choose items that are right for them out of many similar items available.

In general, the items that RS suggest can be of any type such as movies, music, food, magazines, furniture etc. RS can be also used to suggest products for users based on information about the products they have purchased previously. For example, Netflix RS

that recommends movies and series to the user based on the movies and shows that the user has searched for or watched previously or based on the preferences set manually by the user. In fact, Netflix has complex algorithms that define the logic of their RS. The case of Netflix RS gained much interest among researchers that even some papers and theses have been dedicated to it [24], [25].

Another type of items that can be suggested by RS could be actions. RS can advise users to do some actions depending on some other actions that the user has done previously or depending on some other factors, such as user characteristics, categories etc. For example, it could be a system that recommends to a consumer in a smart grid to install solar panels based on their electricity consumption profile that shows high consumption of electricity from the grid during the day.

RS were classified into three categories [26]:

*Collaborative filtering (CF)-based RS:*

- User-to-user RS: This type of system is based on recommending to the user items that other users with similar characteristics have chosen or have been given a high rate previously.
- Item-to-item systems: The main idea behind these systems is that an item is considered to be in one category if it has the same users' like or dislike.

*Content based RS:* The logic of this type of system is based on recommending items that have some similarities to the items that the user has already purchased or were highly rated in the past [22]. In other words, these systems do not depend on the information provided by other users.

*Hybrid RS:* This type of RS combines several types of RS. The motivation for that is to eliminate the drawbacks that some types of systems may have.

## 2.5 Recommender systems for smart grids

Clustering is suggested as a tool for peaks shaving [27]. This technique can be used for identifying patterns in energy consumption, which could also be applied for energy models predictions. At the same time, by the means of clustering RS can be developed [28]. It has been denoted that in a large number of literature clustering is used for developing RS in the context of smart grids.

As a tool to improve the quality of RS including the context of smart grids in particular, past responses to recommendations [29] can be utilized for generating more effective recommendations for the consumer in future. Another point that can improve the quality of provided recommendations is providing incentives that are appropriate to the consumer [29]. In this regard the socio-economic state, specific aspects of the building, location, type of the facility etc. should be taken into account.

Overall, the usage of the RS in the field of smart grids is gaining popularity with new systems continuously being developed. These RS can target different goals and give different recommendations in the context of smart grids. Some of the systems can recommend to purchase new products that can improve the performance of the participant of the smart grid or the system in general. Other RS can suggest certain actions that the players of the smart grid can take to achieve some goals.

Chapter 3 provides more detailed analysis of the existing RS to be used in the context of smart grids.

# 3 Related work

Many papers have been published focusing on developing RS for smart grid users. Different types of RS were proposed. Each type provides a certain category of recommendations and targets certain participants of the smart grid. Some papers used the aggregated consumption data, others used a variety of methods in order to obtain the consumption data of each individual appliance of the dwelling.

The review starts by introducing some of the most popular RS for smart grids, followed by analyzing works that used aggregated data directly. After that, works that utilized disaggregated data for their RS will be discussed.

A summary of the most popular RS for smart grids was provided in [30]. According to it, some of the most widely spread smart grid RS are the following types:

- Energy Saving Electrical Appliance RS
- Electricity Retail Plan RS
- Household Demand Response Schedule RS
- Other RS, such as systems that recommend electrical appliances or products to utilities etc.

## 3.1 Recommender systems using aggregated data

An example of this type of RS is a system recommending electricity retail plans to the consumers that was developed without installing additional smart devices in the facility [31]. In this case the system recommends electricity retail plans depending on the overall household energy consumption data. The system was based on clustering the households by similarity of the electricity consumption curves, then recommending same retail plans for consumers of the same cluster. The system also has some space for improvement. In case of a new plan introduced, the system will not be able to recommend it to the users. Another note is that the individual characteristics such as socio-economic state, geographic location, type of the consumer (residential, industrial etc.) of each user that affect the ability of choosing a certain plan are not considered in this system. Another system recommending tariff plans for users was developed in [32] with the use of aggregated consumption data.

## 3.2 Recommender systems using disaggregated data

Electricity retail plans RS with enhanced accuracy was developed based on disaggregated data in [33]. For the development of this recommender system data from the Smart Grid Smart City (SGSC) project. These data include the overall consumption of electrical power of the dwelling and the power consumption of at least four appliances present in the dwelling. Although this RS was tested and proven by the authors to be effective and accurate, the computational speed remains to be an important area that requires for improvements to be done in.

Another prototype of a RS for saving electrical energy was proposed in [30]. The schematic representation of the system is provided in Figure 4 [30].

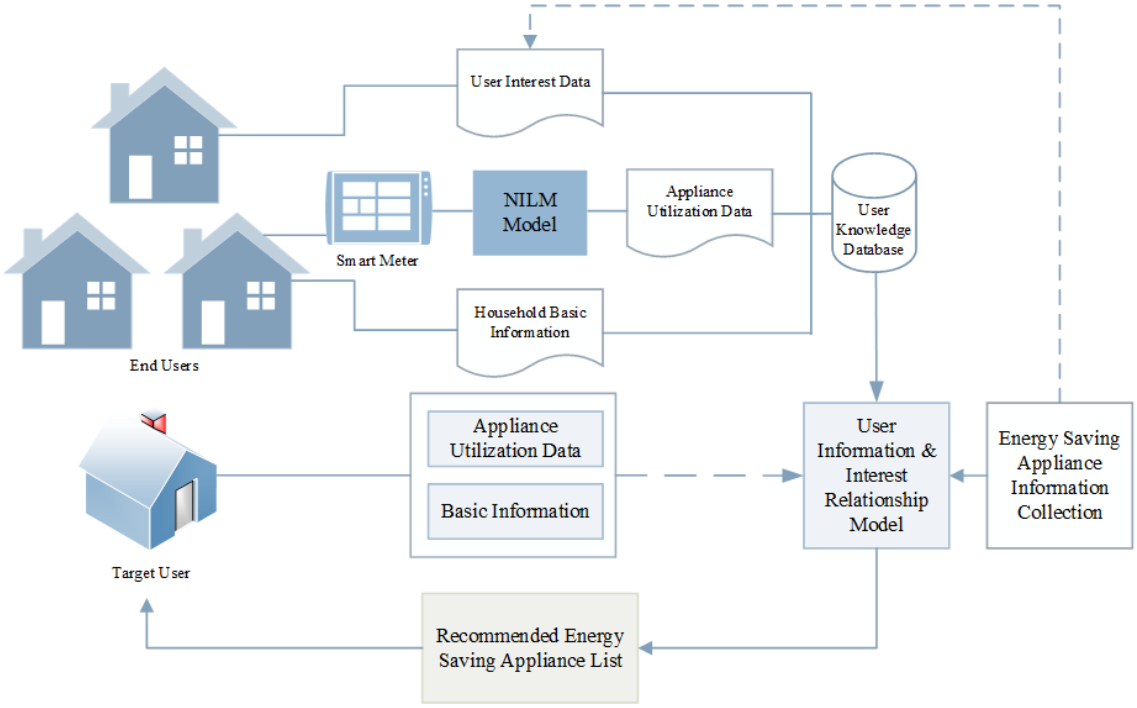



Figure 4: Energy Saving Electrical Appliance RS scheme [30]

This recommender system takes the overall energy consumption of the dwelling as an input. Then, it applies Non-Intrusive Load Monitoring (NILM) technique in order to disaggregate the electricity consumption of the dwelling into electricity consumption profiles of each individual appliance. From these profiles, the system extracts some appliance utilization features and feeds them to the user knowledge database. In addition, some other data can be added to the user knowledge database, such as the location, the type of the building etc. Also, user interest data are stored in the user knowledge database. These

data can be extracted based on some previous history of the user or by direct questioning about the interests of the user. After that, the data in the user knowledge database are used to develop a model which includes all the inputs and outputs the recommended energy saving appliance list and suggests it to the user.

Another example of a RS that uses disaggregated electricity consumption data is the multi-agent RS proposed in [21]. The RS consists of three modules. Each of the modules has agents. Each of the agents is entitled to do a certain task in the RS. There is also a Control agent which monitors and coordinates the operation of the agents.

The first module is Device module where Device agents operate. Their task is to read the measurements from the smart devices inside the building. The communication between the Device agents and the smart devices is performed via specific communication protocols. The Device agents obtain data about the electricity consumption from the smart device with a constant frequency. Each Device agent communicates only with one smart device. Thus, the number of Device agents is defined by the number of smart devices in the building.

The second module is Crawler module. This module has the Crawler agent, the main task of which is to extract information about the prices of electricity from the web page. The agent extracts and records this information every twenty-four hours.

The third module is the Recommendation module. The operation of this module is performed by three agents: Filtering agent, Behavior agent and Recommendation agent.

The main task of the Filtering agent is to filter the data that was obtained by the Device agent and the Crawler agent. The Behavioral agent uses the data provided by the Filtering agent to extract the behavioral patter of how each device is being used by the user. The last agent in this system is the Recommendation agent. This agent receives the data provided by the Behavioral agent and generates and provides the recommendations for the user.

All the modules and agents can communicate with each other forming by that a complete RS that can make short- or long-term recommendation for the user to reduce their electricity consumption. Although this system well-designed and provides a variety of useful recommendations to the user, it has some drawbacks. These drawbacks are pointed out and discussed in more details in section 3.4.

## 3.3 Comparative summary of recommender systems

As a summary and an explanation of why disaggregating the electricity consumption data plays such a vital role in providing recommendations to the users and why this work focuses on this field in particular, Table 1 was designed to depict the existing works on electricity usage recommendations.

Table 1: Comparison of published works on electricity usage RS

| Recommender system | Paper | Type of data used | Type of recommendations |
|---|---|---|---|
| **AgentSwitch** | "Recommending Energy Tariffs and Load Shifting Based on Smart Household Usage Profiling" [32] | Both aggregated and disaggregated data | Personalized recommendations of energy tariffs & load shifting recommendations |
| **Multiagent recommendation system** | "Multi-Agent Recommendation System for Electrical Energy Optimization and Cost Saving in Smart Homes" [21] | Disaggregated data | New hours in which to use the appliances |
| **Residential Energy Usage Recommendation System (REURS)** | "Personalized Residential Energy Usage Recommendation System Based on Load Monitoring and Collaborative Filtering" [34] | Disaggregated data | Energy-efficient appliance usage plans |
| **Social Information Filtering-Based RS** | "Social Information Filtering-Based Electricity Retail Plan RS for Smart Grid End Users" [31] | Aggregated data | Retail plans |
| **Electricity plan recommender system (EPRS)** | "Collaborative Filtering-Based Electricity Plan RS" [33] | Both aggregated and disaggregated data types | Electricity retail plans |
| **PRS with Electrical Intrusive-based Recovery (EPRS-EI)** | "Electricity plan RS with electrical instruction-based recovery" [35] | Both aggregated and disaggregated data | Personal electricity plans |

| | | | |
|---|---|---|---|
| **Hybrid collaborative filtering-based electricity plan recommender system (HCF-EPRS)** | "Big Data-driven Electricity Plan RS" [36] | Both aggregated and disaggregated data | Instructions in retailer and plan selection |
| **Home Area Network (HAN)** | "Design and Evaluation of a Constraint-Based Energy Saving and Scheduling RS" [37] | Disaggregated data from smart plugs | High-level energy consumption plans, electricity usage schedules |

Based Table 1 a conclusion can be made about the vital role of obtaining the disaggregated electricity consumption data in order to build an electricity usage RS. Among all the analyzed published work only one RS was based solely on the usage of aggregated electricity consumption data. In all other cases either disaggregated data were used or both disaggregated and aggregated data. Thus, in order to develop a reliable RS, disaggregation of the aggregated electricity consumption data is necessary. Below, a discussion of two way of disaggregation: intrusive and non-intrusive is provided.

## 3.4 Intrusive disaggregation

A RS that utilizes additional devices measuring the electrical power consumption data of each device was proposed in [21]. Smart plugs were used as a cost effective and easily implemented solutions to provide electricity consumption data of each appliance. Possible recommendations were given three labels: "no recommendation" if it is impossible to implement it, "short-term recommendation" for devices that can demonstrate a limited flexibility in their usage and "long-term recommendation" for devices that can be very flexible in terms of their usage timeframes.

Although the RS developed in [21] was validated by testing, installing an individual smart plug for each device may be a burden for implementing the system. The decision regarding the number of plugs and the devices to be controlled is to be taken solely by the consumer. This may cause installing devices that are eventually not used or installing them in a way that will not contribute significantly to the reduction of electricity consumption. Apart from that, additional installations and interventions into the dwellings are necessary in order to set up the system.

## 3.5 Non-intrusive disaggregation

For the purpose of retrieving the consumption curves of each individual appliance from the aggregate consumption curve various non-intrusive algorithms can be used. Many non-intrusive algorithms were proposed and published [38]–[46].

The available data provided by smart meters usually come in form of aggregated data that represent the electricity consumption of the building or house, without any visible depletion of these data into the consumption of each individual device. In this case disaggregation of the data are necessary to obtain the electricity consumption curve of each particular device.

Non-Intrusive Load Monitoring (NILM) has supervised and unsupervised methods with published algorithms. One of the drawbacks of the supervised methods is that in order to use them some amount of already disaggregated data should be available as a training set. This might be expensive and difficult to obtain. On the other hand, the unsupervised methods are easier to implement although they are considered to be less accurate [46].

NILM can be event-based and state-based. Event-based NILM approaches are based on identifying the event windows, which are the events of switching on or off of an appliance. Each window is characterized by a power raise at the beginning and a power drop at the end of the window [47]. The state-based NILM approaches are based on representing the operation of each appliance using a state machine. These approaches normally use the Hidden Markov Model (HMM) [48].

Some other works were also published proposing NILM algorithms such as [49], where an unsupervised approach based on the nonparametric factorial hidden Markov model was proposed. Another unsupervised NILM algorithm was suggested in [50], which is an unsupervised algorithm based on graph signal processing. This algorithm will be used in this work to demonstrate how the NILM model can be embedded in the RS and provide the output for it.

# 4 Methodology

This chapter describes the working principle of the proposed RS. Based on the analysis of published works, the proposed RS contains a part to disaggregate the whole-house electricity consumption data, using the output to provide recommendations regarding the electrical appliances' usage.

## 4.1 Cold-start problem

The suggested system is based on the CF principle. In order to implement a CF algorithm, the similarity between the target user and other users should be measured. Normally, this can be done using the ratings that the target user has given to the recommendations before and the ratings that other users have given to these recommendations. After that, users whose ratings were similar to the ratings of the target user for the same recommendations, are considered to be the neighbors for the target user. The recommendations, which were highly rated by neighbors but were not introduced to the target user before, are then suggested to the target user.

However, in the case where there are no data about previous ratings of the target user, other data can be used to determine the similarity between users. This situation is typically classified as a the "cold-start problem" in the RS.

The proposed RS tackles the cold start problem. Taking into account that the provided data include only the aggregated electricity consumption reading from the smart meter installed in the building and that there is no additional data that can be used to develop the RS, the cold start problem in this case forms a major issue that should be tackled. Below a more detailed explanation of the cold start problems encountered when developing RS are presented.

According to the literature [51] cold start problems can be classified into three categories, namely:

- *New community:* This problem is often faced when a new RS is being created. New community problem incorporates in itself both new item problem and new user problem at the same time. The reason of this problem is the lack of previously obtained information or data that could be relied on for making recommendations.

- *New item:* This problem occurs when a new item is presented to the RS. In this case the item does not have any votes yet, which causes that the system cannot recommend it to anyone. Although in the case that the item has a few votes and the system is already able to recommend this item, the quality of these recommendations will be doubtful [52].
- *New user:* This problem occurs when a new user decides to use the system. When the user has not yet provided any votes, it is difficult to provide personalized recommendations to the user. Even after the user has provided a few votes, the RS cannot provide very reliable personalized recommendations until the number of votes provided by the new user will be enough to do that.

From the above classification, the new user problem is tackled in the suggested system. To do so, the neighbors of the users are determined not by previously given ratings, but based on some basic information about the buildings. This information is selected as the factors that have an influence on the ratings of the recommendations by each user.

## 4.2 Factors influencing recommendation ratings

Many external factors have a major influence on the electricity consumption behavior in buildings. These factors also have a major contribution to the answers of many questions such as “How applicable a certain recommendation is for this particular building?”, “How effective the application of a certain recommendation is for this particular building?” etc. Which affects the ratings that each user attributes to each recommendation. Which in turn influences the predicted rating of the target user for each recommendation. Thus, including these factors in the RS and using them as parameters determining the target user’s neighborhood as an approach to address the new user cold start problem in CF is a reasonable solution.

There are many factors influencing the ratings of the electricity consumption recommendations. Such as climate, building related characteristics (age, size, envelop fabrics etc.), occupancy rate, socio-economic characteristics (education, culture, income etc.) [53] and many more. For developing the RS, the factors that are the easiest to obtain were used, but other factors can be added as well. The factors used to determine user neighborhoods can be distinguished as follows:

- *Location (country):* Different countries may have different electricity pricing policies. Also, the habits, mentality and even the predominating religion in the

country can affect how people consume electricity and what electricity consumption recommendations they would prefer the most. For example, the working hours vary from country to country, which affect the electricity consumption curve in the offices.

- *Climate* [53]*:* The climate or climatic location in which the dwelling is located influences the needs of the dwelling in cooling and heating. In case of the usage of electrical appliances for these purposes, this factor affects user ratings of the suggested electricity consumption recommendations.
- *Type:* Type of dwelling refers to the type of activities that are mainly practiced there. For example, this can be a library, restaurant, canteen, residential house etc.

## 4.3 Description of the system

The system focuses on providing recommendations targeting to reduce the level of electricity consumption and to reduce the cost of the energy bills paid by the customer. The schematic representation of the RS is depicted on Figure 5.

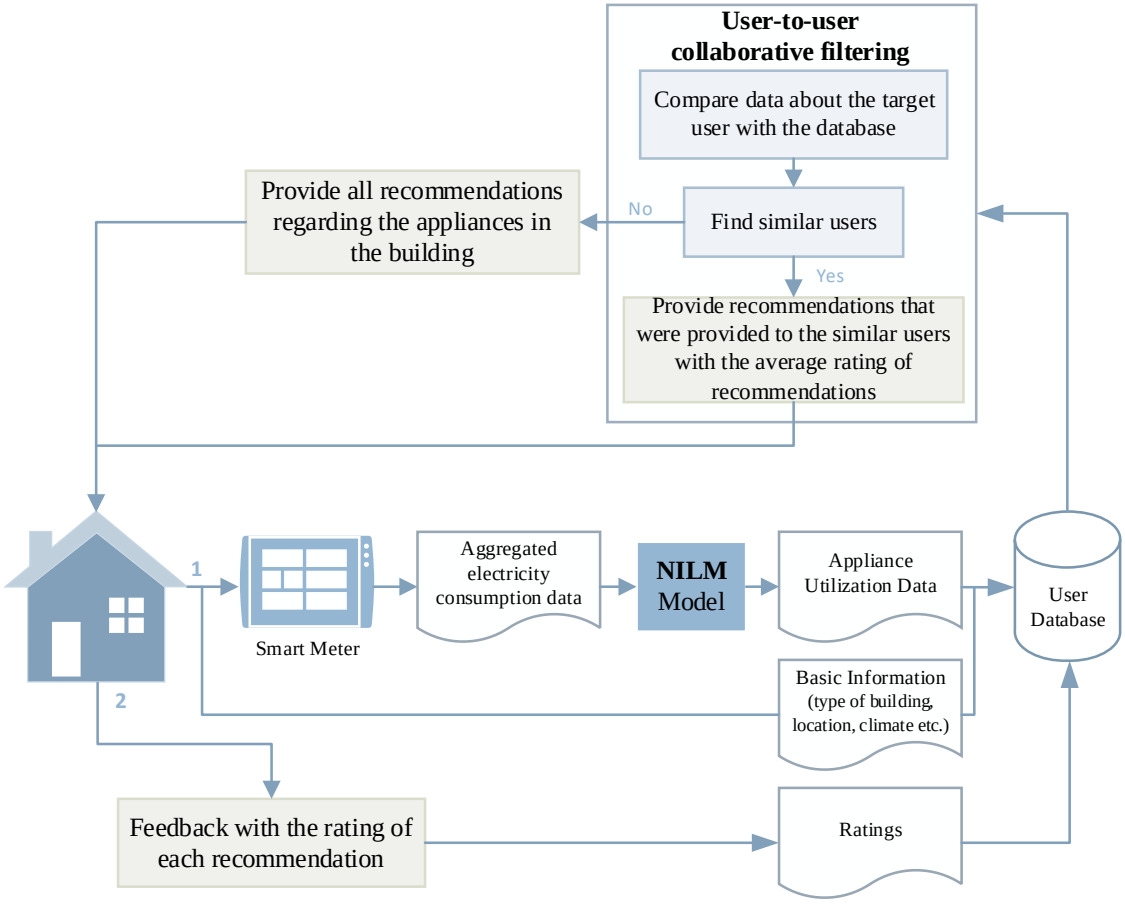



Figure 5: Schematic representation of the proposed RS

The proposed RS is based on the principle of CF type of RS. First, two types of data are collected from the user:

- Aggregated electricity consumption data, extracted from the smart meter installed in the building.
- Basic information about the user, such as the location of the building (country, city etc.), its type (residential house, school, office etc.), the climate where the building is situated and other features that can affect electric energy consumption patterns.

After the aggregated electricity consumption data are collected, they are used as an input for the NILM model that disaggregates the data and provides information about what appliances are used inside the building forming the appliance utilization data.

After that, the appliance utilization data together with the basic information data are fed into the user database.

After the user database is formed, the CF principle of the RS is applied. The system searches for users that have the highest similarity to the target user, based on their basic information. After the closest neighboring users are found, the system takes the recommendations that were given to these users and filters them to exclude recommendations regarding appliances that are not present in the target user building. Then, for each recommendation, the average value of the given ratings is computed. Finally, the recommendations are given to the target user sorted in descending order of rating values.

The last step of the system is getting feedback from the target user with their rating of the provided recommendations. These rating are added to the user building profile and this profile, including the basic information about the user and the feedback, are added to the database, which improves the accuracy of recommendations in the future.

## 4.4 NILM model

From the schematic representation of the system (Figure 5) it can be noticed that in order to disaggregate the whole-house electrical consumption values and find out what electrical appliances are being used, a NILM model is being implemented. Below a detailed explanation of the NILM model algorithm is provided. The used algorithm was developed and proposed in [50].

This NILM algorithm was selected due to a number of reasons:

- It does not require the presence of a training dataset. Typically, NILM algorithms require a training dataset to be trained on before running the algorithm with the test dataset. This training dataset is formed using the disaggregated data from the buildings that are similar to the target building. However, the useful feature of this algorithm is that it can be run without the need to be previously trained on similar data. This feature is useful when addressing the condition of limited input data.
- The algorithm is not specific to a certain type of buildings i.e., it can be used for residential buildings as well as for schools, offices and other common types of buildings. This is due to the fact the algorithm labels the discovered appliances on the very last stage of the disaggregation. Thus, most of the common appliances can be labeled with the aid of a proper signature database.

Following a detailed description of the algorithm is provided [50]. In order to simplify the understanding of the working principle of the algorithm it is explained on a simple example first. Afterwards, a more detailed description is provided.

### 4.4.1 Simplified explanation

This is a demonstration of the algorithm on a simple example that was done in order to summarize and clarify the working principle of the algorithm.

The algorithm takes as an input two data files: aggregated electrical power consumption and signature database.

The aggregated electrical power consumption curve is depicted in Figure 6.

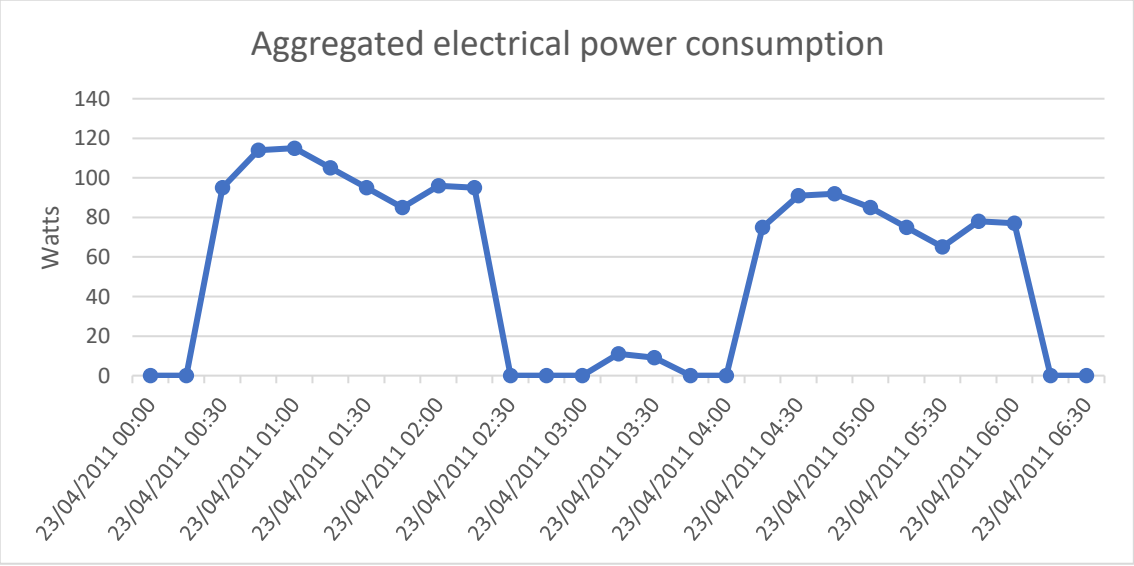



Figure 6: Aggregated electricity consumption curve for the explanatory example

The second file is the file containing signatures of electrical appliances. These appliances may or may not be present in the building. In this example the signature database consists of signatures of three appliances labelled as refrigerator, electrical heater and microwave. The signatures of each of the appliances are depicted on Figure 7, Figure 8 and Figure 9 respectively.

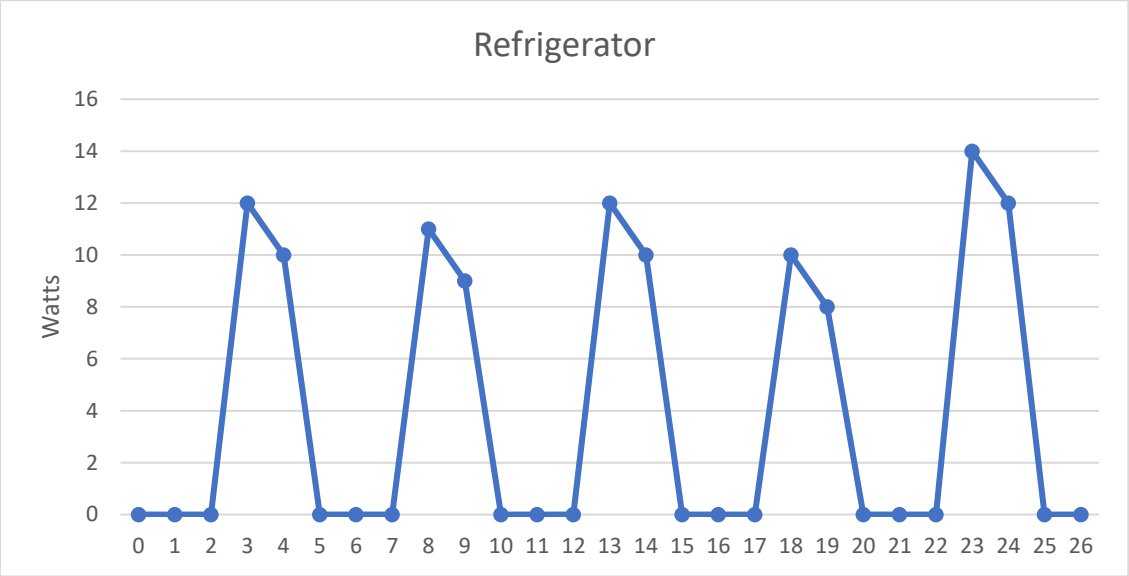



Figure 7: Signature of "Refrigerator" for the explanatory example

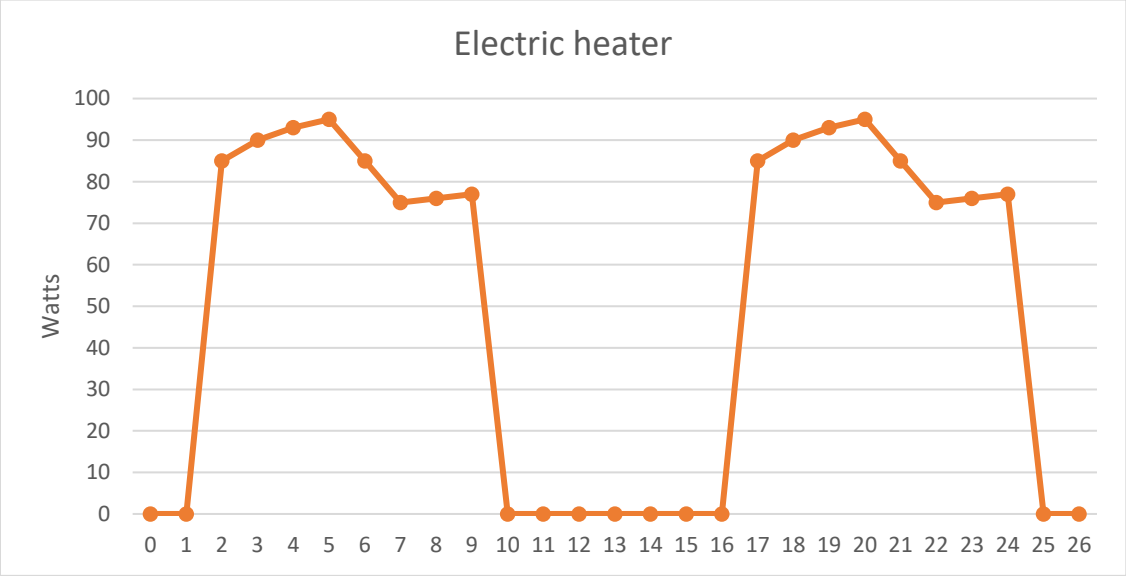



Figure 8: Signature of "Electric heater" for the explanatory example

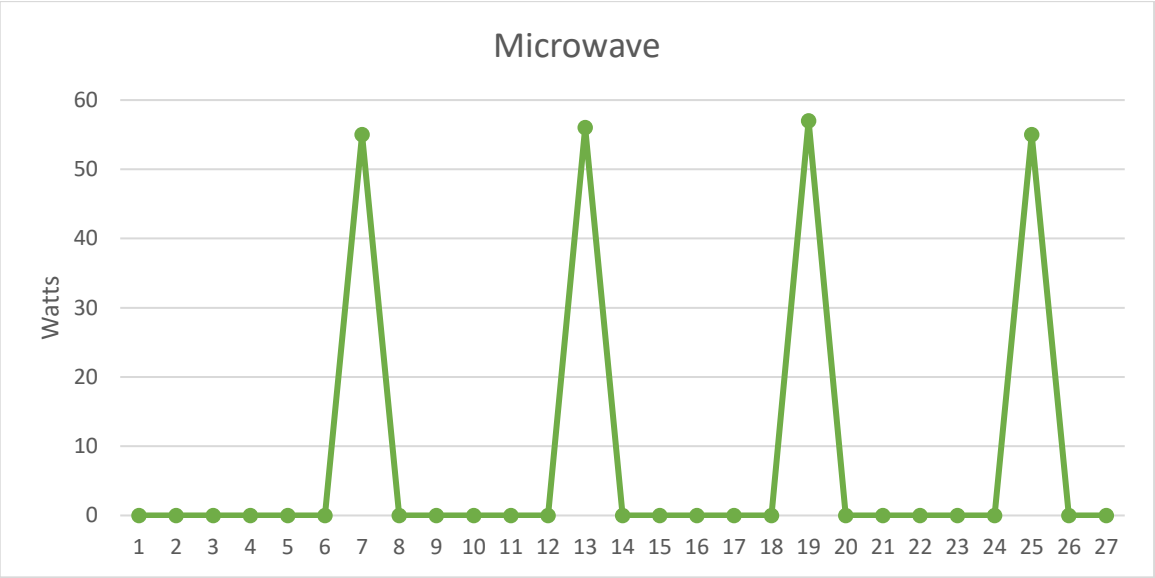



Figure 9: Signature of "Microwave" for the explanatory example

First, the algorithm identifies the increasing and decreasing power edges. This is represented on Figure 10.

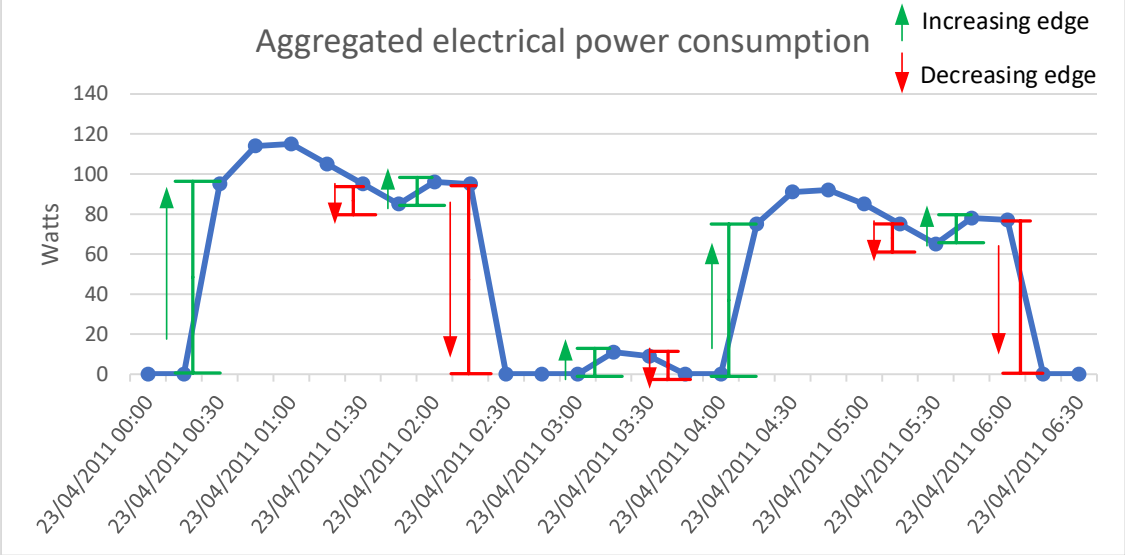



Figure 10: Increasing and decreasing power edges for the explanatory example

After that, the algorithm forms two clusters: one with increasing edges called positive cluster and one with decreasing edges called negative cluster (Figure 11).

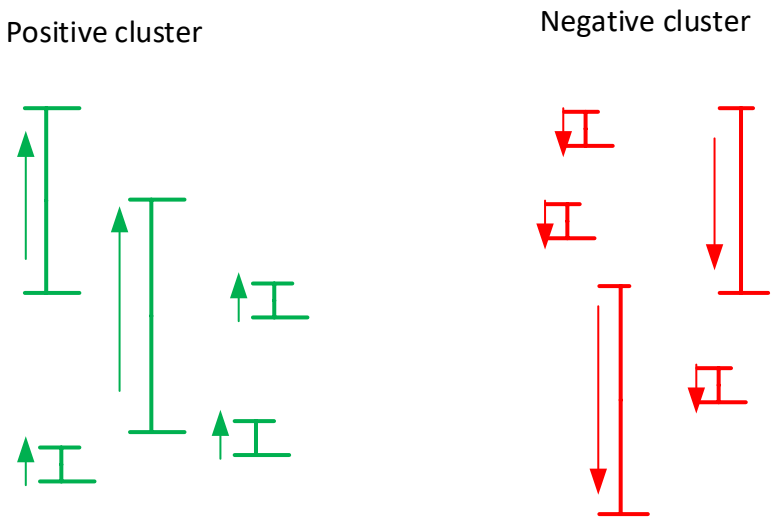



Figure 11: Positive and negative clusters for the explanatory example

After that, depending on the magnitude and the time between these edges the algorithm finds the best match for each edge from the positive cluster with an edge from the negative cluster as depicted on Figure 12.

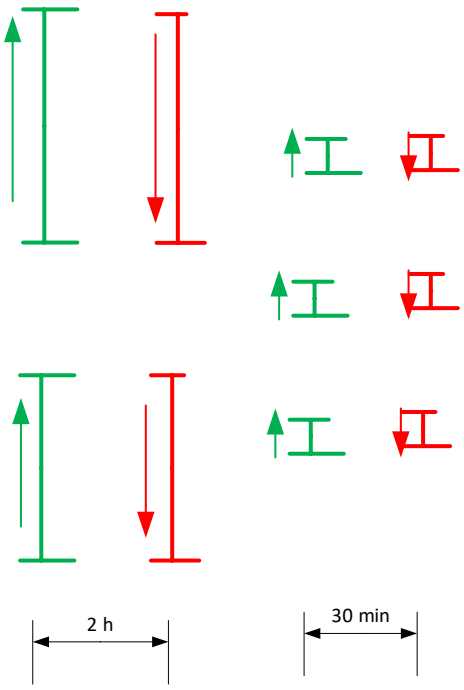



Figure 12: Edge pairs for the explanatory example

The final step is label matching. At this step the algorithm matches the pairs that were found with appliance signatures from the signature database. The pair matching the electric heater and refrigerator are represented on Figure 13 and Figure 14.

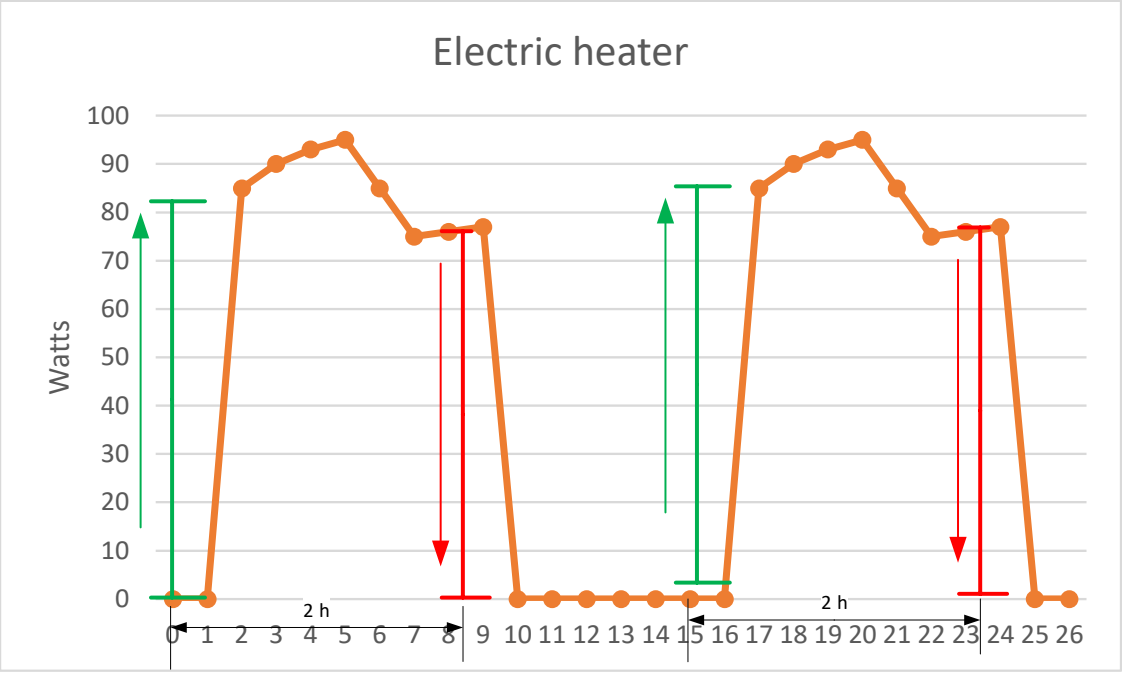



Figure 13: Label matching for electric heater for the explanatory example

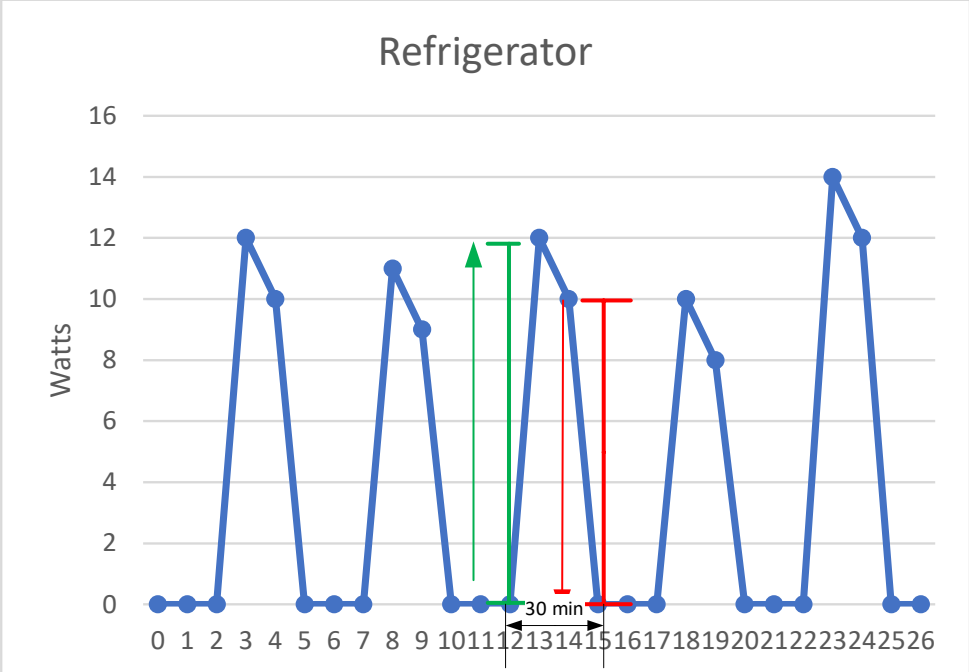



Figure 14: Label matching for refrigerator for the explanatory example

It can be noted that in this example there were no match for the edges with the signature of the microwave. Thus, it is defined that there is no microwave in this building. By repeating the label matching for each of the discovered edge pairs the disaggregated electrical power consumption curves can be outputted (represented on Figure 15).

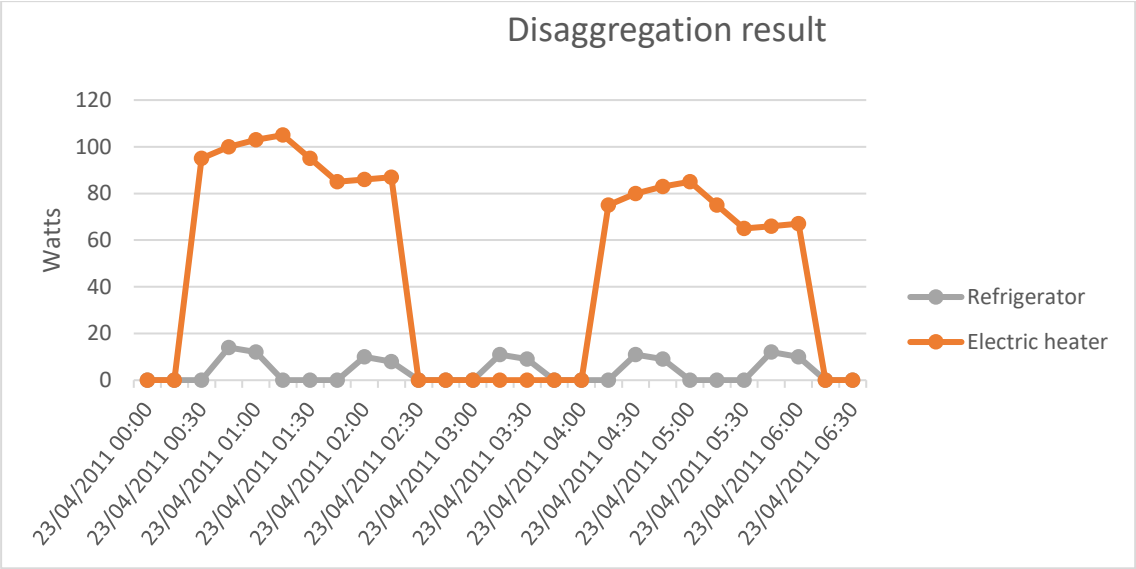



Figure 15: Disaggregation result for the explanatory example

### 4.4.2 Detailed description

First of all, the task that has to be solved by the algorithm is represented by [50] (1).

$$P_{t_i} = \sum_{m \in \mathrm{M}} P_{mt_i} + n_{t_i} \qquad [50] \quad (1)$$

where:

$P_{t_i}$ – total measured active power of at each time instance $t_i$, where $i$ = 1, …, n;

$P_{mt_i}$ – contribution of power of each appliance $m$ into the total measured power;

M – set of the known appliances in the building;

$n_{t_i}$ – noise including the unknown appliances and the base load.

So, the task of the algorithm can be simplified to finding $P_{mt_i}$ according to [50] (2).

$$\min_{P_{mt_i}} \left| P_{t_i} - \sum_{m \in \mathrm{M}} P_{mt_i} \right| \qquad [50] \quad (2)$$

It is an event-based algorithm. The principle of this algorithm is first finding the window of events. This means that it finds the periods of time when each appliance changes its state from on to off and vice versa. After that, using a database that includes signatures of appliances that were formed by other people in the past or by any other means, it matches each device with its signature that is available in the database. In order to find the event windows, this approach uses Graph Signal Processing (GSP), that is obtaining graph signals from indexing a dataset by nodes of a graph. In this work the NILM

approach used is based on GSP. In other words, the data are represented in the form of a graph and its adjacency matrix.

A graph G = {V, A} is built using a set of measurements x, where each node $v_i \in V$ represents one active power measurement, while $A$ is the adjacency matrix that represents the edges of the graph. After that, the set of nodes $V$ is mapped to a set of complex numbers, which represent the graph signal $s$. Each of the elements $s_i$ is indexed by one node $v_i \in V$.

The flowchart of the algorithm of the NILM model used is represented in Figure 16 [50].

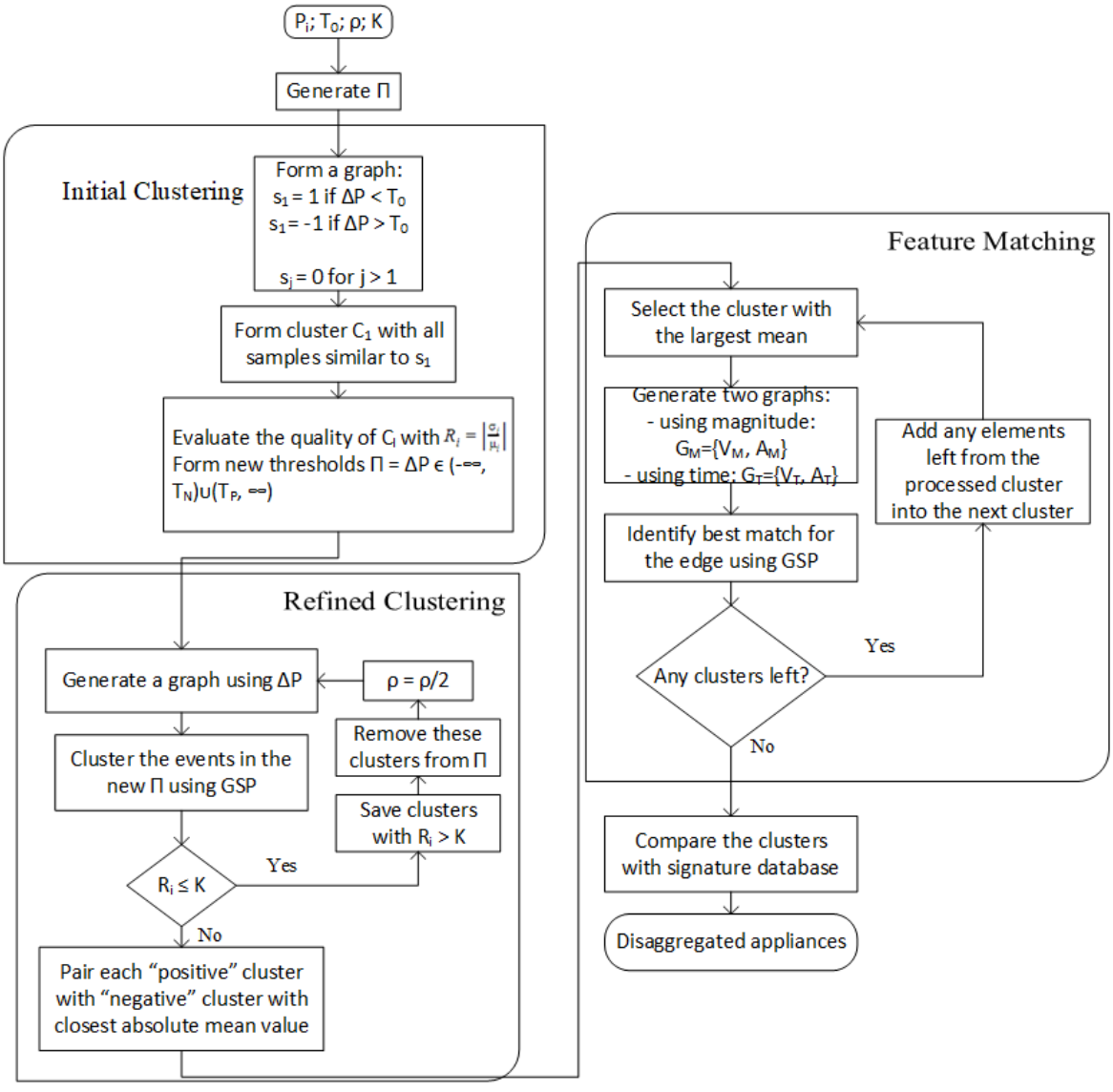



Figure 16: Flowchart of the NILM model [50]

The algorithm consists of three main steps: edge detection, clustering and feature matching. Clustering step is performed twice. First, the input data are fed to the algorithm. It is the aggregated power consumption $P_{t_i}$, initial threshold $T_0$, ρ is the scaling factor and $K$.

After that, comes the edge detection step, where using the threshold $T_0$ and applying the condition $\Delta P_i \in (-\infty, -T_0) \cup (T_0, \infty)$ the set of values Π is generated. Π represents all possible events that happened in the dataset. An event is defined as switching on/off an appliance or changing the mode of operation.

The first clustering step is initial clustering where the graph is built. In order to make the graph, the values from previously generated are used. Each value is a node $v_i$ of the graph. At the beginning the value of $s_1$ is set depending on the value of $\Delta P_1$. If $\Delta P_i > T_0$, then $s_1 = 1$. If $\Delta P_i < T_0$, then $s_1 = -1$. All the remaining $s_j$ for $j > 1$ are set to 0.

All samples that are statistically similar to $s_1$ are clustered. To perform the clustering $s^*$ is calculated using [50] (3):

$$s^* = L^{\#}_{2:N,2:N}(-s_1)L^T_{1,2:N} \qquad [50] \quad (3)$$

where:

$s^*$ – the smoothness optimization solution (minimizes the total graph variation);

$(.)^{\#}$ – the pseudo-inverse matrix;

$N$ – length of the discrete signal $s$;

$L$ – the graph Laplacian operator ($L = D - A$), where $D$ – diagonal matrix with nonzero entries $D_{i,i} = \sum_j A_{ij}$.

A constant value of $q$ is defined and fixed. If $s_j^* > q$s, then $\Delta P_j$ with $s_1$ is added to the first cluster of events and removed from Π. That is how the cluster $C_1$ is formed. After that this procedure is repeated with the remaining elements of Π forming clusters $C_i$ until the set Π is empty. Each cluster $C_i$ will have either only positive or only negative edges.

After that, the quality of clusters $C_i$ is evaluated by calculating [50] (4):

$$R_i = \left|\frac{\sigma_i}{\mu_i}\right| \qquad [50] \quad (4)$$

where:

$R_i$ – relative standard deviation (RSD);

$\sigma_i$ – standard deviation of cluster $C_i$;

$\mu_i$ – mean value of cluster $C_i$.

The lower the value of $R_i$, the better the quality of the cluster $C_i$.

The final process of the initial clustering step is defining new thresholds. Two clusters with the highest RSD are selected. The mean values of these two clusters define the new thresholds $T_P$ and $T_N$ that are used for positive and negative edges respectively.

The second clustering step is refined clustering. In this step all clusters that have $RSD > K$ ($K$ is heuristically obtained constant which represents the acceptable precision level of a cluster) are re-clustered by dividing $\rho$ into 2 in [50] (5).

$$A_{i,j} = exp\left\{-\frac{\left|dist(x_i, x_j)\right|^2}{\rho^2}\right\} \qquad [50] \quad (5)$$

where:

$\rho$ – scaling factor;

$dist(x_i, x_j)$ – can be Euclidean distance.

After every iteration all clusters which have $RSD \leq K$ are removed from the set Π and saved as final clusters. Clustering process runs until the set Π has no more elements to cluster.

After that, small clusters can be merged into bigger clusters to ensure that the number of clusters with increasing power edges equals the number of clusters with decreasing power edges.

The next step of the algorithm starts after all clusters are formed and Π becomes empty. This step is called Feature Matching. In this step each "positive" cluster is paired with a "negative" cluster. The pairs are formed by matching clusters with closest absolute mean values.

After that, each cluster pair is processed separately. Each element from the "positive" cluster is matched with an element from the "negative" cluster. To do that the time intervals between the edges are used in addition to the magnitude difference.

First of all, the cluster with the largest mean is taken. If $C_P$ and $C_N$ are two paired "positive" and "negative" clusters respectively, then for each $C_{P_i} \in C_P$ a match $C_{N_i} \in C_N$ should be found. Since the decreasing edge comes after the increasing edge when looking for a pair for $C_{P_i}$ a graph containing only $C_N$ that come after $C_{P_i}$ and before $C_{P_{i+1}}$ is formed. This set of $C_N$ is denoted as Φ . The set $\Phi_M$ will be a set of magnitude differences between $C_{P_i}$ and each $i$. The set $\Phi_T$ will be a set of time intervals between $C_{P_i}$ and each $i$.

After that GSP is applied to form two graphs:

- $G_M = \{V_M, A_M\}$, where $\Phi_M$ is used to form the nodes and $A_{M_{i,j}} = exp\left\{-\frac{\left|dist(\Phi_{M_i}, \Phi_{M_j})\right|^2}{\rho^2}\right\}$ and the graph signal $s_M$ is formed that $s_{M_1}$ is the average value of the elements in $\Phi_M$ and $s_{M_j} = 0$ for $j > 1$.
- $G_T = \{V_T, A_T\}$, where $\Phi_T$ is used to form the nodes and $A_{T_{i,j}} = exp\left\{-\frac{\left|dist(\Phi_{T_i}, \Phi_{T_j})\right|^2}{\rho^2}\right\}$ and the graph signal $s_T$ is formed that $s_{T_1}$ is the median value of the elements in $\Phi_T$ and $s_{T_j} = 0$ for $j > 1$.

Then [50] (3) is calculated for each of the two graphs resulting in $s_M^*$ and $s_T^*$. The decreasing edge that would be the best match for the increasing edge $C_{P_i}$ is calculated using [50] (6).

$$\arg\max_i\{\alpha s_{M_i}^* + \beta s_{T_i}^*\} \qquad \text{[50] (6)}$$

where:

$i = 1, \dots, n$;

$n$ – length of $s_M^*$ and $s_T^*$ (number of candidates);

$\alpha$ and $\beta$ – chosen heuristically;

$\alpha$ – weight given to magnitude;

$\beta$ – weight given to time.

The result of this calculation gives the best matching edge.

After all edges in the cluster pair are matched, if there are still some edges that were not paired from this cluster pair, these edges are included in the next cluster. This process is carried on until all clusters are paired.

The last stage of the algorithm is generating the output. Each matched pair of clusters from the previous procedure defines one potential appliance state. Each matched sample represents the appliance running event. Each event is compared with the signature database. This is how the appliance is labeled. If there is no match with the signature database, it can be added to the database. The customer can also add labels to the database.

The main advantage of the algorithm is that it does not require training data and it can also be performed without any information about what appliances are present inside the

building. It just matches the output with the signature database which can contain any devices and if for some devices no match could be found, the user can suggest his own labels too.

At this point the information about what appliances are present in the building is obtained. After that, this information is used as an input to the CF model that comes next.

## 4.5 Collaborative-filtering model

After the information about the appliances in the building is obtained, these data are fed to the CF model to obtain the suitable recommendations for the user. The schematic representation of the model of the RS is depicted in Figure 17.

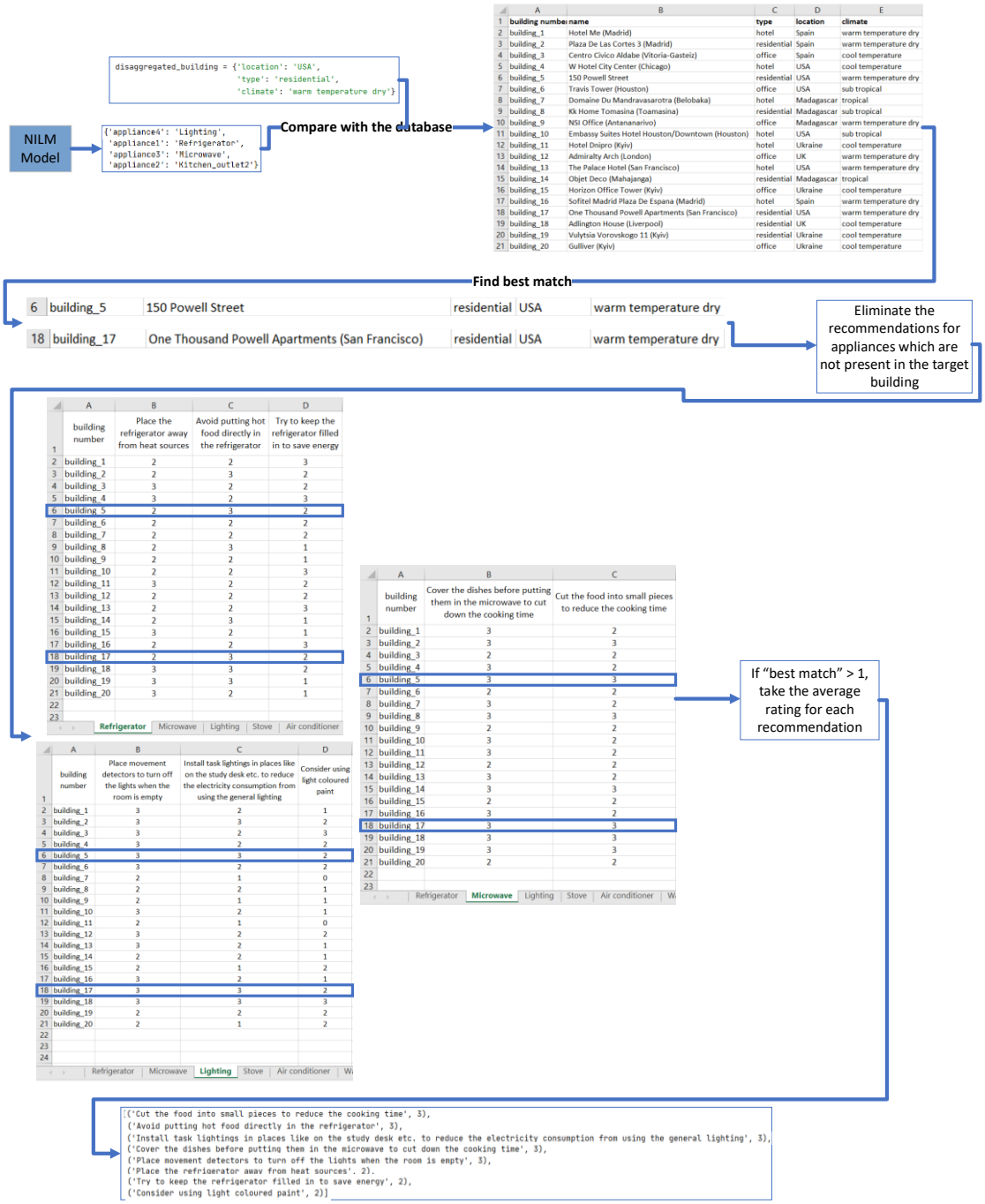



Figure 17: Schematic representation of the CF model

After the disaggregation is done, NILM model outputs the list of appliances that are present in the building. This serves as an input to the CF model. Apart from that, the basic information about the building including its location, type and climate is also an input to the model as shown in Figure 17.

After that, the data are compared with the database of the buildings to find the closest match for the target building. The matches are determined based on the basic information provided about the buildings. So, the algorithm first tries to find buildings that match the

target building in all of the three parameters i.e., they have the same location, type and climate. If the algorithm finds such buildings, then these are considered the best match for the target building. Otherwise, the algorithm tries to find buildings with two matching features i.e., having the same location and type, but different climate or the same location and climate, but different type or the same type and climate but different location. If such buildings are found, the algorithm considers these buildings to be the closest match.

In case if no such buildings are found, the algorithm tries to find buildings with one matching feature. It can be either the same location, type or climate. Then, it considers these buildings to be the best match. In case no building is found with even one matching feature, the recommendations are filtered by eliminating the recommendations for appliances that are not present in the building. After that, all recommendations concerning the appliances in the building are given directly to the user.

This algorithm of finding the best matching buildings can be mathematically expressed by calculating the cosine similarity between the target building and each of the buildings in the database. The cosine similarity is calculated using the formula (7).

$$sim(x, y) = \frac{x \cdot y}{\|x\|\|y\|} \quad [54] \quad (7)$$

where:

$x$ and $y$ – two vectors ($x = (x_1, x_2, \dots, x_n)$ and $y = (y_1, y_2, \dots, y_n)$);

$\|x\|$ and $\|y\|$ – Euclidean norm ($\sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$ and $\sqrt{y_1^2 + y_2^2 + \cdots + y_n^2}$ ) of the vectors $x$ and $y$ respectively;

In order to clarify the application on the buildings database more a simple example is given. Let's assume Table 2 represents the database of the buildings in the RS. The database consists of 6 buildings. The building which is named "target" represents the target building for which the recommendations are being generated.

Table 2: Buildings database for the explanatory example of the RS

| Building number | UK | Spain | Ukraine | Warm | Cold | Hotel | Residential | Office | Similarity with target building |
|---|---|---|---|---|---|---|---|---|---|
| **1** | 1 | | | | 1 | 1 | | | 1 |
| **2** | | 1 | | 1 | | 1 | | | $0.\overline{3}$ |
| **3** | | 1 | | | 1 | 1 | | | $0.\overline{6}$ |
| **4** | 1 | | | | 1 | | | 1 | $0.\overline{6}$ |
| **5** | | 1 | | 1 | | | 1 | | 0 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **6** | | 1 | | 1 | | | | 1 | 0 |
| **target** | 1 | | | | 1 | 1 | | | n/a |

The cosine similarity is measured between the target building and each of the buildings from the database. The calculations for each of the buildings are done as follows:

$$sim(target, 1) = \frac{1 \cdot 1 + 1 \cdot 1 + 1 \cdot 1}{\sqrt{1^2 + 1^2 + 1^2} \cdot \sqrt{1^2 + 1^2 + 1^2}} = 1$$

$$sim(target, 2) = \frac{1 \cdot 1}{\sqrt{1^2 + 1^2 + 1^2} \cdot \sqrt{1^2 + 1^2 + 1^2}} = 0.\overline{3}$$

$$sim(target, 3) = \frac{1 \cdot 1 + 1 \cdot 1}{\sqrt{1^2 + 1^2 + 1^2} \cdot \sqrt{1^2 + 1^2 + 1^2}} = 0.\overline{6}$$

$$sim(target, 4) = \frac{1 \cdot 1 + 1 \cdot 1}{\sqrt{1^2 + 1^2 + 1^2} \cdot \sqrt{1^2 + 1^2 + 1^2}} = 0.\overline{6}$$

$$sim(target, 5) = \frac{0}{\sqrt{1^2 + 1^2 + 1^2} \cdot \sqrt{1^2 + 1^2 + 1^2}} = 0$$

$$sim(target, 6) = \frac{0}{\sqrt{1^2 + 1^2 + 1^2} \cdot \sqrt{1^2 + 1^2 + 1^2}} = 0$$

From the results a conclusion can be made that the buildings with the highest values of similarity to the target building are the neighbouring buildings. In this example the best matching building for the target building is building number 1 because its similarity with the target equals 1. In the case if building 1 was not in the database, the neighbouring buildings would be number 3 and 4 since they have the second biggest value of similarity with the target building.

In case matching buildings are found, the recommendations given to these buildings are filtered too, by eliminating the recommendations given for appliances that are not present in the target building.

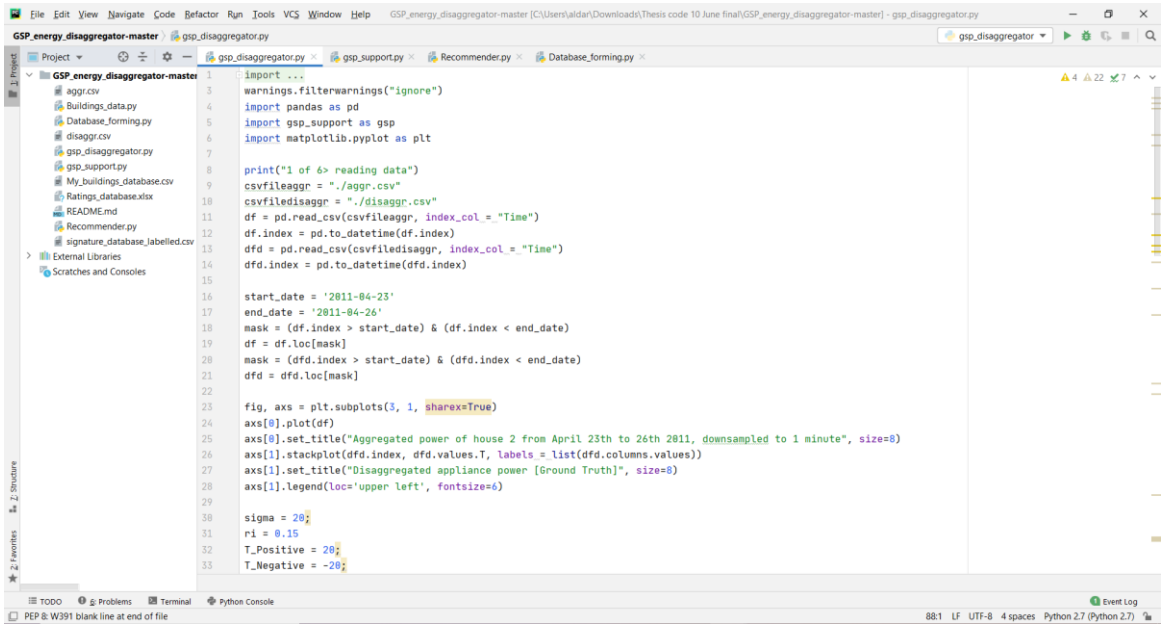
After that, if the number of matching buildings is more than one, then for each recommendation an average rating is calculated. The average rating is the sum of the ratings given to a particular recommendation by the matching buildings divided by the number of the matching buildings.

Then, the resulting recommendations are sorted in descending order based on their ratings, and displayed. For now, the system shows all the resulting recommendations in descending order. However, it is possible to impose a threshold that will filter the

recommendations with rating below the threshold and display only the recommendations with higher average rating.

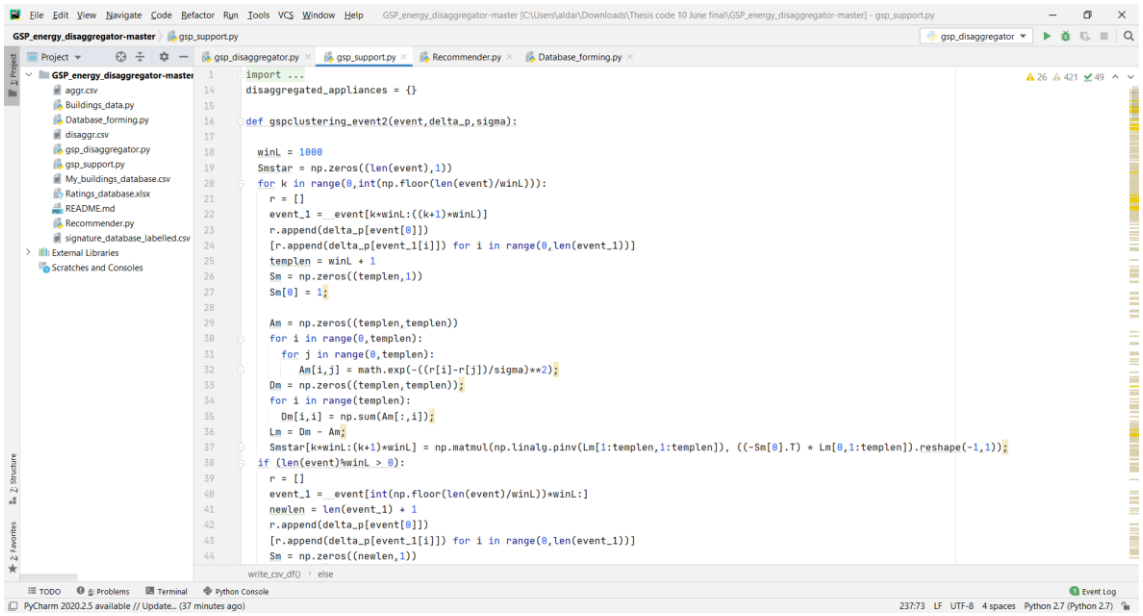
It should be noted that in this algorithm in order to measure the distance between the buildings and to find the neighboring buildings only the parameters in the basic information were used. At this point the output of the NILM algorithm containing the data about the appliances that are present in the building is only used to filter the recommendations and eliminate the irrelevant once for this particular building. The NILM output is not used to find the neighboring buildings. This is due to the fact that the weight that each matching appliance would contribute to the similarity distance between two buildings would be considerably less than the weight of the contribution of each of the basic information parameters. So, for example, there is more probability that two residential buildings will have similar preferences regarding the recommendations than that two buildings with microwaves but different building types will have similar recommendation preferences. Thus, for the NILM output to be considered in defining the neighboring buildings a weighting scale should be introduced that will reduce the weight of the matching of each of the appliances. For now, the NILM output was only used to filter out the irrelevant recommendations. However, taking into account the NILM output when defining the neighboring buildings is something that should be definitely taken into account for the future improvement of the system.

The whole RS including the NILM model was written in Python 2.7 in PyCharm IDE. The NILM model code consists of “gsp_disaggregator.py” and “gsp_support.py” files. The recommender part itself consists of “Recommender.py” and “Database_forming.py” files. The screenshots from the PyCharm IDE with the NILM model code and the recommendation part code are represented in Figure 18, Figure 19, Figure 20 and Figure 21.



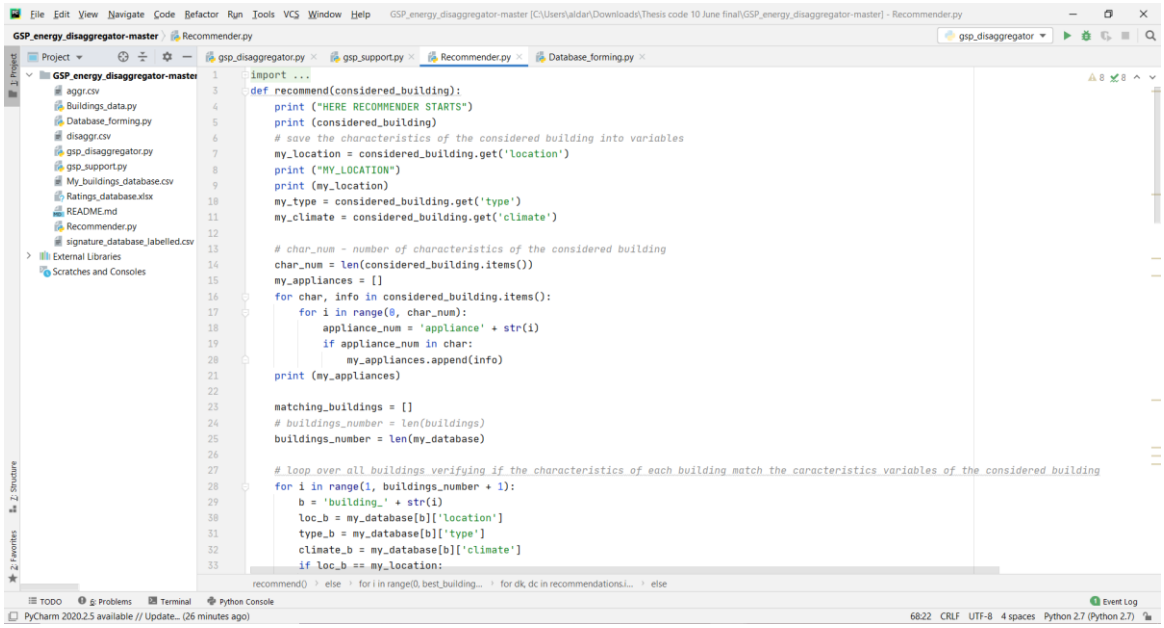
```
import ...
warnings.filterwarnings("ignore")
import pandas as pd
import gsp_support as gsp
import matplotlib.pyplot as plt

print("1 of 6> reading data")
csvfileaggr = "./aggr.csv"
csvfiledisaggr = "./disaggr.csv"
df = pd.read_csv(csvfileaggr, index_col = "Time")
df.index = pd.to_datetime(df.index)
dfd = pd.read_csv(csvfiledisaggr, index_col = "Time")
dfd.index = pd.to_datetime(dfd.index)

start_date = '2011-04-23'
end_date = '2011-04-26'
mask = (df.index > start_date) & (df.index < end_date)
df = df.loc[mask]
mask = (dfd.index > start_date) & (dfd.index < end_date)
dfd = dfd.loc[mask]

fig, axs = plt.subplots(3, 1, sharex=True)
axs[0].plot(df)
axs[0].set_title("Aggregated power of house 2 from April 23th to 26th 2011, downsampled to 1 minute", size=8)
axs[1].stackplot(dfd.index, dfd.values.T, labels = list(dfd.columns.values))
axs[1].set_title("Disaggregated appliance power [Ground Truth]", size=8)
axs[1].legend(loc='upper left', fontsize=6)

sigma = 20;
ri = 0.15
T_Positive = 20;
T_Negative = -20;
```

Figure 18: Screenshot of the PyCharm IDE containing the code of the “gsp_disaggregator.py” file



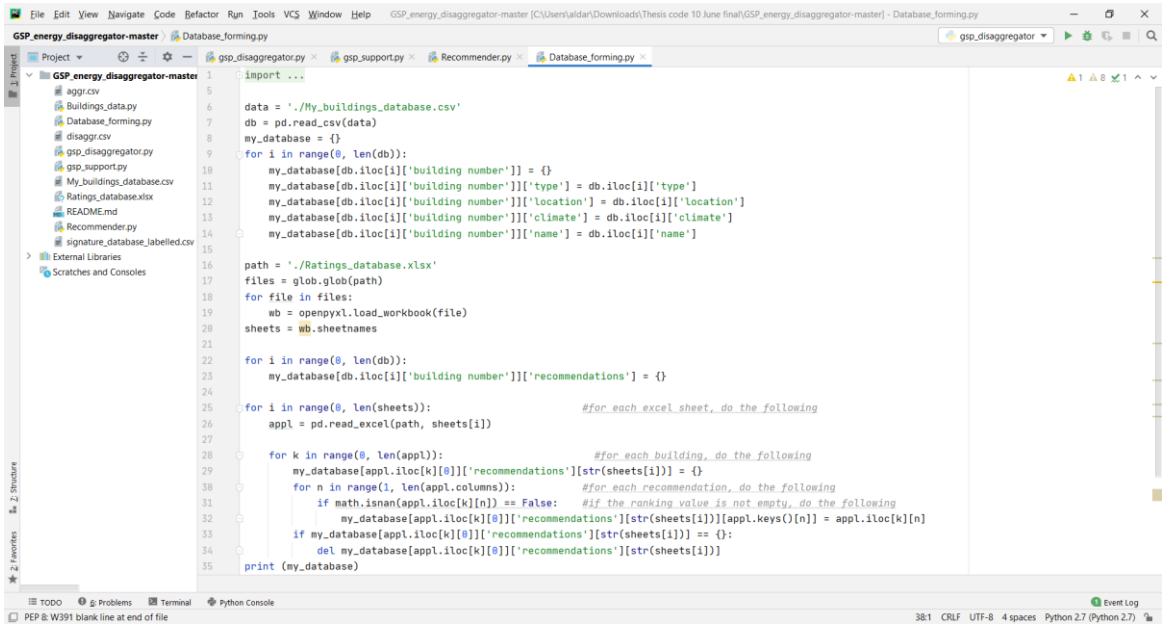
```
import ...
disaggregated_appliances = {}

def gspclustering_event2(event,delta_p,sigma):

  winL = 1000
  Smstar = np.zeros((len(event),1))
  for k in range(0,int(np.floor(len(event)/winL))):
    r = []
    event_1 = event[k*winL:((k+1)*winL)]
    r.append(delta_p[event[0]])
    [r.append(delta_p[event_1[i]]) for i in range(0,len(event_1))]
    templen = winL + 1
    Sm = np.zeros((templen,1))
    Sm[0] = 1;

    Am = np.zeros((templen,templen))
    for i in range(0,templen):
      for j in range(0,templen):
        Am[i,j] = math.exp(-((r[i]-r[j])/sigma)**2);
    Dm = np.zeros((templen,templen));
    for i in range(templen):
      Dm[i,i] = np.sum(Am[:,i]);
    Lm = Dm - Am;
    Smstar[k*winL:(k+1)*winL] = np.matmul(np.linalg.pinv(Lm[1:templen,1:templen]), ((-Sm[0].T) * Lm[0,1:templen]).reshape(-1,1));
  if (len(event)%winL > 0):
    r = []
    event_1 = event[int(np.floor(len(event)/winL))*winL:]
    newlen = len(event_1) + 1
    r.append(delta_p[event[0]])
    [r.append(delta_p[event_1[i]]) for i in range(0,len(event_1))]
    Sm = np.zeros((newlen,1))
```

Figure 19: Screenshot of the PyCharm IDE containing the code of the “gsp_support.py” file



```
import ...
def recommend(considered_building):
    print ("HERE RECOMMENDER STARTS")
    print (considered_building)
    # save the characteristics of the considered building into variables
    my_location = considered_building.get('location')
    print ("MY_LOCATION")
    print (my_location)
    my_type = considered_building.get('type')
    my_climate = considered_building.get('climate')

    # char_num - number of characteristics of the considered building
    char_num = len(considered_building.items())
    my_appliances = []
    for char, info in considered_building.items():
        for i in range(8, char_num):
            appliance_num = 'appliance' + str(i)
            if appliance_num in char:
                my_appliances.append(info)
    print (my_appliances)

    matching_buildings = []
    # buildings_number = len(buildings)
    buildings_number = len(my_database)

    # loop over all buildings verifying if the characteristics of each building match the caracteristics variables of the considered building
    for i in range(1, buildings_number + 1):
        b = 'building_' + str(i)
        loc_b = my_database[b]['location']
        type_b = my_database[b]['type']
        climate_b = my_database[b]['climate']
        if loc_b == my_location:
```

Figure 20: Screenshot of the PyCharm IDE containing the code of the “Recommender.py” file



```
import ...

data = './My_buildings_database.csv'
db = pd.read_csv(data)
my_database = {}
for i in range(0, len(db)):
    my_database[db.iloc[i]['building number']] = {}
    my_database[db.iloc[i]['building number']]['type'] = db.iloc[i]['type']
    my_database[db.iloc[i]['building number']]['location'] = db.iloc[i]['location']
    my_database[db.iloc[i]['building number']]['climate'] = db.iloc[i]['climate']
    my_database[db.iloc[i]['building number']]['name'] = db.iloc[i]['name']

path = './Ratings_database.xlsx'
files = glob.glob(path)
for file in files:
    wb = openpyxl.load_workbook(file)
sheets = wb.sheetnames

for i in range(0, len(db)):
    my_database[db.iloc[i]['building number']]['recommendations'] = {}

for i in range(0, len(sheets)):                    #for each excel sheet, do the following
    appl = pd.read_excel(path, sheets[i])

    for k in range(0, len(appl)):                    #for each building, do the following
        my_database[appl.iloc[k][0]]['recommendations'][str(sheets[i])] = {}
        for n in range(1, len(appl.columns)):        #for each recommendation, do the following
            if math.isnan(appl.iloc[k][n]) == False:    #if the ranking value is not empty, do the following
                my_database[appl.iloc[k][0]]['recommendations'][str(sheets[i])][appl.keys()[n]] = appl.iloc[k][n]
        if my_database[appl.iloc[k][0]]['recommendations'][str(sheets[i])] == {}:
            del my_database[appl.iloc[k][0]]['recommendations'][str(sheets[i])]
print (my_database)
```

Figure 21: Screenshot of the PyCharm IDE containing the code of the “Database_forming.py” file

# 5 Experimental results

A dataset from the publicly available electricity consumption dataset was chosen to run the RS algorithm. An overview on the publicly available electricity consumption datasets is provided here in addition to the results obtained.

## 5.1 Electricity consumption dataset

Currently there are many electricity consumption datasets that are publicly available and can be obtained in most cases for free for research purposes. These datasets have different sources with different buildings locations as well as various number of buildings. Another important parameter is the sampling frequency. This parameter defines the frequency with which the readings were obtained. Table 3 [55]–[57] provides a comparison summary of publicly available electricity consumption datasets.

Table 3: Comparison of publicly available electricity consumption datasets [55]–[57]

| Dataset | Institution/Source | Location | Number of buildings | Appliance sample frequency | Aggregate sample frequency |
|---|---|---|---|---|---|
| **PLAID** | Crowdsourcing | Pittsburgh, Pennsylvania, USA | 65 | 30 kHz | 30 kHz |
| **Dataport** | Pecan Street Inc. | Texas, USA | 722 | 1 min | 1 min |
| **REDD** | Massachusetts Institute of Technology (MIT) | Massachusetts, USA | 6 | 3 sec | 1 sec & 15 kHz |
| **BLUED** | Carnegie Mellon University (CMU) | Pennsylvania, USA | 1 | N/A | 12 kHz |
| **Smart** | UMass | Massachusetts, USA | 3 | 1 sec | 1 sec |
| **Household Electricity Use Study (HES)** | DECC, DEFRA, EST | UK | 251 | 2 or 10 min | 2 or 10 min |
| **UK-DALE** | Imperial College | London, UK | 5 | 6 sec | 1-6 sec & 16 kHz |
| **ECO** | ETH Zurich | Switzerland | 6 | 1 sec | 1 sec |
| **SustData** | University of Madeira | Madeira, Portugal | 50 | N/A | 50 Hz |
| **DRED** | TU Delft | Netherlands | 1 | 1 sec | 1 sec |

| | | | | | |
|---|---|---|---|---|---|
| **iAWE** | IIIT Delhi | Delhi, India | 1 | 1 or 6 sec | 1 sec |
| **AMPds 2** | Simon Fraser University | BC, Canada | 1 | 1 min | 1 min |
| **GREEND** | Alpen-Adria-U. Klagenfurt | Italy & Austria | 9 | 1 sec | N/A |
| **UMass Smart** | Data from 3 houses | UK | 3 | 1 sec | 1 sec |
| **Pecan Street Sample** | Pecan Street Inc. | IND | 10 | 1 min | 1 min |
| **COMBED (Commercial Building Energy Dataset)** | IIITD academic building | NL | 8 | 30 sec | 30 sec |
| **BERDS (Berkeley Energy Disaggregation Data Set)** | Cory Hall on the UC Berkeley campus | USA | 1 | 20 sec | 20 sec |

From the above-mentioned datasets the REDD dataset was chosen for applying the recommender algorithm in this work. This is due to the fact that this dataset is one of the easiest to obtain and interpret, in addition to the availability of both aggregated and disaggregated electricity consumption data, and the comparatively low sample frequency rate. These are all parameters that are expected to help building a good disaggregation model for the RS algorithm.

## 5.2 REDD data

The REDD dataset consists of electricity consumption readings from 6 buildings. Each file consists of a timestamp and the corresponding reading of electricity consumption in watts. Each file corresponds to an appliance in the building and another file contains the labels referring to the content of each file.

In this work the disaggregation results using the training-less NILM model were reproduced.

For the purpose of running the algorithm data from house 2 of the REDD dataset was selected as an example. This house has the least number of appliances and the least number of repetitive devices. Thus, it is reasonable to use this dataset for the demonstrative purposes because of the clarity of the results. Thus, the results of the NILM algorithm were reproduced in this work and then utilized for running the RS algorithm.

For the purpose of simplification and in order to reduce the time needed for processing and running the algorithms, the algorithms were run on data for the period from 23.04.2011 to 26.04.2011 of the dataset. In addition, the data were down sampled to the frequency of 1 minute between measurements. As an input, three files of ".csv" format were used. The first file contains the aggregated electricity consumption data of house 2 down sampled to the frequency of one minute, depicted in Figure 22.

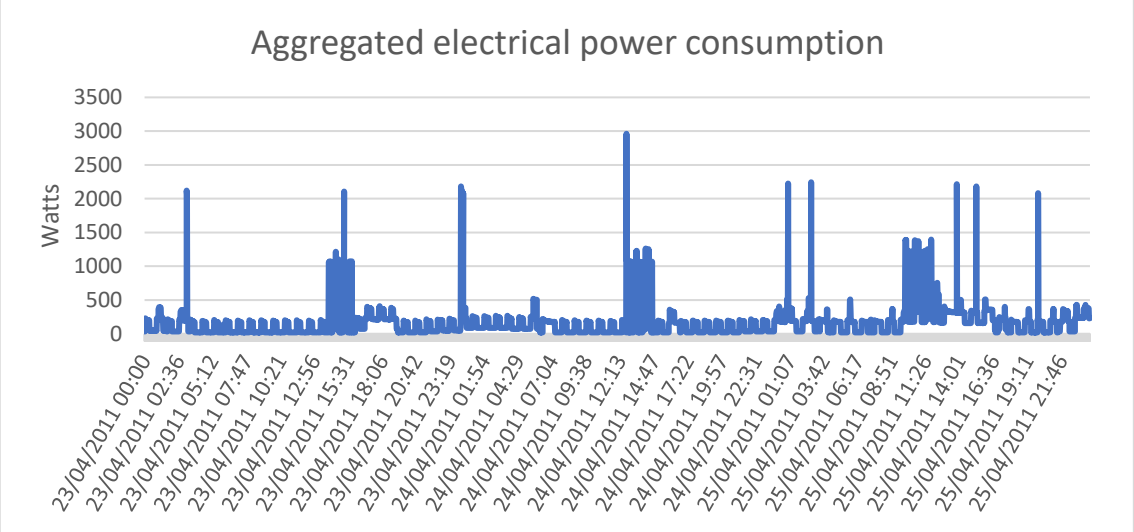



Figure 22: Down sampled electricity consumption data of house 2 from the REDD dataset

The second file contains the power consumption of appliances in house 2 and also down sampled to the frequency of one minute. The consumption data of five appliances were used, namely: refrigerator, kitchen_outlet1, kitchen_outlet2, microwave and lighting. This file was not used as an input for the disaggregation algorithm. However, it was only plotted as ground truth data in order to compare the output of the algorithm with the ground truth. So that it will simplify the validation of the algorithm output. The data are represented in Figure 23, Figure 24, Figure 25, Figure 26 and Figure 27.

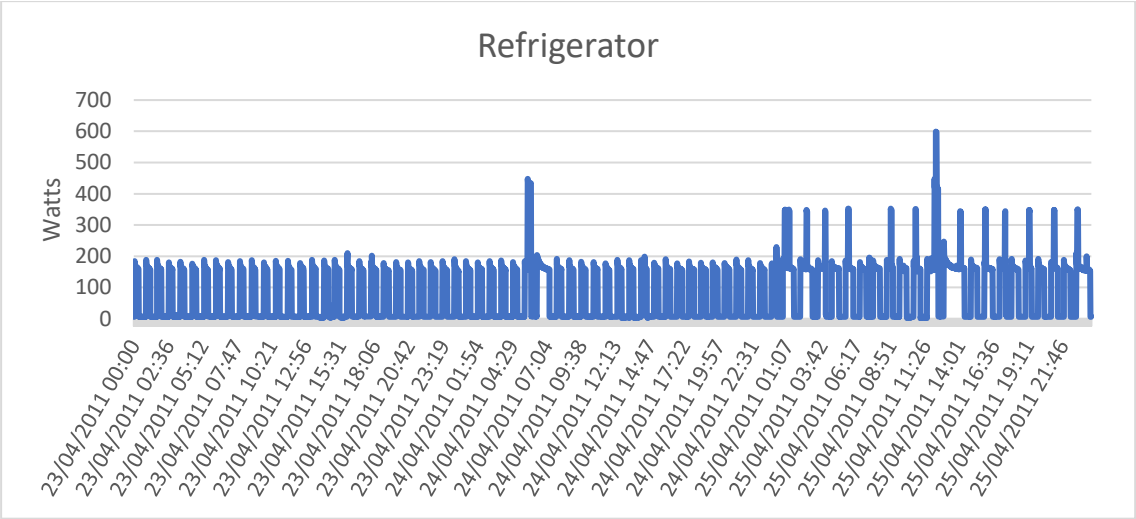



Figure 23: Down sampled electricity consumption data for the refrigerator

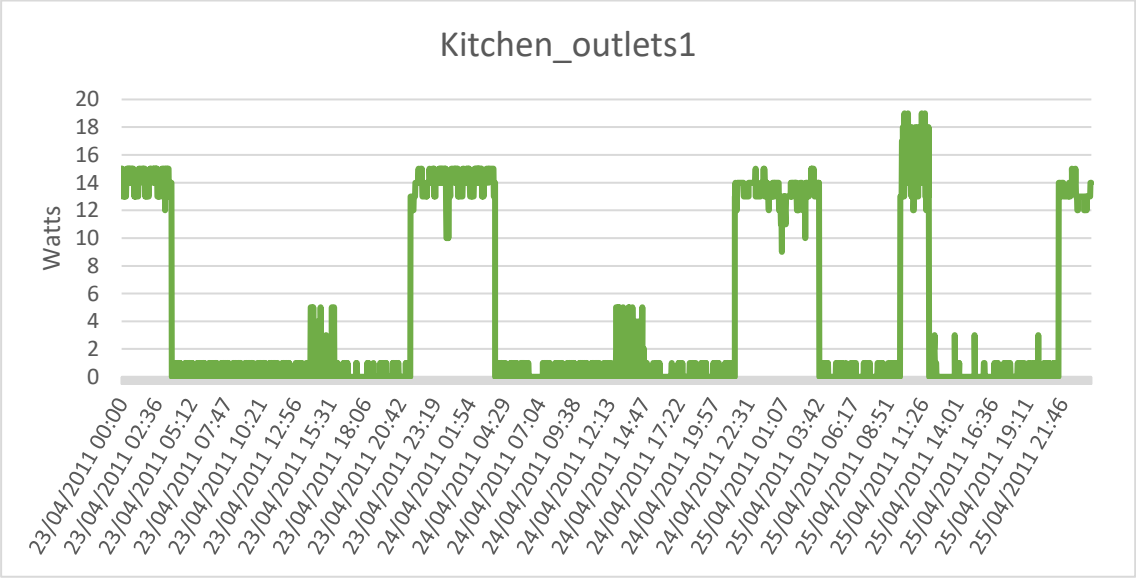



Figure 24: Down sampled electricity consumption data for the kitchen_outlets1

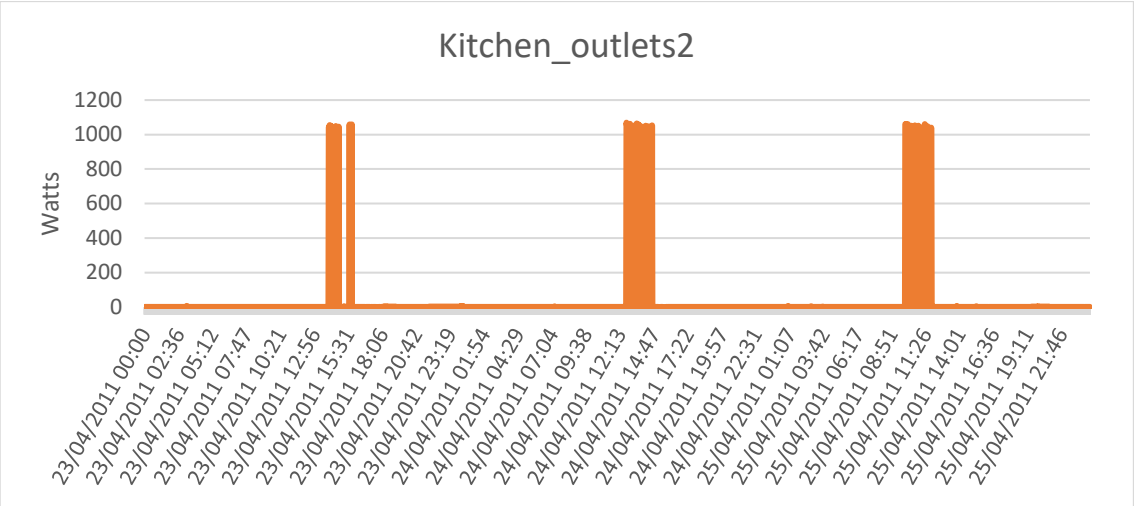



Figure 25: Down sampled electricity consumption data for the kitchen_outlets2

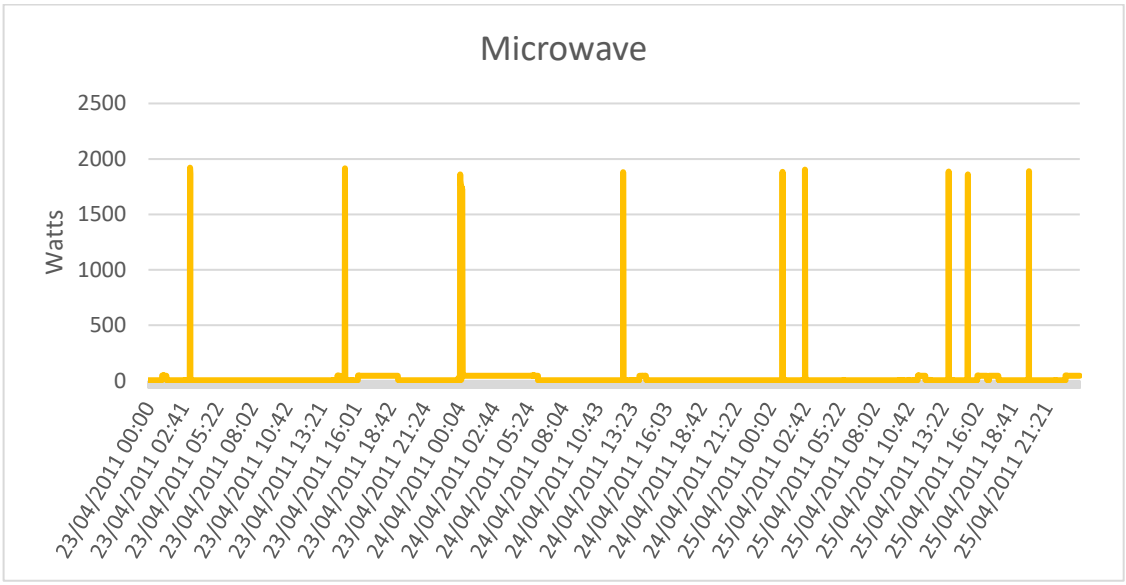



Figure 26: Down sampled electricity consumption data for the microwave

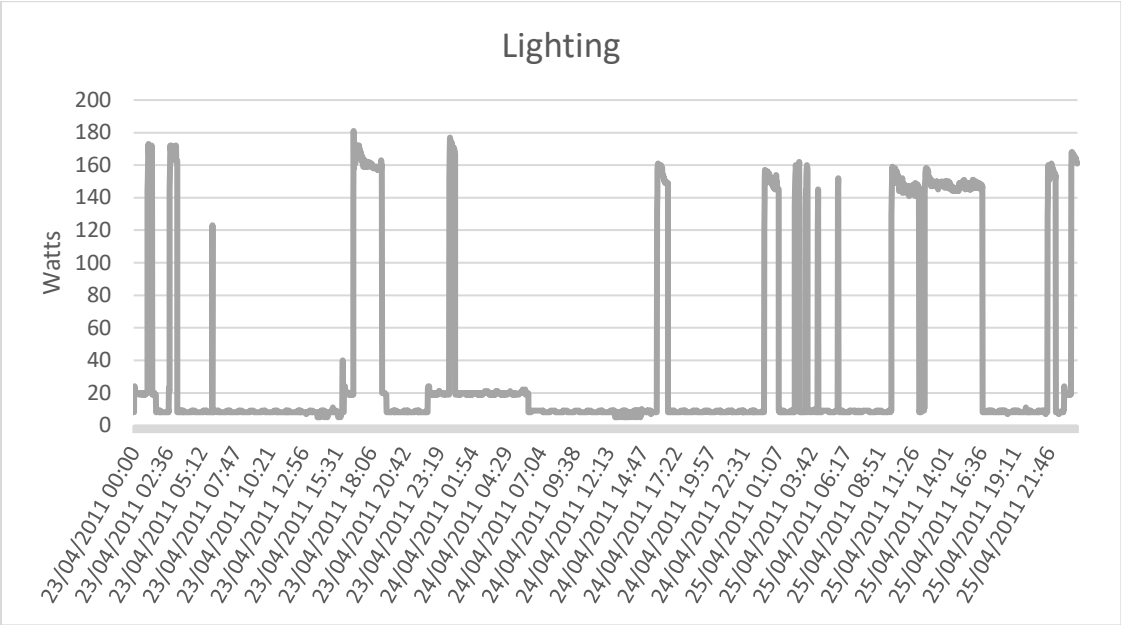



Figure 27: Down sampled electricity consumption data for the lighting

The third file is the signature database. It contains signatures of the electrical appliances. This database was formed by extracting from the ground truth values random sequences of readings when the device is turned on. For each device only one sequence is extracted. This is done mainly to see the behavior of the device during the time when it is turned on. Since the aim of the signature is only to indicate the behavior of each device, no timestamp is assigned in the signature database.

## 5.3 Results of the NILM model

The data described were fed into the disaggregation algorithm using Python code. The code [50] of the disaggregation part consists of two files “gsp_disaggregator.py” which is the main file and “gsp_support.py” which contains the supporting functions. The code from each file is provided in Appendix 1 [50] and Appendix 2 [50] respectively.

After running the NILM algorithm on the dataset, the following results, depicted in Figure 28, were obtained.

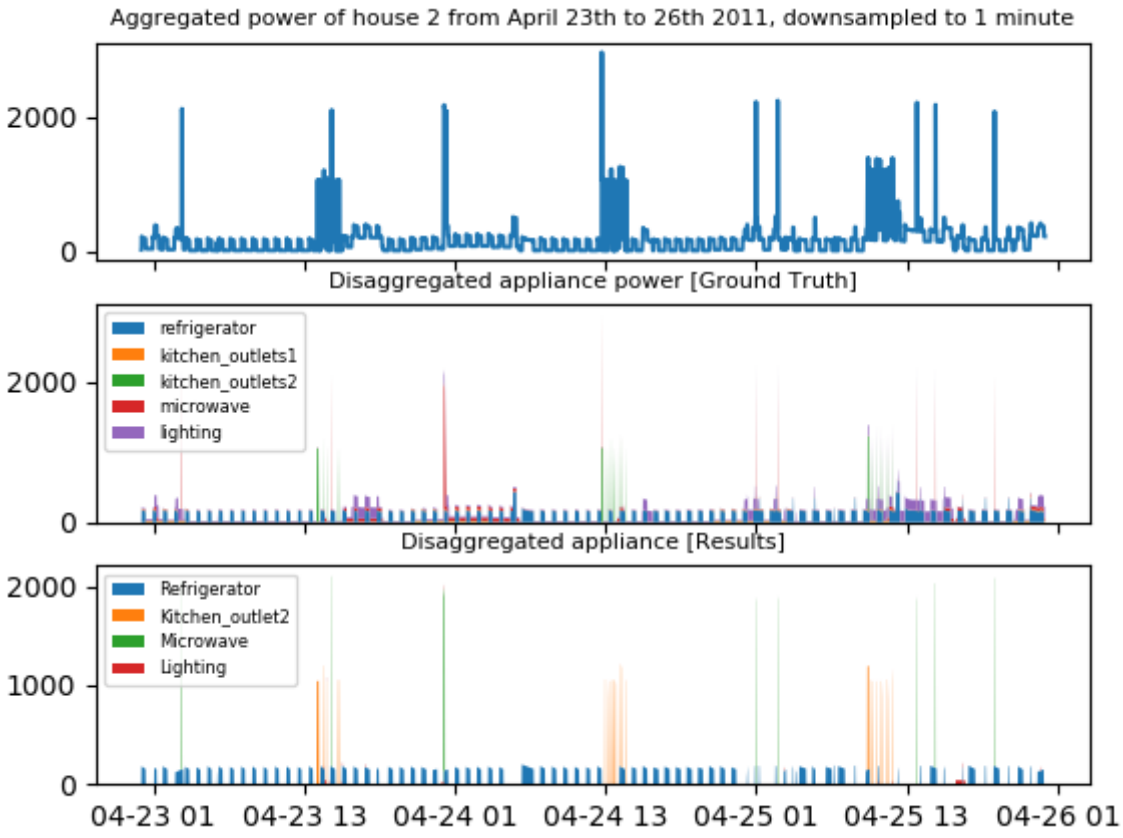



Figure 28: Output of the NILM model

From the obtained results, it can be noticed that out of 5 appliances that were present in the building, the algorithm recognized 4, namely: refrigerator, kitchen_outlets2, microwave and lighting.

By comparing the results outputted by the algorithm with the ground truth, it can be noticed that the algorithm was able to accurately identify the behavior of the fridge as well as the behavior of the microwave and the kitchen_outlets2 appliances. While for the lighting, the predictions were almost negligible.

In order to compare the total energy consumption of each appliance in the result with the ground truth pie charts depicted in Figure 29 were generated.

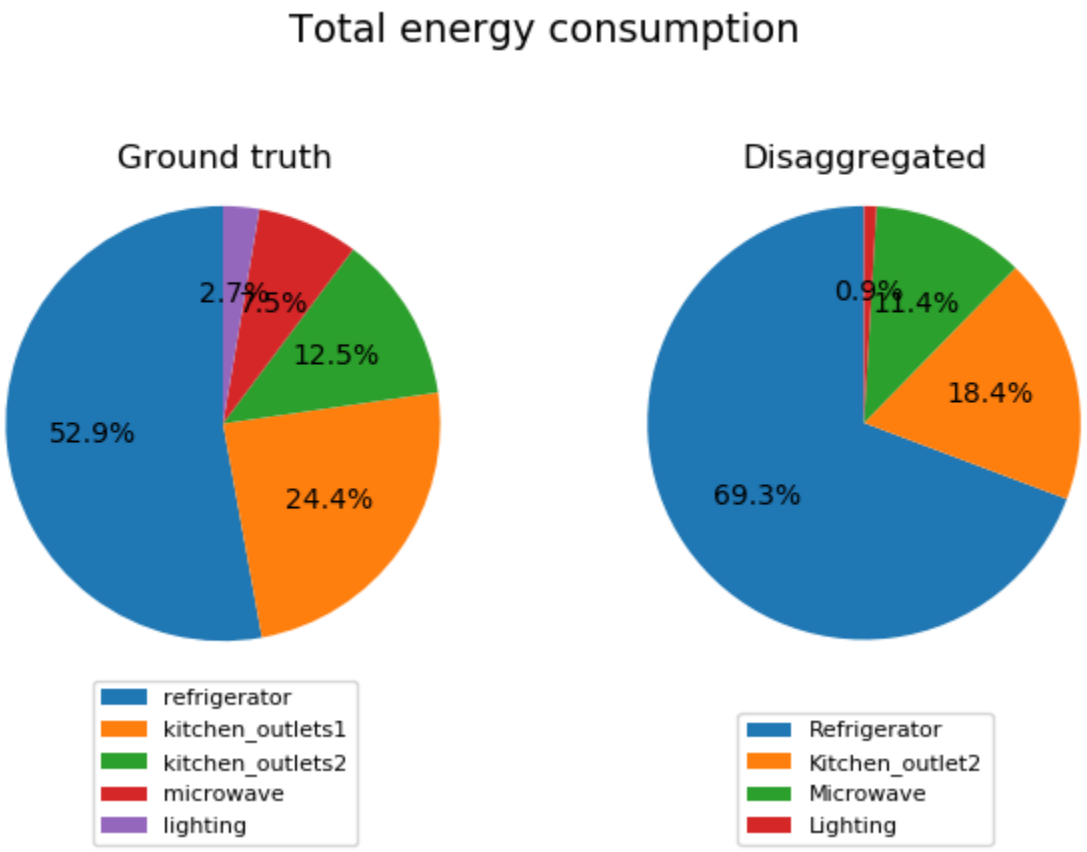



Figure 29: Comparison of the total energy consumption of the disaggregated output with the ground truth

The used NILM algorithm has proven its accuracy in identifying the appliances that are present inside the building. Nevertheless, it also has its drawbacks and space for improvement.

For example, this algorithm would perform with less accuracy when there is a similarity between the signatures of multiple appliances. In this case, the algorithm may falsely attribute the wrong label to the appliance.

The performance accuracy of this NILM model would also drop if a certain appliance in the building is of a certain manufacturer but the same type of the appliance in the signature database belongs to a different manufacturer, and the working power and model are different for these two appliances. In this case the algorithm may not recognize the appliance.

Another point is that this algorithm does not take into account the noise that could be present in the data. This might also affect the accuracy of the performance of the algorithm.

In order to improve the algorithm few suggestions could be made. It is suggested to add a filter that would eliminate the noise in the data. Also, creating a universal signature database for each type of buildings containing the signatures of typical appliances present in the building would be helpful to improve the performance of the algorithm.

## 5.4 Results of the recommender system

The accuracy of designed CF RS model increases with the increasing number of the buildings in the database. Thus, when running the algorithm on the very-first user, the output will be not personalized. It will contain all the recommendations regarding the appliances present in the building. It is assumed that when using this algorithm for real life cases, at the beginning non-personalized recommendations will be given and every time after the feedback is received, the user is added to the database. This way, the database will be formed out of the real users and their real feedbacks.

Thus, in order to verify the accuracy of the model, a synthetic database was formed. The database consists of 20 buildings. These are existing buildings the data about which were obtained from [58]. The data about the climate of the area where each building is located were collected from the Temperature Regime map in [59]. The buildings that were used to form the database are represented in Table 4. A snapshot of the database is provided in Appendix 3.

Table 4: Buildings database for the RS algorithm

| Building | Type | Location | Climate |
|---|---|---|---|
| Hotel Me (Madrid) | hotel | Spain | warm temperature dry |
| Plaza De Las Cortes 3 (Madrid) | residential | Spain | warm temperature dry |
| Centro Civico Aldabe (Vitoria-Gasteiz) | office | Spain | cool temperature |
| W Hotel City Center (Chicago) | hotel | USA | cool temperature |
| 150 Powell Street | residential | USA | warm temperature dry |
| Travis Tower (Houston) | office | USA | sub tropical |
| Domaine Du Mandravasarotra (Belobaka) | hotel | Madagascar | tropical |
| Kk Home Tomasina (Toamasina) | residential | Madagascar | sub tropical |
| NSI Office (Antananarivo) | office | Madagascar | warm temperature dry |
| Embassy Suites Hotel Houston/Downtown (Houston) | hotel | USA | sub tropical |
| Hotel Dnipro (Kyiv) | hotel | Ukraine | cool temperature |
| Admiralty Arch (London) | office | UK | warm temperature dry |

| | | | |
|---|---|---|---|
| The Palace Hotel (San Francisco) | hotel | USA | warm temperature dry |
| Objet Deco (Mahajanga) | residential | Madagascar | tropical |
| Horizon Office Tower (Kyiv) | office | Ukraine | cool temperature |
| Sofitel Madrid Plaza De Espana (Madrid) | hotel | Spain | warm temperature dry |
| One Thousand Powell Apartments (San Francisco) | residential | USA | warm temperature dry |
| Adlington House (Liverpool) | residential | UK | cool temperature |
| Vulytsia Vorovskogo 11 (Kyiv) | residential | Ukraine | cool temperature |
| Gulliver (Kyiv) | office | Ukraine | cool temperature |

For each of the buildings from the table above, certain recommendations were given to reduce the energy consumption. These recommendations are summarized according to the appliance in concern in Table 5.

Table 5: Recommendations for the RS algorithm

| Appliance | Recommendations |
|---|---|
| **Refrigerator** | “Place the refrigerator away from heat sources” |
| | “Avoid putting hot food directly in the refrigerator” |
| | “Try to keep the refrigerator filled in to save energy” |
| **Microwave** | “Cover the dishes before putting them in the microwave to cut down the cooking time” |
| | “Cut the food into small pieces to reduce the cooking time” |
| **Lighting** | “Place movement detectors to turn off the lights when the room is empty” |
| | “Install task lightings in places like on the study desk etc. to reduce the electricity consumption from using the general lighting” |
| | “Consider using light colored paint” |
| **Stove** | “Shift the usage of electrical stove to the late hours” |
| | “Take into account the heating area of your stove to choose pans with proper diameters” |
| **Air Conditioner** | “Keep the curtains and blinds closed to reduce the space from heating up” |
| | “Use a programmable thermostat that turns off the AC when the space is empty” |
| **Washing Machine** | “Use the washing machine of the right size since the bigger the machine is, the more power it consumes” |
| | “Use front load washing machines which consume less electricity than the top load” |
| | “Do not leave the machine in standby mode” |
| **Water Heater** | “Insulate the pipes connected to the heater” |
| | “Prefer taking a short shower instead of a bath” |
| | “Consider installing heat traps on the water heater” |

After that, to assign each recommendation a rating that was given by each of the buildings from the formed dataset, several assumptions were made regarding each of the appliances.

These assumptions were made because different factors of each building have an influence on the final rating that will be given to each recommendation, as was already described in section 4.2. In real life the factors may differ from the assumptions that were made in this work. However, the main goal of making such assumptions is to systematize the ratings in the database, make them consistent, and to be able to validate the model by comparing the outputted ratings with the ratings that should be assigned following the assumptions made.

Some general assumptions were made in addition to assumptions regarding each of the appliances and each recommendation in particular. These assumptions are summarized in Table 6.

Table 6: Assumptions for validating the RS

| Appliance | Recommendations | Assumption |
|---|---|---|
| **Refrigerator** | "Place the refrigerator away from heat sources" | This recommendation is not applicable for the buildings in hot climate zones (warm temperature dry, sub-tropical and tropical) due to the overall hot temperature which makes it difficult to follow the recommendation. |
| | "Avoid putting hot food directly in the refrigerator" | This recommendation is not applicable for hotels and offices because due to the amount of food being prepared and stored in the hotel it is difficult to keep storing it outside of the fridge while it is cooling down. In offices there is a very low possibility that office workers will follow the recommendation because of the lack of responsibility in paying the bill and priorities shifted more towards productive work rather than reducing energy bill for the employer. |
| | "Try to keep the refrigerator filled in to save energy" | This recommendation is not applicable for the residential and office buildings. Because it is difficult to maintain having a big amount of food all the time in these types of buildings. This recommendation is not applicable for countries with GDP lower than 20000$ per capita because of the difficulty to maintain having a big amount of food due to the low level of economy. |
| **Microwave** | "Cover the dishes before putting them in the microwave to cut down the cooking time" | This recommendation is not applicable for the offices due to the lack of equipment and different priorities of the employees. |
| | "Cut the food into small pieces to reduce the cooking time" | This recommendation is not applicable for offices and hotels because it requires time and ruins the visual presentation of the dish which is not acceptable in these types of buildings. |
| **Lighting** | "Place movement detectors to turn off the lights when the | This recommendation is not applicable because countries with GDP lower than 20000$ per capita will not apply recommendations that need additional investments. |

| | | |
|---|---|---|
| | room is empty” | |
| | “Install task lightings in places like on the study desk etc. to reduce the electricity consumption from using the general lighting” | This recommendation is not applicable for hotels and offices because in these types of buildings it is difficult to control the behavior of the people inside. It is also not applicable because countries with GDP lower than 20000$ per capita will not apply recommendations that need additional investments. |
| | “Consider using light colored paint” | This recommendation is not applicable for hotels because it is important to maintain the unique interior design of the hotel which may not accept having some light colors. It is also not applicable for countries with GDP lower than 20000$ per capita because it needs additional investments. It is also not applicable for buildings with warm climate types since the effect of the light paint is negligible due to the big amount of sunlight received. |
| **Stove** | “Shift the usage of electrical stove to the late hours” | This recommendation is not applicable to residential buildings and offices because of the difficulty of shifting the operation hours in these types of buildings. |
| | “Take into account the heating area of your stove to choose pans with proper diameters” | This recommendation is not applicable in offices because of the difficulty of purchasing new pans by the employees. It is also not applicable for countries with GDP lower than 20000$ per capita because of the additional investments needed. |
| **Air Conditioner** | “Keep the curtains and blinds closed to reduce the space from heating up” | This recommendation is not applicable for hotels and offices because of the difficulty of controlling the behavior of people inside these buildings. It is also not applicable for buildings located in cold climates (polar, boreal and cool temperature) because of the negligible amount of sun that heats up the space through the windows in these climates. |
| | “Use a programmable thermostat that turns off the AC when the space is empty” | This recommendation is not applicable for countries with GDP lower than 20000$ per capita because it requires additional investments. |
| **Washing Machine** | “Use the washing machine of the right size since the bigger the machine is, the more power it consumes” | This recommendation is not applicable for residential buildings and offices because the amount of clothes to be washed will vary each time. It is also not applicable for countries with GDP lower than 20000$ per capita because it requires additional investments for purchasing a new washing machine. |
| | “Use front load washing machines which | This recommendation is not applicable for countries with GDP lower than 20000$ per capita because it requires additional investments for purchasing a new washing machine. |

| | | |
|---|---|---|
| | consume less electricity than the top load” | |
| | “Do not leave the machine in standby mode” | This recommendation is not applicable for hotels because of the difficulty of turning off the machine after each wash due to the high frequency of usage. |
| **Water Heater** | “Insulate the pipes connected to the heater” | This recommendation is not applicable for buildings in hot climate zones due to the low impact of insulation because of generally high temperatures in these climates. |
| | “Prefer taking a short shower instead of a bath” | This recommendation is not applicable for offices because typically it is rare to have showers in office buildings and almost unrealistic to have bathtubs. So, this recommendation simply does not change the behavior of the people inside the building. |
| | “Consider installing heat traps on the water heater” | This recommendation is not applicable for countries with GDP less than 20000$ per capita because it requires additional investments. It is also not applicable for hot climate zones because of the overall hot temperatures in these climates. |

GDP data are based on [60].

Based on this information another assumption was made, that each category of the basic information (type, location and climate) contributes by one point to the resulting rating of each recommendation. If the recommendation is not applicable for a certain type of basic information (for example, not applicable for hotels), a contribution of zero will be made from that category towards the resulting rating of the recommendation. With this being said, if, for example there is a building with the basic information (type: hotel, location: UK, climate: polar) and the recommendation is applicable for hotels as well as for the UK and for cold climates, then the rating for this recommendation will be 3. If another recommendation is considered for this building, which is applicable for hotels as well as for the UK but it is not applicable for cold climates, then this recommendation will gain the rating of 2 and so on.

Based on this, ratings were assigned to each recommendation for each of the buildings from the database. The ratings assigned by each building for each recommendation are depicted in the snapshots in Appendix 4.

After that, the output resulting from the NILM model was fed into the CF model and the recommendations for this building were given. The CF model was implemented in Python in PyCharm and connected to the NILM python code to form one connected system. The code that was written for the CF part of the model consists of two files “Database_forming.py” and “Recommender.py”. The first file reads the data from the .csv file containing

the database of the buildings and forms an internal dataset. The second file takes the output of the NILM model and gives the recommendations for the considered building. The codes from the files “Database_forming.py” and “Recommender.py” can be found in Appendix 5 and Appendix 6 respectively.

As an input for the model the following basic information was used: location: ‘USA’, type: ‘residential’, and climate: ‘warm temperature dry’. From the output of the NILM model it was identified that the following appliances are present in the building: refrigerator, kitchen_outlets2, microwave and lighting. As a resulting output the following recommendations were given with the predicted ratings:

- 'Cut the food into small pieces to reduce the cooking time', 3
- 'Avoid putting hot food directly in the refrigerator', 3
- 'Install task lightings in places like on the study desk etc. to reduce the electricity consumption from using the general lighting', 3
- 'Cover the dishes before putting them in the microwave to cut down the cooking time', 3
- 'Place movement detectors to turn off the lights when the room is empty', 3
- 'Place the refrigerator away from heat sources', 2
- 'Try to keep the refrigerator filled in to save energy', 2
- 'Consider using light colored paint', 2

From the obtained results it can be seen that all the suggested recommendations concern only the appliances that were identified by the NILM algorithm. Which means that the filtering of the recommendations was done correctly. In order to measure the accuracy of the predicted ratings, the ratings for the considered building were also calculated based on the assumptions that were made previously in Table 6. After that, the accuracy was measured by considering the calculated ratings based on the assumptions to be ground truth. The closer the ratings predicted by the model to the ground truth, the higher the accuracy.

The “ground truth” ratings were calculated as following:

- 'Cut the food into small pieces to reduce the cooking time', 3
- 'Avoid putting hot food directly in the refrigerator', 3
- 'Install task lightings in places like on the study desk etc. to reduce the electricity consumption from using the general lighting', 3

- 'Cover the dishes before putting them in the microwave to cut down the cooking time', 3
- 'Place movement detectors to turn off the lights when the room is empty', 3
- 'Place the refrigerator away from heat sources', 2
- 'Try to keep the refrigerator filled in to save energy', 2
- 'Consider using light colored paint', 2

In this case it can be seen that the predicted ratings fully match the ground truth ratings. Thus, the accuracy can be calculated using (8).

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{8}$$

Where:

TP – true positive (the predicted rating is positive and the ground truth is positive),

TN – true negative (the predicted rating is negative and the ground truth is negative),

FP – false positive (the predicted rating is positive but the ground truth is negative),

FN – false negative (the predicted rating is negative but the ground truth is positive).

In order to apply (7) it was assumed that the maximum possible rating, which is 3 in our case is the positive rating and everything below that value (2, 1 and 0) are negative ratings. This way the accuracy of the predicted values for this case was calculated as follows: $Accuracy = \frac{(5+3)}{(5+3+0+0)} = 1$.

This means that in this case the model managed to predict the ratings with 100% accuracy.

In order to further validate the model recommendations were generated to six more buildings with different basic information. After that, the accuracy of the predicted ratings was measured.

A summary of the buildings used for validation and the calculated accuracy of the ratings predictions for each building is summarized in Table 7.

Table 7: Validation summary of the RS

| | Location (L) | Type (T) | Climate (C) | Appliances | Matching buildings | Matching features | | Accuracy |
|---|---|---|---|---|---|---|---|---|
| | | | | | | Features | Number | |
| **1** | USA | residential | Warm temperature dry | Refrigerator, microwave, lighting, kitchen_outlets2 | 15 | All | 3 | 1 |
| | | | | | 17 | All | | |
| **2** | Ukraine | Office | Cool temperature | Refrigerator, lighting, microwave, air conditioner | 15 | All | 3 | 1 |
| | | | | | 20 | All | | |
| **3** | USA | Hotel | Warm temperature dry | Refrigerator, lighting, microwave, washing machine | 13 | All | 3 | 1 |
| **4** | UK | Office | Cool temperature | Microwave, lighting, air conditioner, water heater | 3 | T, C | 2 | 0.5 |
| | | | | | 12 | T, L | | |
| | | | | | 15 | T, C | | |
| | | | | | 18 | L, C | | |
| | | | | | 20 | T, C | | |
| **5** | Ukraine | residential | Cool temperature | Microwave, lighting, stove, washing machine, water heater | 19 | All | 3 | 1 |
| **6** | Kazakhstan | residential | Cool temperature | Microwave, lighting, stove, washing machine, water heater | 18 | T, C | 2 | 1 |
| | | | | | 19 | T, C | | |
| **7** | Kazakhstan | hotel | Cool temperature | Microwave, air conditioner, water heater | 4 | T, C | 2 | 1 |
| | | | | | 11 | T, C | | |

From Table 7 it can be seen that the accuracy of the predicted ratings is 1 when the algorithm can find one or more building in the database that have all the features matching the features of the building in concern. The prediction accuracy is also 1 when the algorithm finds buildings in the database with a smaller number of matching features, but the features that match are the same in each of the matching buildings. For example, in case 6 in the table there are two matching buildings that have the same type and climate of the target building.

However, in the case when the algorithm finds more than one matching building and each of the buildings have different matching features with the target building, the prediction accuracy is comparatively low. For example, like in the case 4 in the table above. The algorithm found 5 matching buildings. Three of them have same type and climate as the target building. One of them has the same type and location as the target building, and another one has the same location and climate as the target building. In this case the accuracy of the predictions was 0.5, which is considerably lower than the prediction accuracy in the other cases.

With this being said, a conclusion can be made on how to improve the overall accuracy of the system. This is explained in more details in the next chapter.

# 6 Conclusions and future work

Many RS have been proposed for a variety of fields, yet their application for improving electricity usage has not been widely explored. Thus, in order to implement an electricity usage RS in real life may necessitate implementing such a system from scratch.

Due to this fact, the available data can be very limited, because no previously given recommendations are available. Also, there is limited availability of data that accurately describe target user preferences.

In order to tackle this issue and find a way to develop an electricity usage RS without the availability of extensive data, the system described in this work was proposed. This RS can be used in order to provide customers with personalized electricity usage recommendations. Each time the system is used to provide recommendations for a new building, this building is added to the system's database. Feedback from customers along with their ratings are also added to the database. This way, with more data, recommendations become more accurate.

The electricity usage RS proposed in this work can be used for cases when there are limited input data. The algorithm of the system was implemented in Python. The RS consists of two parts: the NILM model and the recommender model. Depending on the available input data, any NILM model can be embedded in the system. In case of very limited input data like in this work, using a training-less NILM model is a reasonable solution. Any of the NILM models (introduced in section 3.5) can fit the system as long as it outputs the names of the appliances present in the building. For this work the NILM algorithm suggested in [50] was used as the first part of the system.

The second part is the recommender model itself. The algorithm for this model was designed to make the most out of the limited available data. The algorithm is based on the principle of user-to-user CF RS. In order to find the neighboring users, some general information about the user was utilized, referred to as "basic information". After identifying the neighboring users, the algorithm filters the recommendations that were given to these users to eliminate the recommendations for appliances that are not present in the target building. Then, the resulting recommendations are given to the user in descending order, based on the average rating given by the neighboring users for each recommendation.

Based on the validation of the RS algorithm a conclusion on the accuracy of the system can be drawn. It was discovered that the system demonstrates 100% accuracy when the neighboring users match the target user according to all the general parameters. It also demonstrates the same level of accuracy when all the matching buildings have the same parameters that match the target building. However, when the matching features are different in each neighboring building, the accuracy of the algorithm may decrease down to 50%.

Thus, some suggestions can be made as future work, in order to enhance accuracy of the system:

- A weighting scale can be introduced to improve the measurement of the distance from the target user to the neighboring users. For example, the level of importance of each of the parameters can be defined and after that, in the case when matching buildings have different matching parameters, this weight can be included in calculating the rating of each of the recommendations. In this way the accuracy of the model can be improved.
- Other parameters can be added to these identifying neighboring users. For example, the output of the NILM algorithm which identifies what appliances are present in the building can also be used to assess the similarity between the buildings. In this way, the similarity of appliances that are present in two or more buildings will increase the level of similarity between these buildings and vice versa.

RS in general are gaining more popularity in all sectors due to the rapid development of technologies. Thus, the necessity for developing and implementing effective tools that can provide the users with recommendations to improve their performance and reduce the bills is an important task that can tremendously help both the consumers and the service providers.

In real life the available data may be limited. The limitation may be as a consequence of a variety of factors for example, the privacy policy can limit the amount of data available or simply the difficulty of obtaining the data can be an issue as well. Thus, when designing a recommender system these factors should also be taken into account. Moreover, when designing a RS, it is necessary to have some previously existing data in the system. This data helps the RS to identify what are the interests of the target user, what are the interests of the other users that are present in the database, and to measure the level of similarity between the target user and the other users in the database.

The unavailability of these data creates a challenge when developing a new RS. One of the ways to tackle this challenge is to search for some publicly available datasets to form the database that the RS can rely on.

However, there could be no relevant data found that can be used for this purpose. In this case, a reasonable solution could be to accumulate the data about the users every time the RS is used to generate recommendations. This is the principle that was used in the proposed RS.

By proposing an electricity usage RS with limited input data, a solution was suggested for such cases when there is no enough data that allows to directly implement a RS algorithm. The algorithm was proposed for generating recommendations for the usage of electrical appliances. However, the described challenge may arise when developing a RS in any other field, especially the ones on which only few researches were done. In this case another algorithm should be developed in order to design the RS with limited input data for this particular purpose.

Thus, further research and work in the field of designing RS with limited input data conditions in different fields is a topic with undoubtful importance and many open questions that need to be answered.

# Bibliography

[1] I. Schoinas and C. Tjortjis, 'MuSIF: A Product Recommendation System Based on Multi-source Implicit Feedback', in *Artificial Intelligence Applications and Innovations*, Cham, 2019, pp. 660–672. doi: 10.1007/978-3-030-19823-7_55.

[2] O. Nalmpantis and C. Tjortjis, 'The 50/50 Recommender: A Method Incorporating Personality into Movie Recommender Systems', in *Engineering Applications of Neural Networks*, Cham, 2017, pp. 498–507. doi: 10.1007/978-3-319-65172-9_42.

[3] T. A. Runkler, *Data Analytics: Models and Algorithms for Intelligent Data Analysis*. Wiesbaden: Springer Gablerin Springer Fachmedien Wiesbaden GmbH, Springer Fachmedien Wiesbaden GmbH, Springer, 2016.

[4] 'Data Mining vs Data Analysis | Know Top 7 Amazing Comparisons', *EDUCBA*, Apr. 01, 2018. https://www.educba.com/data-mining-vs-data-analysis/ (accessed Mar. 22, 2021).

[5] R. Kubaizi, S. Alotaibi, B. Washigry, E. Suhaim, J. Sughayer, and R. Jumaiah, 'Mining Expertise Using Social Media Analytics', Apr. 2018, pp. 1–5. doi: 10.1109/CAIS.2018.8442014.

[6] S. Zhang, C. Zhang, and Q. Yang, 'Data preparation for data mining', *Applied Artificial Intelligence*, vol. 17, no. 5–6, pp. 375–381, May 2003, doi: 10.1080/713827180.

[7] H. Yang, K. Process, and S. K. Steps, *Data Cleaning Data Integration Databases Data Warehouse Task-relevant Data Selection Data Mining Pattern Evaluation*.

[8] C. Chen, W. K. Härdle, and A. Unwin, *Handbook of Data Visualization*. Springer Science & Business Media, 2007.

[9] F. Shu, W. Quan, B. Chen, J. Qiu, C. R. Sugimoto, and V. Larivière, 'The role of Web of Science publications in China's tenure system', *Scientometrics*, vol. 122, no. 3, pp. 1683–1695, Mar. 2020, doi: 10.1007/s11192-019-03339-x.

[10] P. Farrell *et al.*, 'COVID-19 and Pacific food system resilience: opportunities to build a robust response', *Food Sec.*, vol. 12, no. 4, pp. 783–791, Aug. 2020, doi: 10.1007/s12571-020-01087-y.

[11] National Genomics Data Center Members and Partners, ‘Database Resources of the National Genomics Data Center in 2020’, *Nucleic Acids Research*, vol. 48, no. D1, pp. D24–D33, Jan. 2020, doi: 10.1093/nar/gkz913.

[12] K. Christantonis and C. Tjortjis, ‘Data Mining for Smart Cities: Predicting Electricity Consumption by Classification’, Jul. 2019, pp. 1–7. doi: 10.1109/IISA.2019.8900731.

[13] I. Bruha and A. Famili, ‘Postprocessing in machine learning and data mining’, *SIGKDD Explor. Newsl.*, vol. 2, no. 2, pp. 110–114, Dec. 2000, doi: 10.1145/380995.381059.

[14] European Commission, Ed., *European SmartGrids Technology Platform. Vision and Strategy for Europe’s Electricity Networks of the Future*. Brussels, Belgium: Office for Official Publications of the European Communities, 2006.

[15] S. Paul, M. S. Rabbani, R. K. Kundu, and S. M. R. Zaman, ‘A review of smart technology (Smart Grid) and its features’, in *2014 1st International Conference on Non Conventional Energy (ICONCE 2014)*, Jan. 2014, pp. 200–203. doi: 10.1109/ICONCE.2014.6808719.

[16] C. Teng, ‘Research on Improvement Path of China’s Smart Grid Security Control’, *IOP Conf. Ser.: Earth Environ. Sci.*, vol. 440, p. 032089, Mar. 2020, doi: 10.1088/1755-1315/440/3/032089.

[17] A. Almalaq and J. J. Zhang, ‘Deep Learning Application: Load Forecasting in Big Data of Smart Grids’, in *Deep Learning: Algorithms and Applications*, W. Pedrycz and S.-M. Chen, Eds. Cham: Springer International Publishing, 2020, pp. 103–128. doi: 10.1007/978-3-030-31760-7_4.

[18] ‘Study and comparison of demand response aggregation in Europe and USA’. https://www.politesi.polimi.it/handle/10589/154587 (accessed Mar. 25, 2021).

[19] Y. Wang, Q. Chen, and C. Kang, *Smart Meter Data Analytics: Electricity Consumer Behavior Modeling, Aggregation, and Forecasting*. Singapore: Springer Singapore, 2020. doi: 10.1007/978-981-15-2624-4.

[20] Y. Wang, Q. Chen, T. Hong, and C. Kang, ‘Review of Smart Meter Data Analytics: Applications, Methodologies, and Challenges’, *IEEE Trans. Smart Grid*, vol. 10, no. 3, pp. 3125–3148, May 2019, doi: 10.1109/TSG.2018.2818167.

[21] D. M. Jiménez-Bravo, J. Pérez-Marcos, D. H. De la Iglesia, G. Villarrubia González, and J. F. De Paz, 'Multi-Agent Recommendation System for Electrical Energy Optimization and Cost Saving in Smart Homes', *Energies*, vol. 12, no. 7, Art. no. 7, Jan. 2019, doi: 10.3390/en12071317.

[22] F. Ricci, L. Rokach, and B. Shapira, 'Introduction to Recommender Systems Handbook', in *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, Eds. Boston, MA: Springer US, 2011, pp. 1–35. doi: 10.1007/978-0-387-85820-3_1.

[23] 'Trust-aware recommender systems'. https://dl.acm.org/doi/10.1145/1297231.1297235 (accessed Mar. 26, 2021).

[24] C. A. Gomez-Uribe and N. Hunt, 'The Netflix Recommender System: Algorithms, Business Value, and Innovation', *ACM Trans. Manage. Inf. Syst.*, vol. 6, no. 4, p. 13:1-13:19, Dec. 2016, doi: 10.1145/2843948.

[25] M. F. Gaw, 'Algorithmic logics of taste: Cultural taste and the Netflix recommender system', Thesis, Faculty of Arts and Social Sciences, 2019. Accessed: Mar. 26, 2021. [Online]. Available: https://ses.library.usyd.edu.au/handle/2123/21530

[26] A. Kumar, 'A survey on popular recommender systems', vol. 6, p. 5, Jun. 2019.

[27] Z. Ma *et al.*, 'The Role of Data Analysis in the Development of Intelligent Energy Networks', *IEEE Network*, vol. 31, no. 5, pp. 88–95, 2017, doi: 10.1109/MNET.2017.1600319.

[28] J. Das, P. Mukherjee, S. Majumder, and P. Gupta, 'Clustering-Based Recommender System Using Principles of Voting Theory', presented at the Proceedings of 2014 International Conference on Contemporary Computing and Informatics, IC3I 2014, Nov. 2014. doi: 10.1109/IC3I.2014.7019655.

[29] S. Chadoulos, I. Koutsopoulos, and G. C. Polyzos, 'Mobile Apps Meet the Smart Energy Grid: A Survey on Consumer Engagement and Machine Learning Applications', *IEEE Access*, vol. 8, pp. 219632–219655, 2020, doi: 10.1109/ACCESS.2020.3042758.

[30] F. Luo, G. Ranzi, X. Wang, and Z. Y. Dong, 'Service Recommendation in Smart Grid: Vision, Technologies, and Applications', in *2016 9th International Conference on Service Science (ICSS)*, Oct. 2016, pp. 31–38. doi: 10.1109/ICSS.2016.12.

[31] F. Luo, G. Ranzi, X. Wang, and Z. Y. Dong, 'Social Information Filtering-Based Electricity Retail Plan Recommender System for Smart Grid End Users', *IEEE*

*Transactions on Smart Grid*, vol. 10, no. 1, pp. 95–104, Jan. 2019, doi: 10.1109/TSG.2017.2732346.

[32] J. E. Fischer *et al., Recommending Energy Tariffs and Load Shifting Based on Smart Household Usage Profiling*.

[33] Y. Zhang, K. Meng, W. Kong, and Z. Y. Dong, 'Collaborative Filtering-Based Electricity Plan Recommender System', *IEEE Transactions on Industrial Informatics*, vol. 15, no. 3, pp. 1393–1404, Mar. 2019, doi: 10.1109/TII.2018.2856842.

[34] F. Luo, G. Ranzi, W. Kong, G. Liang, and Z. Y. Dong, 'Personalized Residential Energy Usage Recommendation System Based on Load Monitoring and Collaborative Filtering', *IEEE Transactions on Industrial Informatics*, vol. 17, no. 2, pp. 1253–1262, Feb. 2021, doi: 10.1109/TII.2020.2983212.

[35] 'Electricity plan recommender system with electrical instruction-based recovery | Elsevier Enhanced Reader'. https://reader.elsevier.com/reader/sd/pii/S0360544220308823?token=2025A4D4FF6390E38192C3AD2F7B56DB32D4D52BCCDB0AA2750091E5B9618B1B574287F2F3BC66F8276AC26FF32D1752&originRegion=eu-west-1&originCreation=20210401183410 (accessed Apr. 01, 2021).

[36] Y. Zhang, W. Kong, Z. Y. Dong, K. Meng, and J. Qiu, 'Big Data-driven Electricity Plan Recommender System', in *2018 IEEE Power Energy Society General Meeting (PESGM)*, Aug. 2018, pp. 1–5. doi: 10.1109/PESGM.2018.8585885.

[37] S. Murphy, Ó. Manzano, and K. Brown, 'Design and Evaluation of a Constraint-Based Energy Saving and Scheduling Recommender System', Aug. 2015, vol. 9255, pp. 687–703. doi: 10.1007/978-3-319-23219-5_47.

[38] H. Akbari, 'Validation of an algorithm to disaggregate whole-building hourly electrical load into end uses', *Energy*, vol. 20, no. 12, pp. 1291–1301, Dec. 1995, doi: 10.1016/0360-5442(95)00033-D.

[39] A. Zoha, A. Gluhak, M. A. Imran, and S. Rajasegarar, 'Non-intrusive load monitoring approaches for disaggregated energy sensing: a survey.', *Sensors (Basel, Switzerland)*, vol. 12, no. 12, pp. 16838–16866, 2012, doi: http://dx.doi.org.ezproxy1.hw.ac.uk/10.3390/s121216838.

[40] H. Akbari, J. Eto, S. Konopacki, A. Afzal, K. Heinemeier, and L. Rainer, 'A new approach to estimate commercial sector end-use load shapes and energy use intensities',

*American Council for Energy-Efficient Economy (ACEEE) summer conference, Asilomar, CA (United States), 28 Aug - 3 Sep 1994*, Aug. 01, 1994. https://digital.library.unt.edu/ark:/67531/metadc792383/m1/1/ (accessed Apr. 03, 2021).

[41] J. H. Eto, H. Akbari, R. G. Pratt, and S. D. Brathwait, 'End-Use Load Shape Data Application, Estimation, and Collection: A State-of-the-Art Review', Jun. 1990, Accessed: Apr. 03, 2021. [Online]. Available: https://escholarship.org/uc/item/1f52b37h

[42] H. Kim, M. Marwah, M. Arlitt, G. Lyon, and J. Han, 'Unsupervised Disaggregation of Low Frequency Power Measurements', in *Proceedings of the 2011 SIAM International Conference on Data Mining*, 0 vols, Society for Industrial and Applied Mathematics, 2011, pp. 747–758. doi: 10.1137/1.9781611972818.64.

[43] H. Shao, M. Marwah, and N. Ramakrishnan, 'A Temporal Motif Mining Approach to Unsupervised Energy Disaggregation: Applications to Residential and Commercial Buildings', *AAAI*, vol. 27, no. 1, Art. no. 1, Jun. 2013, Accessed: Apr. 03, 2021. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/8485

[44] Hashem Akbari, Kristin E. Heinemeier, Patrick LeConiac, and Denise L. Flora, 'An Algorithm to Disaggregate Commercial Whole-Building Hourly Electrical Load Into End Uses', 1988. Accessed: Apr. 03, 2021. [Online]. Available: https://www.aceee.org/files/proceedings/1988/data/papers/1988_V10_002.pdf

[45] H. Akbari and S. J. Konopacki, 'Application of an End-Use Disaggregation Algorithm for Obtaining Building Energy-Use Data', *Journal of Solar Energy Engineering*, vol. 120, no. 3, pp. 205–210, Aug. 1998, doi: 10.1115/1.2888070.

[46] 'Non-Intrusive Load Monitoring for Smart Grids.pdf'. Accessed: Mar. 30, 2021. [Online]. Available: https://education.emc.com/content/dam/dell-emc/documents/en-us/2018KS_Schneider-Non-Intrusive_Load_Monitoring_for_Smart_Grid_Decision_Support.pdf

[47] B. Zhao, K. He, L. Stankovic, and V. Stankovic, 'Improving Event-Based Non-Intrusive Load Monitoring Using Graph Signal Processing', *IEEE Access*, vol. 6, pp. 53944–53959, 2018, doi: 10.1109/ACCESS.2018.2871343.

[48] A. Faustine, N. H. Mvungi, S. Kaijage, and K. Michael, 'A Survey on Non-Intrusive Load Monitoring Methodies and Techniques for Energy Disaggregation Problem', *arXiv:1703.00785 [cs]*, Mar. 2017, Accessed: Mar. 31, 2021. [Online]. Available: http://arxiv.org/abs/1703.00785

[49] R. Jia, Y. Gao, and C. J. Spanos, ‘A fully unsupervised non-intrusive load monitoring framework’, in *2015 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, Miami, FL, USA, Nov. 2015, pp. 872–878. doi: 10.1109/SmartGridComm.2015.7436411.

[50] B. Zhao, L. Stankovic, and V. Stankovic, ‘On a Training-Less Solution for Non-Intrusive Appliance Load Monitoring Using Graph Signal Processing’, *IEEE Access*, vol. 4, pp. 1784–1799, 2016, doi: 10.1109/ACCESS.2016.2557460.

[51] J. Bobadilla Sancho, F. Ortega Requena, A. Hernando Esteban, and J. Bernal Bermúdez, ‘A collaborative filtering approach to mitigate the new user cold start problem.’, *Knowledge-Based Systems*, vol. 26, pp. 225–238, Feb. 2012.

[52] ‘Facing the cold start problem in recommender systems | Elsevier Enhanced Reader’. https://reader.elsevier.com/reader/sd/pii/S0957417413007240?token=4258E162A09B3C5C02DCC57C7992E597D3BD5E766CBF9ED56CFF8D3BEC28372692824CCBBD7D9E54B888186F117FA209&originRegion=eu-west-1&originCreation=20210505123733 (accessed May 05, 2021).

[53] ‘Building energy consumption factors : a literature review and future research agenda’. http://dl.lib.uom.lk/handle/123/12050 (accessed May 14, 2021).

[54] J. Han, M. Kamber, and J. Pei, ‘2 - Getting to Know Your Data’, in *Data Mining (Third Edition)*, J. Han, M. Kamber, and J. Pei, Eds. Boston: Morgan Kaufmann, 2012, pp. 39–82. doi: 10.1016/B978-0-12-381479-1.00002-2.

[55] O. Parson *et al.*, ‘Dataport and NILMTK: a building data set designed for non-intrusive load monitoring’, presented at the 1st International Symposium on Signal Processing Applications in Smart Buildings at 3rd IEEE Global Conference on Signal & Information Processing (14/12/15 - 16/12/15), Sep. 2015. Accessed: Apr. 07, 2021. [Online]. Available: https://eprints.soton.ac.uk/381304/

[56] ‘Table 1 An overview of PLAID and similar datasets in terms of submetered data sampled at a frequency <1 Hz or ≥1 Hz, aggregated data sampled at a frequency <1 Hz or ≥1 Hz, different appliance operating modes and the number of different buildings.’, Accessed: Apr. 07, 2021. [Online]. Available: https://www.nature.com/articles/s41597-020-0389-7/tables/2

[57] O. Alrawi, I. S. Bayram, S. G. Al-Ghamdi, and M. Koç, 'High-Resolution Household Load Profiling and Evaluation of Rooftop PV Systems in Selected Houses in Qatar', *Energies*, vol. 12, Oct. 2019, doi: 10.3390/en12203876.

[58] 'World | Buildings | EMPORIS'. https://www.emporis.com/buildings (accessed May 14, 2021).

[59] 'World Climate Regions', *ArcGIS StoryMaps*, Jul. 03, 2020. https://storymaps.arcgis.com/stories/61a5d4e9494f46c2b520a984b2398f3b (accessed Jun. 12, 2021).

[60] 'List of countries by GDP (nominal) per capita', *Wikipedia*. Jun. 13, 2021. Accessed: Jun. 13, 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=List_of_countries_by_GDP_(nominal)_per_capita&oldid=1028390752

# Appendix 1

## Python code for the NILM model (file “gsp_disaggregator.py”)

This appendix provides the Python code that was used for the application of the NILM model. It contains the code from the file “gsp_disaggregator.py”. This code was obtained from the GitHub repository, where according to the provided description, it was developed based on the MATLAB code that was provided by the authors of [50]. In this work this publicly available code was used. It was adopted to read the files that were formed during the work and a part of the code that connects it to the RS was added.

```
from __future__ import division
import warnings
warnings.filterwarnings("ignore")
import pandas as pd
import gsp_support as gsp
import matplotlib.pyplot as plt

print("1 of 6> reading data")
csvfileaggr = "./aggr.csv"
csvfiledisaggr = "./disaggr.csv"
df = pd.read_csv(csvfileaggr, index_col = "Time")
df.index = pd.to_datetime(df.index)
dfd = pd.read_csv(csvfiledisaggr, index_col = "Time")
dfd.index = pd.to_datetime(dfd.index)

start_date = '2011-04-23'
end_date = '2011-04-26'
mask = (df.index > start_date) & (df.index < end_date)
df = df.loc[mask]
mask = (dfd.index > start_date) & (dfd.index < end_date)
dfd = dfd.loc[mask]

fig, axs = plt.subplots(3, 1, sharex=True)
axs[0].plot(df)
axs[0].set_title("Aggregated power of house 2 from April 23th to 26th
2011, downsampled to 1 minute", size=8)
axs[1].stackplot(dfd.index, dfd.values.T, labels = list(dfd.col-
umns.values))
axs[1].set_title("Disaggregated appliance power [Ground Truth]",
size=8)
axs[1].legend(loc='upper left', fontsize=6)

sigma = 20;
ri = 0.15
T_Positive = 20;
T_Negative = -20;
alpha = 0.5
beta  = 0.5
instancelimit = 3

main_val = df.values
main_ind = df.index
```

```
data_vec =  main_val
signature_database = "./signature_database_labelled.csv"
threshold = 2000

delta_p = [round(data_vec[i+1] - data_vec[i], 2) for i in range(0,
len(data_vec) - 1)]
event =  [i for i in range(0, len(delta_p)) if (delta_p[i] > T_Posi-
tive or delta_p[i] < T_Negative) ]

clusters = gsp.refined_clustering_block(event, delta_p, sigma, ri)

finalclusters, pairs = gsp.pair_clusters_appliance_wise(clusters,
data_vec, delta_p, instancelimit)
appliance_pairs = gsp.feature_matching_module(pairs, delta_p, fi-
nalclusters, alpha, beta)

power_series, appliance_signatures = gsp.generate_appliance_pow-
erseries(appliance_pairs, delta_p)

labeled_appliances = gsp.label_appliances(appliance_signatures, signa-
ture_database, threshold)

power_timeseries = gsp.create_appliance_timeseries(power_series,
main_ind)

gsp_result = pd.concat(power_timeseries, axis = 1)

labels= [i[1] for i in list(labeled_appliances.items())]
gsp_result.columns = labels

axs[2].stackplot(gsp_result.index, gsp_result.values.T, labels=labels)
axs[2].set_title("Disaggregated appliance [Results]", size=8)
axs[2].legend(loc='upper left', fontsize=6)

print("6 of 6> plotting the input and results :)")

disaggregated_building = {'location': 'USA',
                          'type': 'residential',
                          'climate': 'warm temperature dry'}

for i in range(1, len(gsp.disaggregated_appliances)):
    for key, value in gsp.disaggregated_appliances.items():
        disaggregated_building[key] = value
print (disaggregated_building)
plt.show()
gsp.calculate_energy_pct(dfd, gsp_result)
import Recommender as recommender
recommender.recommend(disaggregated_building)
```

# Appendix 2

## Python code for the NILM model (file “gsp_support.py”)

This appendix provides the Python code that was used for the application of the NILM model. It contains the code from the file “gsp_support.py”. This code was obtained from the GitHub repository, and according to the provided description, it was developed based on the MATLAB code that was provided by the authors of [50]. The code is publicly available and in this work it was reused.

```
from __future__ import division
import numpy as np
import pandas as pd
from collections import OrderedDict
from copy import deepcopy
from collections import defaultdict
from scipy.stats import norm
import math
import matplotlib.pyplot as plt
import csv
from IPython.display import display
from math import sqrt
import os

disaggregated_appliances = {}

def gspclustering_event2(event,delta_p,sigma):

  winL = 1000
  Smstar = np.zeros((len(event),1))
  for k in range(0,int(np.floor(len(event)/winL))):
    r = []
    event_1 =  event[k*winL:((k+1)*winL)]
    r.append(delta_p[event[0]])
    [r.append(delta_p[event_1[i]]) for i in range(0,len(event_1))]
    templen = winL + 1
    Sm = np.zeros((templen,1))
    Sm[0] = 1;

    Am = np.zeros((templen,templen))
    for i in range(0,templen):
      for j in range(0,templen):
         Am[i,j] = math.exp(-((r[i]-r[j])/sigma)**2);
    Dm = np.zeros((templen,templen));
    for i in range(templen):
      Dm[i,i] = np.sum(Am[:,i]);
    Lm = Dm - Am;
    Smstar[k*winL:(k+1)*winL] = np.matmul(np.linalg.pinv(Lm[1:tem-
plen,1:templen]), ((-Sm[0].T) * Lm[0,1:templen]).reshape(-1,1));
  if (len(event)%winL > 0):
    r = []
    event_1 =  event[int(np.floor(len(event)/winL))*winL:]
    newlen = len(event_1) + 1
    r.append(delta_p[event[0]])
```

```
    [r.append(delta_p[event_1[i]]) for i in range(0,len(event_1))]
    Sm = np.zeros((newlen,1))
    Sm[0] = 1;
    Am = np.zeros((newlen,newlen))
    for i in range(newlen):
      for j in range(newlen):
         Am[i,j] =  math.exp(-((r[i]-r[j])/sigma)**2);
    Dm = np.zeros((newlen,newlen));
    for i in range(newlen):
      Dm[i,i] = np.sum(Am[:,i]);
    Lm = Dm - Am;
    Smstar_temp = np.matmul(np.linalg.pinv(Lm[1:newlen,1:newlen]), ((-
Sm[0].T) * Lm[0,1:newlen]).reshape(-1,1));
    Smstar[(int(np.floor(len(event)/winL))*winL):len(event)] =
Smstar_temp
  cluster = [event[i] for i in range(len(Smstar)) if (Smstar[i] >
0.98)]
  return cluster

def johntable(clusters,precluster,delta_p,ri):
  import math
  for h in range(0,len(clusters)):
    stds = np.std([delta_p[i] for i in clusters[h]],ddof=1)
    if(math.isnan(stds)):
      stds = 0
    means = np.mean([delta_p[i] for i in clusters[h]])
    if abs(stds/means) <= ri :
      precluster.append([i for i in clusters[h]])
  return precluster

def find_new_events(clusters,delta_p,ri):
  import math
  newevents = []
  for h in range(0,len(clusters)):
    stds = np.std([delta_p[i] for i in clusters[h]],ddof=1)
    if(math.isnan(stds)):
      stds = 0
    means = np.mean([delta_p[i] for i in clusters[h]])
    if abs(stds/means) > ri :
      newevents.append([i for i in clusters[h]])
  newevents = [subitem for item in newevents for subitem in item]
  return newevents

def feature_matching_module(pairs,DelP,Newcluster,alpha,beta):
    appliance_pairs = OrderedDict()
    for i in range(len(pairs)):
      pos_cluster = sorted(Newcluster[pairs[i][0]])
      neg_cluster = sorted(Newcluster[pairs[i][1]])
      flag = 0
      state_pairs = []
      for j in range(len(pos_cluster)):
         if j==len(pos_cluster)-1:
             flag = 1
             start_pos = pos_cluster[j]
         if flag:
             neg_set = [h for h in neg_cluster if (h > start_pos)]
         else:
             start_pos = pos_cluster[j]
             next_pos = pos_cluster[j+1]
             if (next_pos - start_pos) == 1:
                 continue
```

```
            neg_set = [h for h in neg_cluster if (h > start_pos and
h< next_pos)]
        if len(neg_set)==1:
            pair= (start_pos,neg_set[0])
            state_pairs.append(pair)
        elif len(neg_set)==0:
            continue
        else:
            phi_m = [DelP[h]+DelP[start_pos] for h in neg_set]
            phi_t = [(h-start_pos) for h in neg_set]
            newlen= len(neg_set)
            Am = np.zeros((newlen,newlen))
            At = np.zeros((newlen,newlen))
            sigma = 1
            for k in range(newlen):
                for p in range(newlen):
                    Am[k,p] = np.exp(-((phi_m[k]-
phi_m[p])/sigma)**2);
            for k in range(newlen):
                for p in range(newlen):
                    At[k,p] = np.exp(-((phi_t[k]-
phi_t[p])/sigma)**2);
            Dm = np.zeros((newlen,newlen));
            for z in range(newlen):
                Dm[z,z] = np.sum(Am[:,z]);
            Lm = Dm - Am;
            Sm = np.zeros((newlen,1))
            Sm[0] = np.average(phi_m)
            Smstar = np.matmul(np.linalg.pinv(Lm[0:newlen,0:newlen]),
((-Sm[0].T) * Lm[0,0:newlen]).reshape(-1,1))
            Dt = np.zeros((newlen,newlen));
            for z in range(newlen):
                Dt[z,z] = np.sum(At[:,z]);
            Lt = Dt - At;
            St = np.zeros((newlen,1))
            St[0] = np.median(phi_t)
            Ststar = np.matmul(np.linalg.pinv(Lt[0:newlen,0:newlen]),
((-St[0].T) * Lt[0,0:newlen]).reshape(-1,1))
            result_vec = []
            for f in range(Smstar.shape[0]):
                temp = np.nansum([alpha * Smstar[f][0] , beta  *
Ststar[f][0] ])
                result_vec.append(temp)
            best_pos = [a for a in range(len(result_vec)) if (re-
sult_vec[a] == min(result_vec))][0]
            pair = (start_pos,neg_set[best_pos])
            state_pairs.append(pair)
      appliance_pairs[i] = state_pairs
    return appliance_pairs

def generate_appliance_powerseries(appliance_pairs,DelP):
    print ("3 of 6> generates full power series of appliances")
    appliance_signatures = OrderedDict()
    power_series = OrderedDict()
    ctlf = OrderedDict()
    for i in range(len(appliance_pairs)):
        events = appliance_pairs[i]
        timeseq= []
        powerseq  = []
        for event in events:
            start= event[0]
```

```
            end = event[1]
            duration = end - start
            instance = []
            instance.append([DelP[start]])
            temp= np.repeat(np.nan,duration-1).tolist()
            instance.append(temp)
            instance.append([abs(DelP[end])])
            final = [j for sub in instance for j in sub]
            timeval = range(start,end+1,1)
            powerval = interpolate_values(final) if sum(np.isnan(fi-
nal)) else final
            timeseq.append(timeval)
            powerseq.append(powerval)
        powerseq =  [j for sub in powerseq for j in sub]
        timeseq =  [j for sub in timeseq for j in sub]
        power_series[i] = pd.Data-
Frame({'timestamp':timeseq,'power':powerseq})
        appliance_signatures[i] = pd.DataFrame(powerseq)

    return power_series, appliance_signatures


def label_appliances(appliance_signatures, signature_database, thresh-
old):
    print ("4 of 6> checking appliance power signatures matches")
    labeled_appliances = OrderedDict()
    dfw = pd.concat(appliance_signatures, axis = 1, ignore_index=True)
    dfw.drop(dfw.index[1], axis=1)

    dfr = pd.read_csv(signature_database, index_col=0)
    rowr, columnsr = dfr.shape
    roww, columnsw = dfw.shape
    print("        > found "+ str(columnsw) + " appliances. Verifying
signature matching")
    for i in range(columnsw):
        for j in range(columnsr):
            last_idxr = dfr.iloc[:,j].last_valid_index()
            last_idxw = dfw.iloc[:,i].last_valid_index()
            D = FastDTW(dfw.iloc[:last_idxw,i].values,
dfr.iloc[:last_idxr,j].values, 10)
            if D < threshold:
                print("          > found match " + str(i+1) + " with "
+ dfr.iloc[:0,j].name)
                M = 'appliance' + str(i+1)
                disaggregated_appliances[M] = str(dfr.iloc[:0,j].name)
                labeled_appliances[i] = dfr.iloc[:0,j].name
    print (disaggregated_appliances)
    return labeled_appliances

def DTW(s1, s2):
    DTW={}

    for i in range(len(s1)):
        DTW[(i, -1)] = float('inf')
    for i in range(len(s2)):
        DTW[(-1, i)] = float('inf')
    DTW[(-1, -1)] = 0

    for i in range(len(s1)):
        for j in range(len(s2)):
            dist= (s1[i]-s2[j])**2
```

```
            DTW[(i, j)] = dist + min(DTW[(i-1, j)],DTW[(i, j-1)],
DTW[(i-1, j-1)])

    return sqrt(DTW[len(s1)-1, len(s2)-1])

def FastDTW(s1, s2, w):
    DTW={}

    w = max(w, abs(len(s1)-len(s2)))

    for i in range(-1,len(s1)):
        for j in range(-1,len(s2)):
            DTW[(i, j)] = float('inf')
    DTW[(-1, -1)] = 0

    for i in range(len(s1)):
        for j in range(max(0, i-w), min(len(s2), i+w)):
            dist= (s1[i]-s2[j])**2
            DTW[(i, j)] = dist + min(DTW[(i-1, j)],DTW[(i, j-1)],
DTW[(i-1, j-1)])

    return sqrt(DTW[len(s1)-1, len(s2)-1])

def write_csv_df(path, filename, df):
    pathfile = os.path.normpath(os.path.join(path,filename))
    files_present = os.path.isfile(pathfile)
    if not files_present:
        df.to_csv(pathfile)
    else:
        overwrite = raw_input("WARNING: " + pathfile + " already ex-
ists! Overwrite <y/n>? \n ")
        if overwrite == 'y':
            df.to_csv(pathfile)
        elif overwrite == 'n':
            return
        else:
            print "Not a valid input. Data is NOT saved!\n"
    return

def calculate_energy_pct(dfd, dfc):
    fig = plt.figure()
    ax1 = fig.add_axes([0, .3, .5, .5], aspect=1)
    ax2 = fig.add_axes([.5, .3, .5, .5], aspect=1)
    fig.suptitle('Total energy consumption', fontsize = 14)

    cons1 = dfd[dfd.columns.values].sum().sort_values(ascending=False)
    cons2 = dfc[dfc.columns.values].sum().sort_values(ascending=False)

    ax1.pie(cons1.values, autopct='%1.1f%%', startangle=90)
    ax2.pie(cons2.values, autopct='%1.1f%%', startangle=90)
    first_legend = ax1.legend(dfd.columns, loc = 'lower center',
bbox_to_anchor=(.5, -.4), fontsize = 8)
    second_legend = ax2.legend(dfc.columns, loc = 'lower center',
bbox_to_anchor=(.5, -.4), fontsize = 8)
    ax1.set_title('Ground truth')
    ax2.set_title('Disaggregated')
    ax1.axis('equal')
    ax2.axis('equal')
    plt.tight_layout()
    plt.show()
```

```
def interpolate_values(A):
    if type(A) ==list :
        A= np.array(A)
    ok = ~np.isnan(A)
    xp = ok.nonzero()[0]
    fp = A[~np.isnan(A)]
    x  = np.isnan(A).nonzero()[0]
    A[np.isnan(A)] = np.interp(x, xp, fp)
    A = [round(i) for i in A]
    return A

def create_appliance_timeseries(power_series,main_ind):
    print ("5 of 6> creating appliance power timeseries")
    result = OrderedDict()
    for i in range(len(power_series)):
        temp = power_series[i]
        if len(temp) < 1:
            continue
        temp.index = temp.timestamp
        dummy = pd.Series(0,main_ind)
        dummy = dummy.loc[~dummy.index.duplicated(keep='first')]
        dummy[main_ind[temp.index.values]] = temp.power.values
        result[i] = dummy
    return(result)

def refined_clustering_block(event,delta_p,sigma,ri):
    sigmas =
[sigma,sigma/2,sigma/4,sigma/8,sigma/14,sigma/32,sigma/64]
    Finalcluster = []
    for k in range(0,len(sigmas)):
        clusters = []
        event = sorted(list(set(event)-set(clusters)))
        while len(event):
            clus =  gspclustering_event2(event,delta_p,sigmas[k]);
            clusters.append(clus)
            event = sorted(list(set(event)-set(clus)))
        if k == len(sigmas)-1:
            Finalcluster = Finalcluster + clusters
        else:
            jt = johntable(clusters,Finalcluster,delta_p,ri)
            Finalcluster = jt
            events_updated = find_new_events(clusters,delta_p,ri)
            events_updated = sorted(events_updated)
            event = events_updated
    if len(event) > 0:
      Finalcluster.append(event)
    return Finalcluster

def find_closest_pair(cluster_means,cluster_group):
    distances = []
    for i in range(len(cluster_means)-1):
        for j in range((i+1),len(cluster_means)):
           distance = abs(cluster_means[i] - cluster_means[j])
           distances.append((i,j,distance))
    merge_pair = min(distances, key = lambda h:h[2])
    cluster_dict = {}
    for i in range(len(cluster_group)):
        cluster_dict[i] =  cluster_group[i]
    tempcluster = []
    tempcluster.append(cluster_dict[merge_pair[0]] + clus-
ter_dict[merge_pair[1]])
```

```
    del cluster_dict[merge_pair[0]]
    del cluster_dict[merge_pair[1]]
    for k,v in cluster_dict.items():
        tempcluster.append(v)
    return tempcluster

def pair_clusters_appliance_wise(Finalcluster, data_vec, delta_p, in-
stancelimit):
    print ("2 of 6> pair clusters appliance wise")
    Table_1 =  np.zeros((len(Finalcluster),4))
    for i in range(len(Finalcluster)):
      Table_1[i,0] = len(Finalcluster[i])
      Table_1[i,1] = np.mean([delta_p[j] for j in Finalcluster[i]])
      Table_1[i,2] = np.std([delta_p[j] for j in Finalclus-
ter[i]],ddof=1)
      Table_1[i,3] =  abs(Table_1[i,2]/ Table_1[i,1])
    sort_means = np.argsort(Table_1[:,1]).tolist()
    sort_means.reverse()
    sorted_cluster =[]
    FinalTable = []
    for i in range(len(sort_means)):
      sorted_cluster.append(Finalcluster[sort_means[i]])
      FinalTable.append(Table_1[sort_means[i]].tolist())

    DelP = [round(data_vec[i+1]-data_vec[i],2) for i in
range(0,len(data_vec)-1)]
    Newcluster_1 = []
    Newtable = []
    for i in range(0,len(FinalTable)):
      if (FinalTable[i][0] >= instancelimit):
        Newcluster_1.append(sorted_cluster[i])
        Newtable.append(FinalTable[i])
    Newcluster = Newcluster_1
    for i in range(0,len(FinalTable)):
      if(FinalTable[i][0] < instancelimit ):
        for j in range(len(sorted_cluster[i])):
          count =  []
          for k in range(len(Newcluster)):
            count.append(norm.pdf(DelP[sorted_cluster[i][j]],New-
table[k][1],Newtable[k][2]))
          asv = [h == max(count) for h in count]
          if sum(asv) == 1:
            johnIndex = count.index(max(count))
          elif DelP[sorted_cluster[i][j]] > 0:
            tablemeans = [r[1] for r in Newtable]
            tempelem = [r for r in tablemeans if r < DelP[sorted_clus-
ter[i][j]]][0]
            johnIndex = tablemeans.index(tempelem)
          else:
            tablemeans = [r[1] for r in Newtable]
            tempelem = [r for r in tablemeans if r > DelP[sorted_clus-
ter[i][j]]].pop()
            johnIndex = tablemeans.index(tempelem)
          Newcluster[johnIndex].append(sorted_cluster[i][j])
    Table_2 =  np.zeros((len(Newcluster),4))
    for i in range(len(Newcluster)):
      Table_2[i,0] = len(Newcluster[i])
      Table_2[i,1] = np.mean([delta_p[j] for j in Newcluster[i]])
      Table_2[i,2] = np.std([delta_p[j] for j in Newclus-
ter[i]],ddof=1)
      Table_2[i,3] =  abs(Table_2[i,2]/ Table_2[i,1])
```

```
    Newtable = Table_2

    pos_clusters = neg_clusters = 0
    for i in range(Newtable.shape[0]):
        if Newtable[i][1] > 0:
            pos_clusters += 1
        else:
            neg_clusters += 1
    Newcluster_cp = deepcopy(Newcluster)
    while pos_clusters != neg_clusters:
        index_cluster = Newcluster_cp
        power_cluster = []
        for i in index_cluster:
            list_member = []
            for j in i:
                list_member.append(delta_p[j])
            power_cluster.append(list_member)

        clustermeans = [np.mean(i) for i in power_cluster]
        positive_cluster_chunk= []
        negative_cluster_chunk = []
        positive_cluster_means= []
        negative_cluster_means = []
        pos_clusters = neg_clusters = 0
        for j in range(len(clustermeans)):
           if clustermeans[j] > 0:
                pos_clusters += 1
                positive_cluster_chunk.append(index_cluster[j])
                positive_cluster_means.append(clustermeans[j])
           else:
                neg_clusters += 1
                negative_cluster_chunk.append(index_cluster[j])
                negative_cluster_means.append(clustermeans[j])

        if pos_clusters > neg_clusters:
             positive_cluster_chunk = find_closest_pair(positive_clus-
ter_means, positive_cluster_chunk)
        elif neg_clusters > pos_clusters:
             negative_cluster_chunk = find_closest_pair(negative_clus-
ter_means, negative_cluster_chunk)
        else:
            pass
        Newcluster_cp = positive_cluster_chunk + negative_clus-
ter_chunk

    clus_means = []
    for i in Newcluster_cp:
        list_member = []
        for j in i:
            list_member.append(delta_p[j])
        clus_means.append(np.mean(list_member))
    pairs = []
    for i in range(len(clus_means)):
      if clus_means[i] > 0:
        neg_edges = [ (abs(clus_means[i] + clus_means[j]),j) for j in
range(i+1,len(clus_means)) if clus_means[j] < 0]
        edge_mag = [j[0] for j in neg_edges]
        match_loc = neg_edges[edge_mag.index(min(edge_mag))][1]
        pairs.append((i,match_loc))
    dic_def = defaultdict(list)
    for value,key in pairs:
```

```
        dic_def[key].append(value)

    updated_pairs= []
    for neg_edge in dic_def.keys():
        pos_edges = dic_def[neg_edge]
        if len(pos_edges) >1:
            candidates = [abs(clus_means[edge]+ clus_means[neg_edge])
for edge in pos_edges]
            good_pos_edge =  [el_pos for el_pos in range(len(candi-
dates)) if candidates[el_pos] == min(candidates)][0]
            good_pair = (pos_edges[good_pos_edge],neg_edge)
        else:
            good_pair = (pos_edges[0],neg_edge)
        updated_pairs.append(good_pair)
    return Newcluster_cp,updated_pairs

def find_closest_pairs(start_cluster,end_cluster,cluster_means,re-
quired_reduction):
    distances = []
    for i in range(start_cluster, end_cluster):
        for j in range((i+1),end_cluster+1):
           distance = abs(cluster_means[i] - cluster_means[j])
           distances.append((i,j,distance))
    distances  = pd.DataFrame.from_records(distances)
    distances.columns = ['cluster_1','cluster_2','difference']
    distances.sort_values('difference',axis=0,inplace=True)
    return distances.head(required_reduction)
```

# Appendix 3

## Buildings database

This appendix provides a screenshot of the database that was formed for the validation of the RS algorithm.

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | **building number** | **name** | **type** | **location** | **climate** |
| 2 | building_1 | Hotel Me (Madrid) | hotel | Spain | warm temperature dry |
| 3 | building_2 | Plaza De Las Cortes 3 (Madrid) | residential | Spain | warm temperature dry |
| 4 | building_3 | Centro Civico Aldabe (Vitoria-Gasteiz) | office | Spain | cool temperature |
| 5 | building_4 | W Hotel City Center (Chicago) | hotel | USA | cool temperature |
| 6 | building_5 | 150 Powell Street | residential | USA | warm temperature dry |
| 7 | building_6 | Travis Tower (Houston) | office | USA | sub tropical |
| 8 | building_7 | Domaine Du Mandravasarotra (Belobaka) | hotel | Madagascar | tropical |
| 9 | building_8 | Kk Home Tomasina (Toamasina) | residential | Madagascar | sub tropical |
| 10 | building_9 | NSI Office (Antananarivo) | office | Madagascar | warm temperature dry |
| 11 | building_10 | Embassy Suites Hotel Houston/Downtown (Houston) | hotel | USA | sub tropical |
| 12 | building_11 | Hotel Dnipro (Kyiv) | hotel | Ukraine | cool temperature |
| 13 | building_12 | Admiralty Arch (London) | office | UK | warm temperature dry |
| 14 | building_13 | The Palace Hotel (San Francisco) | hotel | USA | warm temperature dry |
| 15 | building_14 | Objet Deco (Mahajanga) | residential | Madagascar | tropical |
| 16 | building_15 | Horizon Office Tower (Kyiv) | office | Ukraine | cool temperature |
| 17 | building_16 | Sofitel Madrid Plaza De Espana (Madrid) | hotel | Spain | warm temperature dry |
| 18 | building_17 | One Thousand Powell Apartments (San Francisco) | residential | USA | warm temperature dry |
| 19 | building_18 | Adlington House (Liverpool) | residential | UK | cool temperature |
| 20 | building_19 | Vulytsia Vorovskogo 11 (Kyiv) | residential | Ukraine | cool temperature |
| 21 | building_20 | Gulliver (Kyiv) | office | Ukraine | cool temperature |

# Appendix 4

## Ratings of the recommendations

This appendix provides a screenshot of the ratings that were provided by each of the buildings in the database for each recommendation.

| | A | B | C | D |
|---|---|---|---|---|
| 1 | building number | Place the refrigerator away from heat sources | Avoid putting hot food directly in the refrigerator | Try to keep the refrigerator filled in to save energy |
| 2 | building_1 | 2 | 2 | 3 |
| 3 | building_2 | 2 | 3 | 2 |
| 4 | building_3 | 3 | 2 | 2 |
| 5 | building_4 | 3 | 2 | 3 |
| 6 | building_5 | 2 | 3 | 2 |
| 7 | building_6 | 2 | 2 | 2 |
| 8 | building_7 | 2 | 2 | 2 |
| 9 | building_8 | 2 | 3 | 1 |
| 10 | building_9 | 2 | 2 | 1 |
| 11 | building_10 | 2 | 2 | 3 |
| 12 | building_11 | 3 | 2 | 2 |
| 13 | building_12 | 2 | 2 | 2 |
| 14 | building_13 | 2 | 2 | 3 |
| 15 | building_14 | 2 | 3 | 1 |
| 16 | building_15 | 3 | 2 | 1 |
| 17 | building_16 | 2 | 2 | 3 |
| 18 | building_17 | 2 | 3 | 2 |
| 19 | building_18 | 3 | 3 | 2 |
| 20 | building_19 | 3 | 3 | 1 |
| 21 | building_20 | 3 | 2 | 1 |

Refrigerator | Microwave | Lighting | Stove | Air conditioner | Wash

| | A | B | C |
|---|---|---|---|
| 1 | building number | Cover the dishes before putting them in the microwave to cut down the cooking time | Cut the food into small pieces to reduce the cooking time |
| 2 | building_1 | 3 | 2 |
| 3 | building_2 | 3 | 3 |
| 4 | building_3 | 2 | 2 |
| 5 | building_4 | 3 | 2 |
| 6 | building_5 | 3 | 3 |
| 7 | building_6 | 2 | 2 |
| 8 | building_7 | 3 | 2 |
| 9 | building_8 | 3 | 3 |
| 10 | building_9 | 2 | 2 |
| 11 | building_10 | 3 | 2 |
| 12 | building_11 | 3 | 2 |
| 13 | building_12 | 2 | 2 |
| 14 | building_13 | 3 | 2 |
| 15 | building_14 | 3 | 3 |
| 16 | building_15 | 2 | 2 |
| 17 | building_16 | 3 | 2 |
| 18 | building_17 | 3 | 3 |
| 19 | building_18 | 3 | 3 |
| 20 | building_19 | 3 | 3 |
| 21 | building_20 | 2 | 2 |

Refrigerator | **Microwave** | Lighting | Stove | Air conditioner | Washing n

| | A | B | C | D |
|---|---|---|---|---|
| 1 | building number | Place movement detectors to turn off the lights when the room is empty | Install task lightings in places like on the study desk etc. to reduce the electricity consumption from using the general lighting | Consider using light coloured paint |
| 2 | building_1 | 3 | 2 | 1 |
| 3 | building_2 | 3 | 3 | 2 |
| 4 | building_3 | 3 | 2 | 3 |
| 5 | building_4 | 3 | 2 | 2 |
| 6 | building_5 | 3 | 3 | 2 |
| 7 | building_6 | 3 | 2 | 2 |
| 8 | building_7 | 2 | 1 | 0 |
| 9 | building_8 | 2 | 2 | 1 |
| 10 | building_9 | 2 | 1 | 1 |
| 11 | building_10 | 3 | 2 | 1 |
| 12 | building_11 | 2 | 1 | 0 |
| 13 | building_12 | 3 | 2 | 2 |
| 14 | building_13 | 3 | 2 | 1 |
| 15 | building_14 | 2 | 2 | 1 |
| 16 | building_15 | 2 | 1 | 2 |
| 17 | building_16 | 3 | 2 | 1 |
| 18 | building_17 | 3 | 3 | 2 |
| 19 | building_18 | 3 | 3 | 3 |
| 20 | building_19 | 2 | 2 | 2 |
| 21 | building_20 | 2 | 1 | 2 |

Refrigerator | Microwave | **Lighting** | Stove | Air conditioner | Washing machine | Water heater

| | A | B | C |
|---|---|---|---|
| 1 | building number | Shift the usage of electrical stove to the late hours | Take into account the heating area of your stove to choose pans with proper diameters |
| 2 | building_1 | 3 | 3 |
| 3 | building_2 | 2 | 3 |
| 4 | building_3 | 2 | 2 |
| 5 | building_4 | 3 | 3 |
| 6 | building_5 | 2 | 3 |
| 7 | building_6 | 2 | 2 |
| 8 | building_7 | 3 | 2 |
| 9 | building_8 | 2 | 2 |
| 10 | building_9 | 2 | 1 |
| 11 | building_10 | 3 | 3 |
| 12 | building_11 | 3 | 2 |
| 13 | building_12 | 2 | 2 |
| 14 | building_13 | 3 | 3 |
| 15 | building_14 | 2 | 2 |
| 16 | building_15 | 2 | 1 |
| 17 | building_16 | 3 | 3 |
| 18 | building_17 | 2 | 3 |
| 19 | building_18 | 2 | 3 |
| 20 | building_19 | 2 | 2 |
| 21 | building_20 | 2 | 1 |

Refrigerator | Microwave | Lighting | **Stove** | Air conditione

| | A | B | C |
|---|---|---|---|
| 1 | building number | Keep the curtains and blinds closed to reduce the space from heating up | Use a programmable thermostat that turns off the AC when the space is empty |
| 2 | building_1 | 2 | 3 |
| 3 | building_2 | 3 | 3 |
| 4 | building_3 | 1 | 3 |
| 5 | building_4 | 1 | 3 |
| 6 | building_5 | 3 | 3 |
| 7 | building_6 | 2 | 3 |
| 8 | building_7 | 2 | 2 |
| 9 | building_8 | 3 | 2 |
| 10 | building_9 | 2 | 2 |
| 11 | building_10 | 2 | 3 |
| 12 | building_11 | 1 | 2 |
| 13 | building_12 | 2 | 3 |
| 14 | building_13 | 2 | 3 |
| 15 | building_14 | 3 | 2 |
| 16 | building_15 | 1 | 2 |
| 17 | building_16 | 2 | 3 |
| 18 | building_17 | 3 | 3 |
| 19 | building_18 | 2 | 3 |
| 20 | building_19 | 2 | 2 |
| 21 | building_20 | 1 | 2 |
| 22 | | | |

Refrigerator | Microwave | Lighting | Stove | **Air conditioner** | Washir

| | A | B | C | D |
|---|---|---|---|---|
| 1 | building number | Use the washing machine of the right size since the bigger the machine is, the more power it consumes | Use front load washing machines which consume less electricity than the top load | Do not leave the machine in standby mode |
| 2 | building_1 | 3 | 3 | 2 |
| 3 | building_2 | 2 | 3 | 3 |
| 4 | building_3 | 2 | 3 | 3 |
| 5 | building_4 | 3 | 3 | 2 |
| 6 | building_5 | 2 | 3 | 3 |
| 7 | building_6 | 2 | 3 | 3 |
| 8 | building_7 | 2 | 2 | 2 |
| 9 | building_8 | 1 | 2 | 3 |
| 10 | building_9 | 1 | 2 | 3 |
| 11 | building_10 | 3 | 3 | 2 |
| 12 | building_11 | 2 | 2 | 2 |
| 13 | building_12 | 2 | 3 | 3 |
| 14 | building_13 | 3 | 3 | 2 |
| 15 | building_14 | 1 | 2 | 3 |
| 16 | building_15 | 1 | 2 | 3 |
| 17 | building_16 | 3 | 3 | 2 |
| 18 | building_17 | 2 | 3 | 3 |
| 19 | building_18 | 2 | 3 | 3 |
| 20 | building_19 | 1 | 2 | 3 |
| 21 | building_20 | 1 | 2 | 3 |

Refrigerator | Microwave | Lighting | Stove | Air conditioner | **Washing machine** | Water heater

| | A | B | C | D |
|---|---|---|---|---|
| 1 | building number | Insulate the pipes connected to the heater | Prefer taking a short shower instead of a bath | Consider installing heat traps on the water heater |
| 2 | building_1 | 2 | 3 | 2 |
| 3 | building_2 | 2 | 3 | 2 |
| 4 | building_3 | 3 | 2 | 3 |
| 5 | building_4 | 3 | 3 | 3 |
| 6 | building_5 | 2 | 3 | 2 |
| 7 | building_6 | 2 | 2 | 2 |
| 8 | building_7 | 2 | 3 | 1 |
| 9 | building_8 | 2 | 3 | 1 |
| 10 | building_9 | 2 | 2 | 1 |
| 11 | building_10 | 2 | 3 | 2 |
| 12 | building_11 | 3 | 3 | 2 |
| 13 | building_12 | 2 | 2 | 2 |
| 14 | building_13 | 2 | 3 | 2 |
| 15 | building_14 | 2 | 3 | 1 |
| 16 | building_15 | 3 | 2 | 2 |
| 17 | building_16 | 2 | 3 | 2 |
| 18 | building_17 | 2 | 3 | 2 |
| 19 | building_18 | 3 | 3 | 3 |
| 20 | building_19 | 3 | 3 | 2 |
| 21 | building_20 | 3 | 2 | 2 |

Refrigerator | Microwave | Lighting | Stove | Air conditioner | Washing machine | **Water heater**

# Appendix 5

## Python code for the RS (file “Recommender.py”)

This appendix provides the Python code that was used for the application of the collaborative-filtering part of the RS algorithm. It contains the code from the file “Recommender.py”. This code was written specifically for the purpose of this work.

```
import pandas as pd
import glob
import openpyxl
import math

data = './My_buildings_database.csv'
db = pd.read_csv(data)
my_database = {}
for i in range(0, len(db)):
    my_database[db.iloc[i]['building number']] = {}
    my_database[db.iloc[i]['building number']]['type'] =
db.iloc[i]['type']
    my_database[db.iloc[i]['building number']]['location'] =
db.iloc[i]['location']
    my_database[db.iloc[i]['building number']]['climate'] =
db.iloc[i]['climate']
    my_database[db.iloc[i]['building number']]['name'] =
db.iloc[i]['name']

path = './Ratings_database.xlsx'
files = glob.glob(path)
for file in files:
    wb = openpyxl.load_workbook(file)
sheets = wb.sheetnames

for i in range(0, len(db)):
    my_database[db.iloc[i]['building number']]['recommendations'] = {}

for i in range(0, len(sheets)):                          #for each ex-
cel sheet, do the following
    appl = pd.read_excel(path, sheets[i])

    for k in range(0, len(appl)):                          #for each
building, do the following
        my_database[appl.iloc[k][0]]['recommenda-
tions'][str(sheets[i])] = {}
        for n in range(1, len(appl.columns)):            #for each rec-
ommendation, do the following
            if math.isnan(appl.iloc[k][n]) == False:     #if the rank-
ing value is not empty, do the following
                my_database[appl.iloc[k][0]]['recommenda-
tions'][str(sheets[i])][appl.keys()[n]] = appl.iloc[k][n]
        if my_database[appl.iloc[k][0]]['recommenda-
tions'][str(sheets[i])] == {}:
            del my_database[appl.iloc[k][0]]['recommenda-
tions'][str(sheets[i])]
print (my_database)
```

# Appendix 6

## Python code for the RS (file “Database_formming.py”)

This appendix provides the Python code that was used for the application of the collaborative-filtering part of the RS algorithm. It contains the code from the file “Database_forming.py”. This code was written specifically for the purpose of this work.

```
import operator
from Database_forming import my_database


def recommend(considered_building):
    print ("HERE RECOMMENDER STARTS")
    print (considered_building)
    # save the characteristics of the considered building into variables
    my_location = considered_building.get('location')
    print ("MY_LOCATION")
    print (my_location)
    my_type = considered_building.get('type')
    my_climate = considered_building.get('climate')

    # char_num - number of characteristics of the considered building
    char_num = len(considered_building.items())
    my_appliances = []
    for char, info in considered_building.items():
        for i in range(0, char_num):
            appliance_num = 'appliance' + str(i)
            if appliance_num in char:
                my_appliances.append(info)
    print (my_appliances)

    matching_buildings = []
    # buildings_number = len(buildings)
    buildings_number = len(my_database)

    # loop over all buildings verifying if the characteristics of each building match the caracteristics variables of the considered building
    for i in range(1, buildings_number + 1):
        b = 'building_' + str(i)
        loc_b = my_database[b]['location']
        type_b = my_database[b]['type']
        climate_b = my_database[b]['climate']
        if loc_b == my_location:
            matching_buildings.append(b)
        if type_b == my_type:
            matching_buildings.append(b)
        if climate_b == my_climate:
            matching_buildings.append(b)
    print ('MATCHING BUILDINGS')
    print (matching_buildings)

    # 'matching_buildings' - all the builings that have at least one
```

```
characteristic in common with the considered building
    if not matching_buildings:
        print ('NO MATCHING BUILDINGS')
    else:
        # save the buildings with the highest number of characteris-
tics that match the considered building into 'best_buildings'
        dic = {}
        for num in matching_buildings:
            if num in dic:
                dic[num] += 1
            else:
                dic[num] = 1
        vals = max(dic.values())
        best_buildings = [k for k, v in dic.items() if v == vals]
        print ("BEST BUILDINGS")
        print (best_buildings)
        print (len(best_buildings))

        best_buildings_number = len(best_buildings)
        all_recommendations = {}
        # put all the recommendations of the similar buildings into
one dictionary
        for i in range(0, best_buildings_number):
            recommendations = my_database[best_buildings[i]]['recom-
mendations']
            # mutual_appliances = set(considered_building.val-
ues()).intersection(recommendations)
            for dk, dc in recommendations.items():
                if dk not in all_recommendations:
                    all_recommendations[dk] = {k: [v] for k, v in
dc.items()}
                else:
                    for k, v in dc.items():
                        if k in all_recommendations[dk]:
                            all_recommendations[dk][k].append(v)
                        else:
                            all_recommendations[dk][k] = [v]
        # 'all_recommendations' - all recommendations from the most
similar buildings merged into one dictionary
        # loop over all the recommendations from the most similar
buildings and put the recommendations for the appliances in the con-
sidered building into 'matching_appliances'
        matching_recommendations = {}
        for k, v in all_recommendations.items():
            for i in my_appliances:
                if i == k:
                    matching_recommendations[k] = v

        # remove the appliance labeling from the mathing recommenda-
tions
        pure_recc = []
        for appl, rec in matching_recommendations.items():
            pure_recc.append(rec.items())
            values = rec.values()

        # recalculate the ratings by finding the average values and
form the recommendations with new ratings in 'new_recc' dictionary
        new_recc = {}
        for list in pure_recc:
            for statement in list:
                ratings = statement[1]
```

```
                pure_statement = statement[0]
                new_ratings = sum(ratings) / len(ratings)
                print (pure_statement)
                print (new_ratings)
                new_recc[pure_statement] = new_ratings
        print (new_recc)

        new_rec_sorted = sorted(new_recc.items(), key=opera-
tor.itemgetter(1), reverse=True)
        print (new_rec_sorted)

        final_recommendations = []

        for key, value in new_rec_sorted:
            final_recommendations.append(key)

        print (final_recommendations)
```