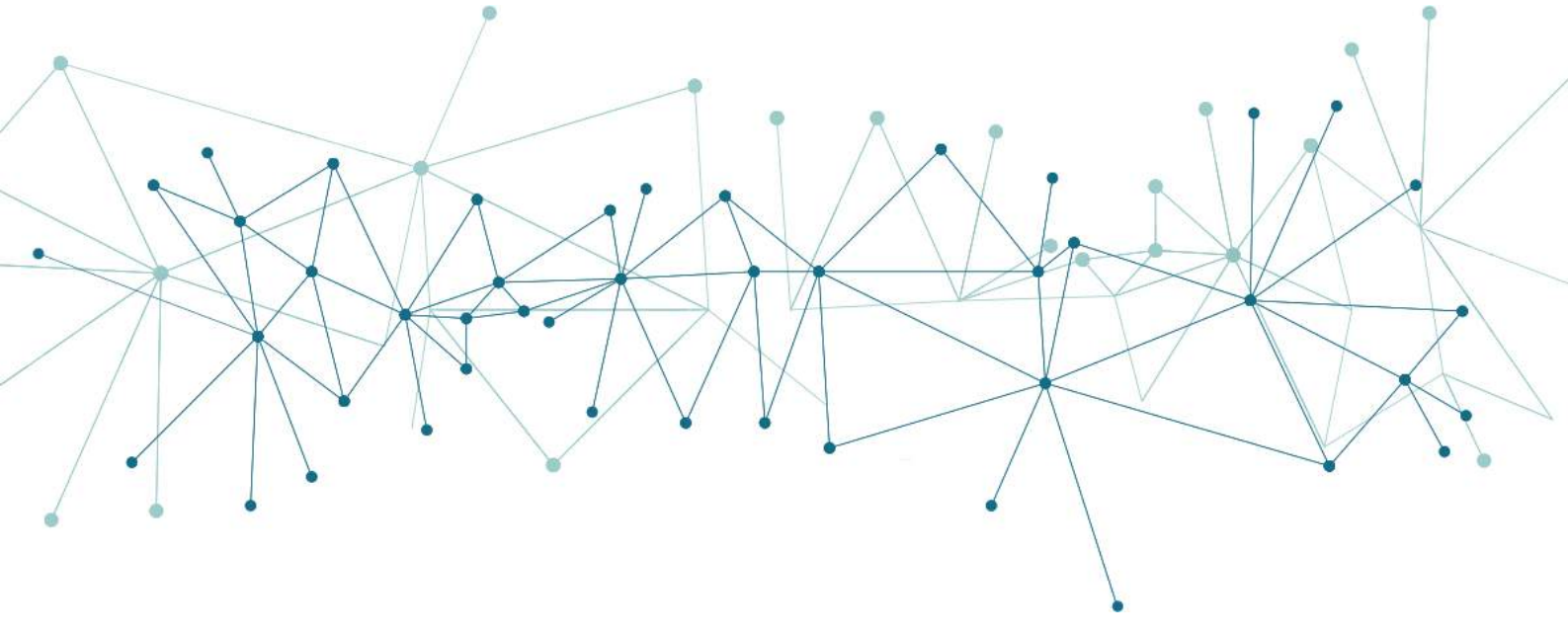
The eDREAM project is co-founded by the EU's Horizon 2020 innovation programme under grant agreement No 774478

enabling new Demand REsponse Advanced, Market oriented and secure technologies, solutions and business models

DELIVERABLE: LOAD PROFILES AND CUSTOMER CLUSTERS V1

Authors:

Ugo Stecchi (ATOS), Lourdes Gallego (ATOS), Javier Gomez (ATOS)



Imprint

LOAD PROFILES AND CUSTOMER CLUSTERS V1, May 2019

Contractual Date of Delivery to the EC:	31.05.2019
Actual Date of Delivery to the EC:	31.05.2019
Author(s):	Ugo Stecchi (Atos), Javier Gomez (Atos), Lourdes Gallego Miguel (Atos), Antigoni Noula (Certh/Iti), Dimos Ioannidis (CERTH), Napoleon Bezas (certh/Iti), Angelo Cardellicchio (E@W), Giuseppe Mastrandrea (E@W), Luigi D'oriano (E@W), Francesca Santori (ASM), Alessio Cavadenti (ASM), Tommaso Bragatto (ASM), Mircea Bucur (Kiwi), Dara Kolajo (Kiwi).
Participant(s):	ATOS, E@W, CERTH, ASM, KIWI
Project:	enabling new Demand Response Advanced, Market oriented and secure technologies, solutions and business models (eDREAM)
Work package:	WP4 – Next generation DR Services for Aggregators and Customers
Task:	4.2 – Big Data Clustering Techniques for load profiling and customer segmentation
Confidentiality:	public
Version:	1.0

Legal Disclaimer

The project enabling new Demand Response Advanced, Market oriented and secure technologies, solutions and business models (eDREAM) has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 774478. The sole responsibility for the content of this publication lies with the authors. It does not necessarily reflect the opinion of the Innovation and Networks Executive Agency (INEA) or the European Commission (EC). INEA or the EC are not responsible for any use that may be made of the information contained therein.

Copyright

Executive Summary

The deliverable D4.2 *"Load Profile and Customer Clusters V1"* is related to the task 4.2 *"Big Data Clustering techniques for load profiling and customer segmentation"*. It describes the techniques and methodologies for extracting load and generation profiles of prosumers and for dividing the prosumers portfolio in clusters, according to specific objectives. Task 4.2 includes the development of three modules of eDREAM platform architecture part of the architectural layer called "*Next Generation Services for Aggregators and Customers*": "*Load Profiling*", "*Big Data Clustering at Multiple Scale*" and "*Customer Segmentation*". They form the so-called Big Data Layer, a portion of the entire platform devoted to managing and provide services in big data domain.

This document describes the techniques and methodologies to be adopted for the components' development and it is organized as follows. Chapter 1 describes the overall methodology for the components' integration, their role in the eDREAM platform and how they interact with the rest of modules. This information is used to design the main process of profiling and clusterization, establishing the different flows across the modules taking into account the architectural requirements and the use cases described in T2.2 and T2.4. Chapter 2 aims at defining the so-called Big Data layer and it describes the adopted tools to implement such layer and it is designed to achieve some stability and scalability requirements.

Chapter 3 describes the pre-processing activities, all those tasks devoted to filter and clean signals arriving from the field, in order to achieve a given data quality for analysis processes. It is a horizontal sub-task and it is preparatory for all the calculations described in following chapters.

Chapter 4, 5 and 7 describe the profiling, the clusterization and the segmentation respectively. Each chapter includes a description of the process, the proposed techniques and tools and the expected goals. Finally, Chapter 7 provides a brief description of the pilots where those applications will be tested and validated.

Table of Contents

List of Figures

List of Tables

List of Acronyms and Abbreviations

ANN	Artificial Neural Network
ART	Adaptive Resonance Theory
DR	Demand Response
DSS	Decision Support System
EM	Expectation Maximization
ESD	Extreme Studentized Deviate
GMM	Gauss Mixture Model
HL	High Level Use Case
IID	Identically and Independently Distributed
KDE	Kernel Density Estimation
KPI	Keep Performance Indicator
LL	Low Level Use Case
LV	Low Voltage
MAD	Mean Absolute Deviation
MSC	Mean Shift Clustering
MV	Medium Voltage
PCA	Principal Component Analysis
RFE	Recursive Feature Elimination
RMSE	Root Mean Square Error
SARIMA	Seasonal Auto Regressive Integrated Moving Average
SM	Smart Meter
SSE	Sum of Squared Errors
STL	Seasonal Trend decomposition using Loess
VPP	Virtual Power Plan

1. Introduction

This document is a technical deliverable that reports the big data analytics solutions for the eDREAM platform. This is the first version of the deliverable associated to Task 4.2 "*Big data clustering techniques for load profiling and customer segmentation*". The task is aimed at designing and implementing big data services based on the WP3 software components, in order to contribute to the development of useful tools for demand response applications. The final objective will be the integration of a big data services into the core modular platform, able to able to extract energy profiles, prosumers clusters and customers segments starting from pre-processed data coming from other eDREAM components like energy forecast, VPP generation modelling and forecast, etc. In the same way the output of this task will be useful for further analysis in the same architectural layer or for the flexibility management.

In particular, this deliverable is devoted to the investigation and identification of big data techniques for the flexibility analysis of prosumers and the development of three specific tools: load profiling, big data clusterization at multiple scale and customer segmentation. These modules will be implemented and deployed into an architectural layer with specific non-functional requirements like stability and scalability. This deliverable refers to the first part of the task (from month 9 to month 17) where the preparatory work is done, while in the final part of the task another deliverable (D4.6) will take care of actual development of the solutions.

The successful criteria for this deliverable are the identification of the proper solutions, considering:

- mutual dependencies among software modules;
- specific requirements asked to the task;
- ad-hoc solutions for supporting flexibility management;
- features and characteristics of the pilot site where those tools will be tested.

Thus, this task is strongly dependent from T2.4 where the overall architecture is described, and the first versions of the use cases try to envision the application framework of the components. From WP3 tasks 3.1, 3.3 and 3.4 will provide data input for a second round of analysis to be performed. At the same time Decision Support System & Demand Response Strategies Optimization modules will depend by the output of this task, by receiving the results of clusterization.

1.1. Methodology

The methodology (or methodologies) adopted for the whole clustering & segmentation processes are outlined below, describing the relation between use cases in each component and identifying requisition data format including its inputs and outputs connections. Basically, the whole process must be intended as a complex procedure where different components cooperate to solve more problems. The combination of the possible interactions among these tools, together with the range of inputs and outputs allows answering several petitions and providing the clusterization service for different aims. The involved components as defined in the eDREAM architecture from deliverable D2.4 are: "*Big Data Clustering at multiple scale*", "*Load Profiling and Disaggregation*" and "*Customer Segmentation*". The operation of the three components and their mutual interaction can be derived from D2.4.

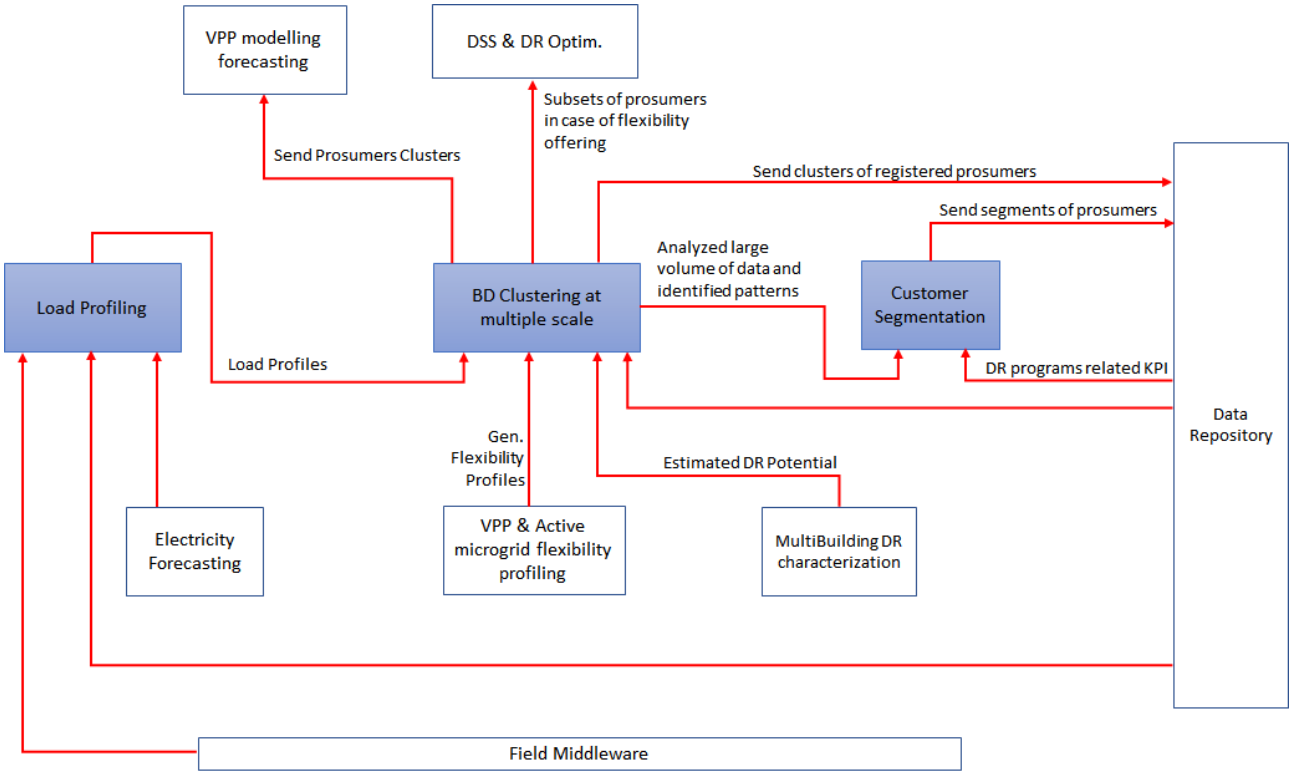


Figure 1: Interconnections among T4.2 components (blue boxes) and other platform's components (white boxes) as represented in eDREAM Architecture

Architectural Dependencies

The Big Data Clustering at Multiple scale component holds the central spot of the big data layer and it is the module with the higher number of interconnections (see Figure 1). It will receive the prosumers load profiles from the component Load Profiling and the batch data from the repository for clusterization based on historical measurements.

The objective of this component is to provide several well-defined and separate clusters of prosumers based on different features like: type of customers, time interval, maximum consumption/production, etc; in a scalable environment. The operation of the three components and their mutual interaction can be derived from the Deliverable 2.4.

Moreover, it will receive inputs from the VPP and active Microgrid Flexibility Profiling that will send the flexibility margins of the prosumers generation assets. In particular, this component receives two arrays from Baseline Flexibility Estimation module; the estimated energy consumption flexibility and estimated energy production flexibility values of the prosumers and then it calculates the margins. The component also receives the optimal coalition of prosumers connected to the VPP from the "*VPP Generation, Modelling & Forecasting*" and it provides as output an array of the flexibility margins of the prosumers that are part of the VPP.

Furthermore, the Multi-building DR characterization module through thermal, optical and LIDAR information fusion can send the potential flexibility of new prosumers by matching thermography aerial scan and smart metering values. All those should be evaluated for estimating the energy consumption profiles of the customers.

The Big Data Clustering at multiple scale sends the identified subset to the Decision Support System & DR Strategies Optimization for periodic process for creating segments into the prosumers pool, based on their characteristics. At the same time, it will send the results to Customer Segmentation in order to assign new set of prosumers to the calculated clusters.

Load Profiling component is responsible of:

- detecting load profile patterns,
- extracting prosumer load profile and
- providing customer profiles based on historical data or even on near real time data, if available.

Load Profiling has an easier data flow, because it receives the load and generation measurements from the field devices[1] or from the data repository (when historic data are considered) and it sends its output to the Big Data Clustering at Multiple scale module. It can receive forecasted data as well, but the data flow won't change.

Finally, the Customers Segmentation is responsible for recognizing the customer´s load profile pattern clustered in Big Data and assign it to them. Prosumer´s segmentation could be useful to categorize the participation of small and medium generation to different energy markets. It receives created prosumers' clusters from Big Data Clustering at Multiple Scales and related KPIs from the Decentralized Repository and after the calculation, it sends the customers segments for secure storage.

Use Cases Dependencies

The actual operation of these modules is planned in the use cases description (D2.2 and D2.4) where dependencies to/from other components and data sets are described in two of the three High Level Use Case (HL-UC): use cases HLU1 and HLU3; each one of them defines specific components utilization in several low-level (LL) use cases.

In use case HL1-LL6 "*Big Data Clustering at Multiple Scale*" component is used to find subsets of prosumers that might participate in a DR event (see Figure 2). In this case an aggregator can receive a flexibility request from a DSO and he can leverage on Big Data Clustering tool to identify the prosumers in its portfolio that would match with such request. The petition is sent via "*DSS & DR Strategies Optimization*" module and result of the clusterization is received by the same component. In this case the flexibility request is intended as a petition for changing energy profile.

Figure 2: Interaction between "Big Data clustering at Multiple Scale" and "DSS & DR Strategies Optimization" as designed in use case HL1-LL6

In use case HL3-LL1 Figure 3 an aggregator is supposed to cluster his customers in order to assign them for different ancillary and balance markets. The DSS & DR Strategies Optimization module sends a request to Big Data Clustering module to provide the clusters that may participate in the different energy markets. Thus, the Big Data Clustering module is sending a request to the Load Profiling module, that send back the profiles of the customers as defined by the aggregator. With this input the Big Data Clustering module is able to give the requested data to DSS&DR strategies optimization module.

[1] During the writing and editing phase of this document, progress in Task 2.4 in the same time, have arisen the possibility of minor changes to the platform architecture. In this case it seems that field devices could likely provide their data through a dedicated database in the decentralized repository. This change, if confirmed, will not affect the logic of the interconnection among modules or their operation. Similar variations could occur in the re-definition of the use cases during T2.2 and T2.4 progresses.

Figure 3: Interaction of "Big Data Clustering at Multiple Scale" component as described in use case HL3-LL1

Finally, Big Data Clustering module is also described in use case HL3-LL5/6/7 (Figure 4), where an aggregator needs to estimate 30 minutes generation and load forecasts to select suitable clusters of loads to be shed and set points of dispatchable generators. In this case the VPP Generation and Modelling sends its data to the Big Data Clustering at Multiple Scale to perform such analysis and identify the requested clusters. Results are sent back to the same VPP Generation & Modelling module.

Figure 4: Interaction between "Big Data Clustering at Multiple Scale" and VPP Generation and Modelling" components as described in use case HL3-LL5,6 and 7.

The operation of the Load Profiling component is described in use case HL3-LL1: "*Prosumers profiling and Clusterization*" (Figure 5). In this use case the aggregator receives data from Field Devices (likely to be connected through a database in the decentralized repository according to forthcoming version of the architecture) and calculates profiles of its customers updating these profiles with forecasted and real time data and clustering them in order to categorize their participation in ancillary and balance markets.

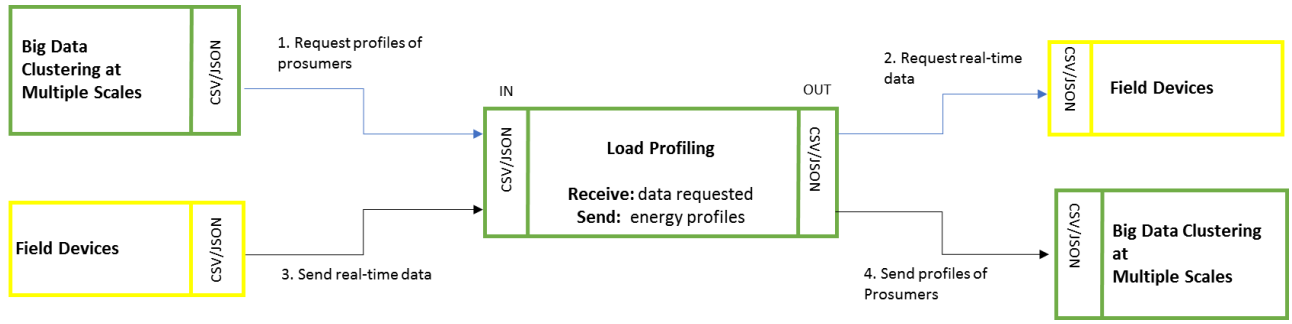


Figure 5: Interaction of Load Profiling Module as described in use case HL3-LL1

2. Big Data Layer

The Big Data Layer is composed by the following components:

- Big Data Tool Engine: A set of orchestrated functionalities to provide data lifecycle management for energy domain advanced services
- Load Profiling: A non-intrusive appliance load analysis technique with the goal of obtaining time series of the actual behaviour of customers
- Big Data Clustering at Multiple Scale: Analytical component for clusterization of energy customers (core component embedded in Big Data Analytics Engine tool)
- Customer Segmentation: Big data tool for clusterization of energy customers

Figure 6 represents the conceptual organization of Big Data Layer with its corresponding inputs and outputs described below.

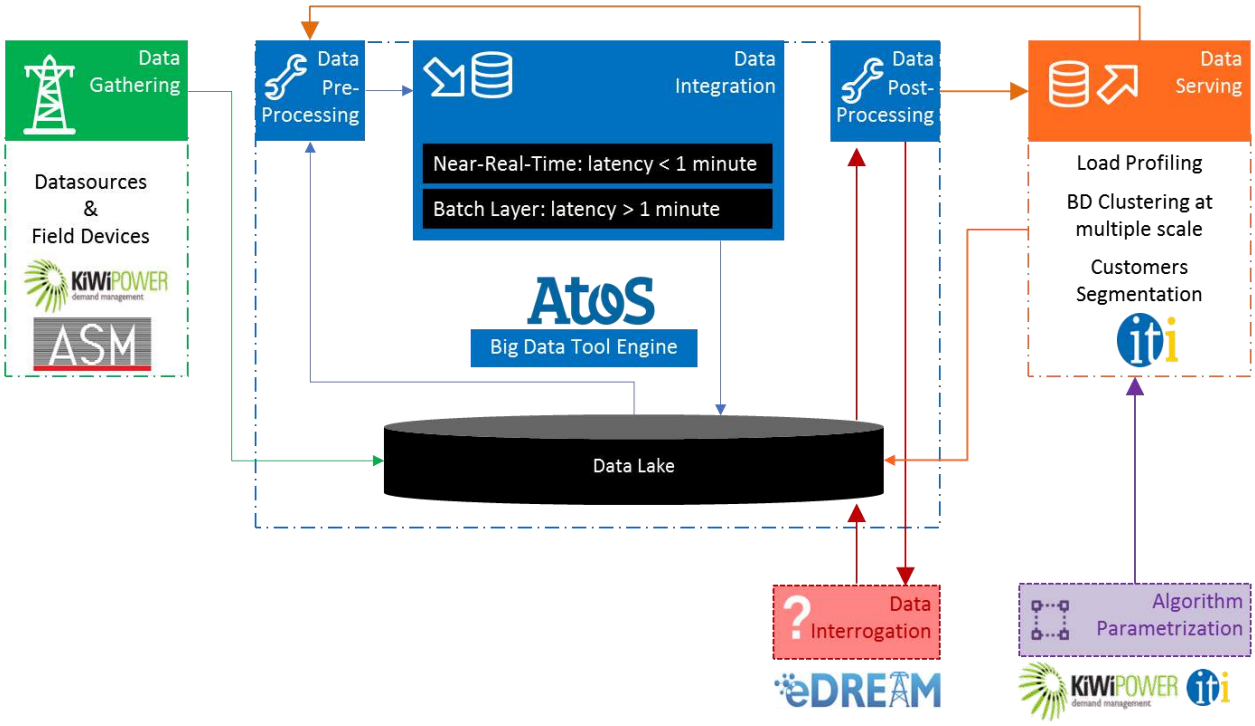


Figure 6: Conceptual Organization of the Big Data Layer

Firstly, KIWIPOWER and ASM provide the data from data sources and field devices which will be stored in the Data Lake. Data can be received from field devices, but also from other components in bottom layer of the core modular platform like “Electricity Consumption/Generation Forecast”, “VPP Generation Modelling & Forecasting” and “Multibuilding DR characterization through thermal optical and lidar information fusion”. This module, included in Big Data Tool Engine module, provides the inputs data to pre-processing elements, where data is transformed in a correct format to be easily and effectively processed. Data Integration module filters: latency in near real-time data for less than a minute and batch layer for latency higher than a minute. These latest data will be stored in Data Lake again. Last module, that integrates Big Data Tool Engine module, is Data Post–Processing which subdivides the dataset in order to send them to the different components of data serving module. Load Profiling, BD Clustering at Multiple Scales and Customer Segmentation (orange module), corresponding to data serving, receive

data from data post-processing component and KIWIPOWER algorithm parametrization tool. Once data is processed, results are collected in data lake.

2.1. Software pre-requirements

The Big Data Layer must accomplish all the requirements defined in the use cases, thus several components shall be developed and deployed.

To deploy a fully functional big data architecture as well as all the necessary components, it shall be useful to take advantage of the following technologies:

Python (Python, 2019)***:*** It shall be the reference programming language for the whole layer for its productivity, fast and easy prototyping of their models and the almost endless list of packages that can be added to extend the native functionalities of the language.

For the actual big data layer some of the python packages mentioned before might result useful. Below are described some of the most important ones:

- ***Numpy*** (Numpy, 2019)***:*** It is a package aimed to scientific computing and it is also convenient for array operations.
- ***Pandas*** (Pandas, 2019)It is the "Python Data Analysis Library". Pandas helps developers to work with datasets, it allows to read and write those datasets in several formats: csv, h5, parquet, etc. Pandas can also be suitable for performing reshaping tasks and other transformations over the datasets.
- ***Dask*** (DASK, 2018)***:***It can parallelize many of the tasks that pandas or numpy perform. Thus, it can help to optimize and scale algorithms.
- ***SciPy*** (SciPy.org, 2019): "Is a free and open-source Python library […] SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering". (Wiki SciPy, 2019)

Both Numpy and Pandas, as well as Dask since it is very similar to them, work well together and shall help throughout value chain, from data pre-processing to results visualization stages.

On the machine learning and deep learning field there are some packages that are strongly recommended. These packages are Scikit-learn, Tensorflow and Keras.

- ***Scikit-learn*** (scikit-learn, 2019)***:*** It provides some prebuilt machine learning and data analysis algorithms. It is built on Numpy and some other packages.
- ***Tensorflow*** (TensorFLow, 2019)***:*** It is the most important framework for developing neural networks on python. It is developed and maintained by Google. Currently, it has become one of the most adopted machine learning framework by the community (Hale, 2018).
- ***Keras*** (Keras, 2019)***:*** As Tensorflow is the framework, Keras is the high-level API. It supports Tensorflow, Theano and CNTK. It can run on CPU or GPU, and it can scale up to hundreds of machines.
- ***Statsmodels*** (StatsModels, 2017)***:*** It is a Python package, released under open source Modified BSD (3-clause) license, that allows to explore statistical data, estimate many different statistical models, as well as to conduct statistical tests.

- ***XGBoost*** (XGBoost, 2016)***:*** It is an open-source optimized distributed gradient boosting software library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework and provides Scalable, Portable and Distributed Gradient Boosting (GBM, GBRT, GBDT) library able to solve many data science problems in a fast and accurate way, supporting the major distributed environments such as Apache Hadoop, Apache Spark, and Apache Flink.
- ***pyAstronomy*** (PyAstronomy, 2019): it provides a collection of packages, which fulfil a certain standard both in code and documentation quality. These packages give an answer to a several statistical problems allowing to explore and model statistical data.
- **Tsfresh** (tsfresh, 2019): is a python package capable of calculating a large number of time series characteristics, features, automatically. It contains methods to evaluate the explaining power and importance of such characteristics for regression or classifications tasks.
- **Sarima** (Brownlee, 2018)**:** is a Seasonal autoregressive Integrated Moving Average forecasting method, an extension to ARIMA, that supports univariates time series data with a seasonal component.

2.2. Technological view

Given the large availability of tools and libraries, python has become the most chosen option in the data analytics, big data and machine learning knowledge fields like. After the identification of the technologies that shall be used, a brief description of the architecture of the big data layer and its components is provided hereby.

First of all, the platform shall manage two different types of data: historical data and real-time/nearly real-time data. On the one hand, historical data should be provided as csv files. Attached to these files must be provided a detailed explanation of the data model of the dataset.

Alternatively, real-time data (or nearly real-time data) generated by field devices should be directly ingested to the big data layer. The workflow shall be similar to the following one:

- Use FIWARE IoT-Agents (Telefonica Investigación y Desarrollo, 2019) philosophy to connect field devices with a context broker.
- FIWARE Orion (Orion, 2019) shall be the context broker which receives the real-time data from the IoT-Agents.
- FIWARE Cygnus (Cygnus, 2019) shall take data from Orion and store it in a database making new historical datasets.

Cassandra NoSQL (Cassandra, 2019) database is likely the best option for real-time request, been able to offers scaling with minimal administration and high reliability. It offers flexible wide-column and it is a good choice for unstructured data which expect rapid growth of the database.

Although Cassandra seems to be the best option, MongoDB or Hadoop could be other good options. MongoDB also offers good real-time requests, been able to provide responses in milliseconds, but data storage is stored in documents. If necessary, it can easily interact with Hadoop (https://hadoop.apache.org/) to perform batch aggregation tasks, big data storage and big data analytics.

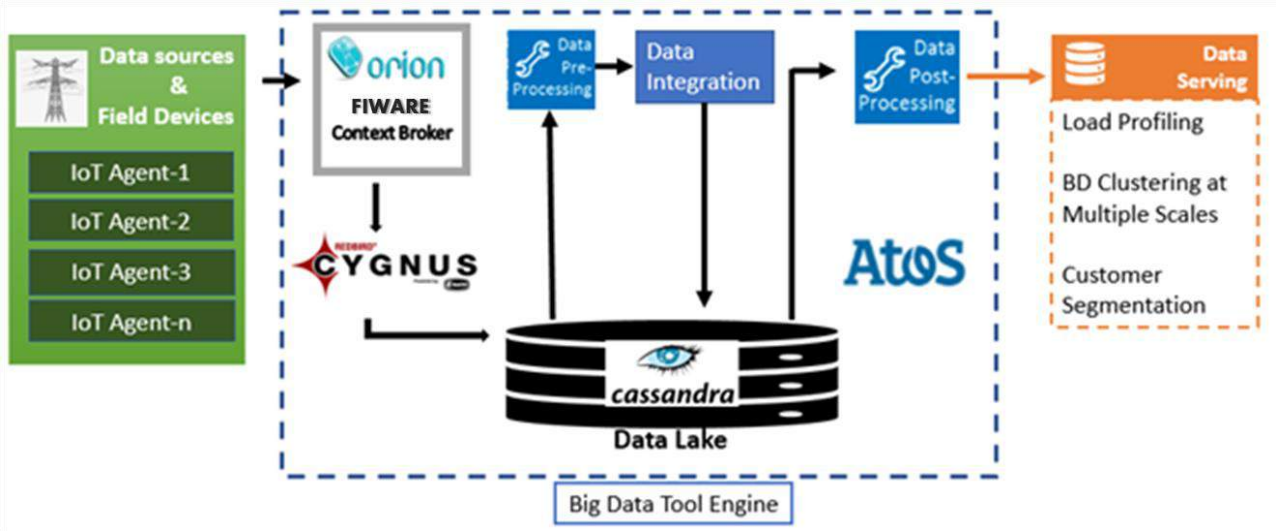


Figure 7: Schematic View of the Big Data Platform

Figure 7 shows the workflow with Fiware (Fiware, 2019) and database described previously. IoT-Agents included in data sources and fields devices module send real-time/nearly real-time data to FIWARE Orion. Orion context broker is a C++ implementation that allows you to manage the entire lifecycle of context information including updates, queries, registrations and subscriptions. It provides the Fiware NGSIv2 API which is a simple yet powerful Restful API that allows to create/update attributes (and metadata) whose values use JSON native types. FIWARE Cygnus takes data from Orion and store it in Cassandra`s database. Cygnus is a connector in charge of persisting certain sources of data, creating a historical view of such data; internally, it is based on Apache Flume, a technology addressing the design and execution of data collection and persistence agents.

2.3. Stability and Scalability

This section deals with the definition of the main design features for ensuring scalability and stability to the so-called big data layer in the eDREAM core modular platform. According to the description of the task and deliverable, for a better approach to stability and scalability problem in big data some fundamentals requirements should be considered when designing the big data platform.

Basic references about big data are evolved from 6 Cs (not univocally defined and lacking a well-structured categorization (Lee Jay, 2014)) to 5 Vs a little more defined with a general acceptance even by the European ICT community. So, the main five characteristic the platform shall accomplish are: Volume, Velocity, Variety, Veracity and Value (Gandomi & Haider, 2015), (Ishwarappa & Anuradha, 2015), (Jain, 2016) and (Marr, 2015).

Volume denotes to the huge quantity of data that could be generated or processed every second. Just think of all the smart meters and other sensors installed at pilot sites that can generate a set of data with sampling frequency in minute or seconds range. In addition to traditional equipment, eDREAM is also considering the possibility to acquire and process pictures and videos from aerial survey through a set of devices (lidar, thermal cameras, optical cameras) that would need a very hard processing (pre and post) and computing tasks in order to extract information from signals. On the other hand, blockchain technology due to its inner peculiarity is a very big data demanding application too (Karafiloski A., 2017).

Velocity refers to "the speed at which new data is generated and the speed at which data moves around". Velocity is the other factor, together with Volume responsible for the increasing number of data available and the time interval these data are generated, processed or stored (Ishwarappa & Anuradha, 2015).

Variety refers to the heterogeneity of data. Differently from the past, where were the challenge was to categorize data in a structured way within relational database, today we are able to store and work with uncategorized data (or unstructured data). Of course, this is a main characteristic to be accomplished in eDREAM modular platform, since we are going to exchange data with a different sources and platforms that were not originally designed to work together.

Veracity refers to the unreliability or trustworthiness of the data. This is a very critical feature because it affects the quality of data and, therefore, the quality of the information that could be extracted from data. eDREAM big data layer will be designed in order to accomplish with this characteristic, due to the high number sources, services and user categories that will interact with the platform.

Value refers to the capacity to convert the information extracted from data in profitable and valuable tools. It basically represents the junction point among the aforementioned features and the business leveraging on data. Due to its typical relationship with the more end point of the market process it is frequently depicted in a different category with respect of the other ones. Figure 8 represents a schematic concept of the five Vs and through that, it is possible to appreciate the comparison among features. (Marr, 2015) states without any doubt that "the one V of big data that matters the most" is Value because all the previous ones must be converted into it. At the same time the Big Data Value Association, in the reference document cited in footnote 2specifically highlights and summarizes the impact and the business support Value would bring in four bullets, that fits very well with eDREAM goals:

- "Improving Efficiency";
- "Creating Transparency";
- "Discovering Users' Needs";
- "Better product/service customization".

Figure 8: Conceptual interaction of the 5 big data features (source BDVA2)

[2] http://www.bdva.eu/sites/default/files/brochure_bdv_150327_0.pdf

In accordance to the 5 main features to keep in mind when designing the eDREAM big data layer, *stability* and *scalability* must be ensured for the proper execution and performance of the services. They are both mandatory in the eDREAM designing and development phase, because of the replicability and validation of the solution in larger scale pilots. Therefore, it is important to guarantee both concepts by design in order to give a wider validation range to the proposed solutions. At the moment it is foreseen a potential application on few terabyte datasets (limited to the size of the involved pilots), but of course the layer will be able to handle larger sizes.

The *scalability* is the first main achievement to ensure and it can be defined as the capacity to handle an increase in the amount of data, number of requests, number of users, etc. without this implying a loss of performance. Of course, it is tightly bounded with the volume and velocity of the ingested data because these are the main features that would require the stability as direct solution. The volume of the data may increase by adding different data sources that would likely be part of the pilot site: if we consider that virtual power plant will be the reference pilot model to consider, then it is easy to imagine that the number of data sources it is expected to increase in order to allow the VPP to participate in several market options. Moreover, the benefit provided by the services -DR and VPP modelling based on the big data- and the access to the services themselves must be ensured regardless the number of users of the platform (Amudhavel J., 2015). The problem basically resides in ensuring how the proposed solutions can be able to scale up to handle much heavier loads even though the data access might be limited in the foreseeable future.

In a general meaning it is possible to identify common bottlenecks or issues in when approaching scalability problem. There are two different scalability approaches: horizontal and vertical scalability. Vertical scalability means to increase the number of resources in a node. Meanwhile, horizontal scalability is to increase the number of nodes. It is a very well-known problem and consolidated solutions have been today generally accepted and spread out (Michael M., 2007). The first approach (basically adding computational resources to the system) is not sustainable in our case, thus for obvious reasons including system efficiency and the increasing of costs, the second one fits better with the goal of the project. This horizontal scalability allows to add more servers in the case they will be necessary based on partitioning the data; which means adding more machines. One good example of horizontal scalability is Cassandra that can add more nodes to the cluster when it is required, and it will utilize the new resources automatically without reconfiguration (Pfeil, 2010).

Considering eDREAM goals and purposes, the big data layer shall be based on a microservice architecture. A microservice is an isolated and replicable module that can play a specific role in the architecture, this type of architecture allows to easily scale out the layer by only replicating the services needed and allowing to easily parallelize the workload. Among a large variety of tools and technologies a couple of modules have been specifically selected:

- Tensorflow (TensorFLow, 2019): it is a machine learning framework that is able to design and develop Deep Neural Networks. It also can train the models in a distributed way, up to hundreds of GPUs or even TPU (Tensor Processing Unit);
- Hadoop (Hadoop, 2019): it is a complete ecosystem aimed to store vast amounts of data across multiple number of servers. This system fixes well with the idea of microservices and horizontal scale. Hadoop replicates the stored information (3 times by default) to ensure high stability and availability

The *stability* is the second main feature characterizing the big data solution of eDREAM. Basically, it is the capacity of a system to work as it was designed for, no matter the internal or external conditions, excessive workload, unexpected data, external attacks, etc. According to the standard (ISO/IEC, 2001) the stability is defined as the "Capability to avoid unexpected effects from modifications during deployment time or the normal operation of

the system". Thus, it is possible to stand out deployment stability from the operational stability, but the general meaning is still the same. The importance to achieve stability requirement for the big data layer is crucial for the project's purposes; if a service or the platform is not stable it may fall into several problems, loss of important data, become unreachable for users, etc. Reaching stability can be a design task as well an implementation challenge. On one hand every piece of software developed shall pass several testing phases (unit tests, functional tests, integration tests…) aiming to ensure the stability of the system. Additionally, the microservice architecture can replicate some critical services to work in a backup mode (databases, API services…) and avoid losing data. Testing procedure will assume a crucial part for ensuring the system's stability; the following test categories shall be performed:

- unit tests;
- integration tests;
- functional tests;
- Other tests (if needed like linter tests).

All these tests shall be executed automatically by gitlab every time a change in code is pushed to it. Further details about this procedure are described in deliverable D6.1 (Integration and Interconnection Plan).

3. Pre-Processing

Data pre-processing is the first, and often mandatory, step in the processing pipeline of data acquired within the framework of real use cases. This is due to several factors, such as different sources of noise (e.g. thermal noise, quantization noise, etc.), communication issues with the field devices, and so on. Also, further pre-processing may be needed to transform data, as machine learning techniques often make strong assumptions regarding parameters such as data distribution or scale. As an example, k-means clustering algorithm assume data are distributed according to a Gaussian distribution, while recurrent neural networks with LSTM gates require data to be scaled within the range of [-1, 1].

The pre-processing pipeline is different according to the type of data to pre-process. Specifically, two types of data can be distinguished: the first are identically and independently distributed (IID) data, while the second are time series. The main difference between IID data and time series lies in their temporal correlation. On one hand, IID data are not temporally correlated: that is, given two data samples y_1 and y_2, they are both independently representative of the underlying data generation mechanism. A typical example of IID data are prosumers: each one can be considered an independent instance of a more complex generation mechanism. On the other hand, time series are temporally correlated, meaning that the specific value of the time series at time t, that is, y_t, can be predicted by knowing the values of the time series at previous time instants, that is:

$$y_t = f(y_{t-1}, y_{t-2}, \dots, y_{t-n})$$

Obviously, the previous relationship is simplified, and more refined models should consider also other parameters (such as exogenous variables, noise, etc.).

3.1. Data Cleaning

As it was previously mentioned, gathered data is required to be pre-processing in order to achieve good result in the further analysis. This pre-processing avoids the most frequently poor gathered data quality:

- **Incomplete data:** it appears when an attribute does not have all its values, for example there is the header but not the details. This problem can be solved in different ways, for example:
    - Ignore the column if it is not relevant in the further analysis
    - Eliminate or replace the entire column or row, with the risk of obtaining scanty data
    - Replace the value by the mean, variance or mean or predict the value.
- **Inconsistent data:** in this case data don´t match (e.g. two gathered consumers` data have same values with different names).
- **Not enough amount of data**: the amount of data is not enough for the analysis of all the classes that want to perform.
- **Noisy data:** are the errors in the data (e.g. negative nominal power values)

In this context, with the objective of avoiding this unreliable data and achieve good result in the further analysis, it is important to prepare raw data to meet the requirements of data mining algorithms. Data cleaning is an assembly of data mining techniques that enable the data better to work with, standardize, remove nulls, duplicated or invalid data:

- **Standardization data:** unify all gathered data structures;

- **Remove duplicated data:** the duplication of values in any database leads to an analysis by the wrong algorithm, applying more weight
- **Remove nulls:** sometimes in gathered data, it may be the case where one column will have zeros values. If this situation happens that column will be removed, no values add value to the analysis;
- **Invalid data**: connection problems can trigger missing values in datasets. Two ways in order to solve this condition are: replacing the missing values with its mean, median or mode, or predict the missing values with algorithm, as linear regression, in the case it does not have high variance.

All these techniques are applied with SciPy, a library implemented in Python.

3.2. Determining the data generation mechanism

The first, mandatory step to data processing is to evaluate the type of *data generation mechanism* under analysis.

Usually, as already mentioned in this chapter, data may belong to two types of data generation process shown below:

- **Time Series**: where each sample has a temporal dependency with previous samples;
- **Identically and independently distributed data**: where each individual sample is considered to be independent from the others.

Correctly determining this aspect is crucial to properly select the tools which will be used in next processing steps.

In eDREAM case, data are treated as time series. That means that each prosumer can be associated to an energy outcome model, which accounts for conditioning parameters such as geographic location, type of power plant, and many more.

3.3. Exploratory Data Analysis and Pre-processing

Once the data generation mechanism has been determined, an *exploratory data analysis* should be performed. This approach differs from the classic *confirmatory* data analysis, as there are no previous assumptions on a model to which the data generation process must adhere.

Exploratory data analysis may be performed through a variety of tools, ranging from visual to analytical tools.

From previous considerations, different strategies should be used to deal with different types of data. Let us then define a generic pre-processing pipeline for such data, which can be refined and adapted to the specific use case.

Usually, time series acquired within real use cases show the absence of a set of data, due to various issues (such as lack of communication between devices, noisy data, etc.). However, statistical and machine learning methods used for time series modelling assume the time series to be completely available, therefore a procedure to fill missing data is needed.

To this end, several methods are available, according to the specific needs of the use case. A first, naive (yet effective) approach would be the use of a moving average filter to fill missing data. Specifically, the approximation of the value of y_t of the time series Y at time t can be defined as:

$$\hat{y}_t = \frac{1}{t}\sum_i \alpha_i \cdot y_{t-i}, i = 0, \dots, t-1$$

Therefore, the prediction of the value $\hat{y}_{(t+1)}$ is given by a weighted average of the previous values assumed by the time series. Obviously, one should consider only a subset of previous value, by using a moving window of a predetermined size.

Another approach implies the use of interpolation methods, which are inherently most expensive as for the computational cost but can be more effective. A well-known interpolation method is cubic spline interpolation, which defines an interpolating cubic polynomial which guarantees for a smaller interpolation error with respect to other types of polynomial.

A most sophisticated method to fill missing data implies the use of statistical modelling tools, such as Seasonal Auto Regressive Integrated Moving Average (SARIMA) (Box, 2015) models, or machine learning modelling tools, such as recurrent neural networks. With this approach, which can be used when large amounts of data are available, a base model Y_K for all the (fully available) time series *Y(K)* which represent a certain data generation mechanism *K* can be computed. Then, let us suppose that the *j*-th series Y_{K_j} shows the absence of data at a certain time *t*; hence, the value y_{K_j} at time *t* can be filled using the value at the same time of the base model Y_K.

The approach is different for the case of IID data, for which the pre-processing pipeline is based on different assumption with respect to the pre-processing pipeline for time series, due to their independent nature.

Specifically, these data need to be subject to a transformation; however, the imputation of missing values is not needed, as incomplete data can just be discarded, since they are not relevant for the overall analysis of the underlying process. Furthermore, especially when a high number of features is given per each data sample, either a dimensionality reduction or a feature selection may be needed.

- **Dimensionality reduction** consider a reduced number of dimensions with respect to the ones embedded within the initial dataset. Specifically, well-known techniques such as Principal Component Analysis (PCA) allows to retain the most relevant components of the data by means of a series of consequent orthogonal projections. Thus, while selecting the principal components which mostly explain the original variance of data, one can apply algorithms (such as k-means clustering) which fail in high-dimensional spaces;
- **Feature selection** also reduces the number of dimensions of the original dataset; however, while dimensionality reduction modifies the hyper-space of data itself (due to the transformations applied to extract principal components), feature selection retains the same space, removing the least relevant features according to parameters such as variance or relevance. As an example, Recursive Feature Elimination (RFE) exploits an estimator (i.e. a classifier, such as a Support Vector Machine) to select features recursively, by considering increasingly smaller sets of features in the analysis.

3.3.1. ASM Terni Dataset exploratory Analysis and Preprocessing

The ASM Terni dataset gathers the samples from 137 different prosumers collected along the whole 2017. Data are sampled every 15 minutes, and holds the following fields:

- `$E_a$`: active energy absorbed by the prosumer (absorbed from the grid);
- `$E_o$`: total active energy delivered from the prosumer to the grid (injected into the grid).
- `$E_{ra}$`: absorbed inductive reactive energy;
- `$E_{ri}$`: injected inductive reactive energy.

The dataset also holds some cumulative values per-user, that is:

- `$P_{count}$`, nominal supplying contractual power (in kW);
- `$P_{imm}$`, nominal power (in kW) delivered by the prosumer (nominal power of the generation unit connected to the grid);
- `$E_{imm}$`, total annual energy (in kWh) delivered by the prosumer to the grid;
- `$E_A$`, total annual active energy absorbed/consumed by the prosumer (in kWh);
- `$E_P$`, total annual active energy produced by the prosumer (in kWh).

It is interesting to evaluate the ratio τ between `$E_P$` and `$P_{imm}$` *which* represents the annual energy outcome normalized by the output power:

$$\tau = \frac{E_P}{P_{imm}}$$

The value of τ is given in hours. Dividing the total production E_P by the nominal power delivered by the prosumer P_{imm} gives an indication about the *operating time* of each prosumer plant. Just for the sake of preliminary tool development, we can hypothetically imagine a full operational capacity of a prosumer generation plant at an average of 30% of the overall daily operation, a threshold value ρ could be defined as the value under which it is believed there are malfunctions or inconsistencies. In these experiments, $\rho = 700h$ (which should be considered different for each type of user); this value is set experimentally through a dataset evaluation. It is important to underline that the dataset shows two critical issues, mainly related to missing values and inconsistent sampling. Specifically:

1. as for missing values, there are several of them within the dataset. This may be caused by several factors, such as temporary lack of communication with the smart meter, or failure in database CRUD operation;
2. as for inconsistent sampling, there are several inconsistent prosumer plants (e.g. the data may fail for some period of time leading to inconsistent sampling rates due to heterogeneous causes).

	Id Meter PROSUMER	Id Meter2	Registration number	Power Absorbed (kW)	Power Delivered (kW)	Energy Delivered (kWh) 2017	Energy Absorbed (kWh) 2017	Total Prod. 2017	Type of Use	Model	Model 2	Energy Source
0	509620649073	620649073	620649073	3.30	6.00	5990	3919	7828	Domestico	311-ELET CONS&PROD F ATT S	311-ELET CONS&PROD F ATT S	Energia solare
1	509620681123	620681123	620681123	3.30	3.00	3508	749	4547	Domestico	111-ELETTRONICO FASCE ATT	111-ELETTRONICO FASCE ATT	Energia solare
2	512670014346	670014346	670014346	33.00	57.33	1190	22430	2710	Altri Usi	115-ELET F A/R/P	314-ELET CONS&PROD F ATT REA S	Energia solare
3	511620935490	620935490	620935490	4.95	3.00	2247	4958	3701	Domestico	111-ELETTRONICO FASCE ATT	311-ELET CONS&PROD F ATT S	Energia solare
4	509620681813	620681813	620681813	3.30	3.48	3230	1731	5204	Domestico	101-ELETTRONICO FU ATT	311-ELET CONS&PROD F ATT S	Energia solare

Figure 9: ASM Terni dataset Prosumers Nominal Data

Regarding the first case, it would be necessary an evaluation on a per-case basis whether a proper filling procedure should be used, or if data should be discarded. This procedure can be automatized using a set of decision criteria, such as the availability and the size of the dataset. While in the second case, the fixed threshold indicates that the

daily result of each prosumer should be, on average, almost twice its nominal output power. In order to address these issues, the following steps are proposed:

1. Removing the inconsistent prosumers (prosumer for which `$\tau_p < \rho$`);
2. Performing data-filling procedure on remaining data, or removing the prosumer completely;

In particular, for the second step two different procedures are proposed as follows:

- **Daily-based cubic spline interpolation**: Let us consider a prosumer `$\boldsymbol{p} \in \boldsymbol{P}$`, where `$\boldsymbol{P}$` is the set of prosumers. The ASM Terni dataset holds values which can be interpreted as a three-dimensional tensor, where the X-direction and the Y-direction represent the sampling relative to a single day, while the Z direction is given by the day of measurements. As missing values are normally found on a daily basis, the day-based cubic spline interpolation is a classic cubic spline interpolation performed considering all daily measures at a given time, and then accordingly interpolating missing values.

  More formally, given that $\boldsymbol{E_{imm\text{-}tdp}}$ is the value of $\boldsymbol{E_{imm}}$ at a given time `$\boldsymbol{t}$` and a given day `$\boldsymbol{d}$` for the prosumer `$\boldsymbol{p}$`, let us consider the vector $V_{Eimm\text{-}tdp}(d)$ which holds all the values of $\boldsymbol{E_{imm}}$ at a given time $\boldsymbol{t}$ for the prosumer $\boldsymbol{p}$. Then, a cubic spline interpolation procedure is performed on the vector $rV_{Eimm\text{-}tdp}(d)'$.

- **Prosumer context-based data retrieval:** The daily-based interpolation is simple to perform, and, as it is based on data acquired for a certain prosumer `$p$`, retrieved data are representative of the data generation process relative to prosumer `$p$`.

However, there may be some scenarios where this approach is not robust, as more context information, other than the data generation process, may be needed. Specifically, conditioning parameters such as daily weather condition can influence the daily (and even hourly) energetic outcome and, therefore, they should be considered in the analysis.

Therefore, the *prosumer context-based data retrieval* considers a set of context-based information, such as the prosumers typology, or its geographic location. Prosumers are therefore labelled beforehand, and an *average energetic profile* is computed for each class of users. The average energetic profile is used to fill missing data values.

This procedure can be further refined if the average energetic profile is modelled using specific tools as in Figure 10 and Figure 11 (i.e. ARIMA models for time series (George E. P. Box, 2015)).

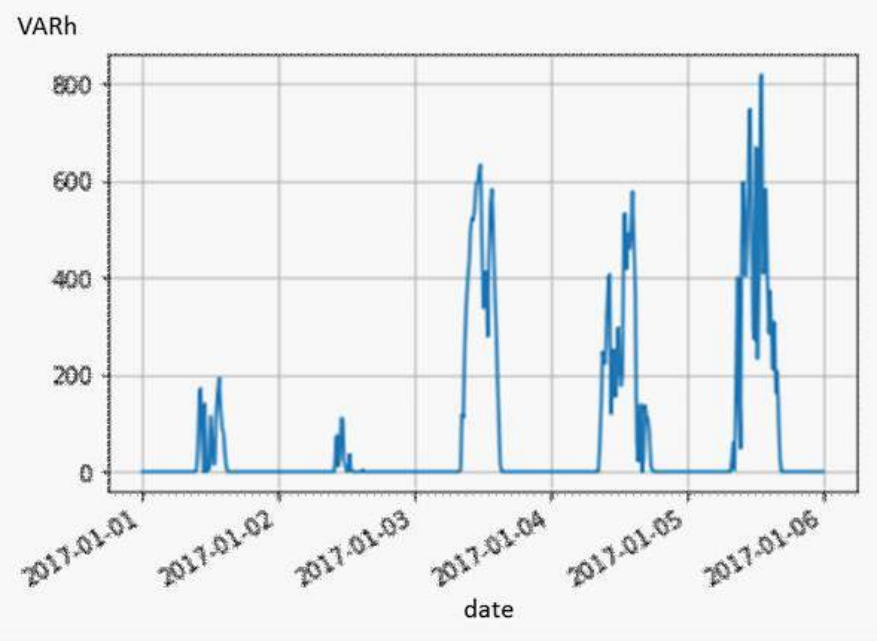


Figure 10: Data interpolated according to the prosumer context-based data retrieval method

Date	E_a	E_{ra}	E_o	E_{ri}
2017-01-01 00:00:00	199.0	23.0	0.0	0.0
2017-01-01 00:15:00	232.0	25.0	0.0	0.0
2017-01-01 00:30:00	249.0	34.0	0.0	0.0
2017-01-01 00:45:00	225.0	31.0	0.0	0.0
2017-01-01 01:00:00	212.0	0.0	0.0	0.0
2017-01-01 01:15:00	229.0	0.0	0.0	0.0
2017-01-01 01:30:00	220.0	0.0	0.0	0.0
2017-01-01 01:45:00	230.0	9.0	0.0	0.0
2017-01-01 02:00:00	225.0	0.0	0.0	0.0
2017-01-01 02:15:00	219.0	0.0	0.0	0.0
2017-01-01 02:30:00	215.0	0.0	0.0	0.0
2017-01-01 02:45:00	226.0	9.0	0.0	0.0
2017-01-01 03:00:00	223.0	1.0	0.0	0.0

Figure 11: Portion of ASM Terni dataset after the pre-processing procedures

Other than data retrieval, a possible solution can be to use, in the clustering/classification step, algorithms robust to missing values, such as **XGBoost**, which works with sparse feature matrices.

3.4. Data Normalization

Often, a set of time series do not hold normalized values. This may not be an issue if an univariate time series modelling is performed, as forecasted values should directly come from previous values (and, therefore, should be comparable with them); however, when different time series are compared, as an example in time series clustering, or specific tools are used, a normalization procedure may be a mandatory step which, obviously, envisage for a previous data-filling procedure.

To this end, several techniques are available. A first approach would be scaling the values of the time series between two different ranges, such as [-1, 1]. Another approach would be applying a standardization procedure to the values of the time series, to make their distribution resample a Gaussian one. However, these approaches may be sensitive to outliers; hence, more robust methods may be applied, which consider non-parametric transformation to map data to uniform distribution within a certain range.

It should be noted that, once data have been processed and analysed, an inverse transformation should be applied to normalized data; otherwise, the overall outcome of the processing would be biased by the normalization procedure itself.

As regards the ASM Terni dataset, the normalization procedure should be performed on the whole dataset, and not on a per-used basis, so, the normalization has been done accordingly. This is obviously due to the need to preserve information on both the maximum and minimum value of power and/or energy acquired for each prosumer.

Furthermore, raw data are biased by a conversion factor `$C_p$`, related to each prosumer: `$p$`, and given by:

$$C_p = \frac{E_{imm}}{\sum_p E_{o_{imm}}}$$

That is, the conversion factor given by the ratio of the annual, cumulative energetic outcome and the sum of the energetic outcome computed in each sample. This is due to conversion issues specific to this dataset and, as a consequence, must be considered in the pre-processing.

3.5. Context Filtering

Data can be labelled according to context information, that is, metadata available within the dataset.

For the ASM Terni dataset, context information is given by the prosumer type, which can be “domestic” or “other uses”. As for the geographic location, which is important to assess factors such as weather condition, are not explicitly given. However, as the prosumers are located in a small area, this information is not considered to be relevant, and the (simplistic) assumption that the geographic location is the same for all the prosumers could be done. Moreover, as an unsupervised approach for data clustering has been applied to the preliminary data set, at this stage, it was not considered necessary to perform the labelling.

3.6. Outlier Detection

Outlier (anomaly) detection aims to identify data which do not belong to the data generation mechanism under analysis but, they are still available within the dataset. These data can be representative of anomalous situation and can be treated accordingly (e.g. by raising an alert in a real-time monitoring system). Let us start by classifying outliers according to their type.

There are different types of Outlier that can be detected, such as:

- **Global outliers** are data samples which are considered to be globally outside their dataset. An example of a global outlier is a relevant, single spike found within a time series, or a prosumer with a noticeable difference in her global annual power consumption;
- **Contextual outliers** are a subset of data samples whose collective behavior deviates from the data set, even if their individual behavior is considered to be normal. An example of such outliers is a set of points within a time series which deviates from a seasonal pattern, or a group of prosumers whose behavior differs from the one of their kind, but it is commonly found in a different type of prosumers;
- **Collective outliers** have a behavior which significantly deviates from the rest of dataset. This type of anomaly can be also representative of novelty, that is, may be a symptom of the presence of a new data generation mechanism.

Due to the differences between IID data and time series, different outlier detection methods should be used with each type of data.

Outlier detection with IID data: Several methods are available to perform outlier detection with IID data. As an example, one may assume that regular data come from a Gaussian distribution, therefore a simple method may envisage for a variance threshold above which data are considered to be outliers. A direct application of this method is an outlier detection approach which uses z-score to determine whether a data point is an outlier. Another class of approaches are density-based approaches, such as the ones used by the DBSCAN clustering algorithm: specifically, points which are not adequately represented by a data cluster are discarded and considered as outliers. A slightly more sophisticated approach is the one used by isolation forests, which are based on the concept of tree ensembles, and computes outliers according to a score related to the specific path length (i.e. the number of splittings performed by the forest).

Outlier detection with time series: Outlier detection algorithms for time series are slightly different from the ones used for IID data. A first approach would be to use an STL decomposition (Cleveland, 1990), and then to evaluate residuals; if the value of a residual is above a certain threshold (that is, the residual is a spike), then that specific data point may be treated as a global outlier. Another approach uses a base model for time series, computed using the statistical tools already described for data preprocessing, to evaluate, through a statistical test (such as a Student's t-test), whether the base model and the possible outlier belong to the same distribution. If a machine learning tool is used, a supervised approach may be used to let the classifier discern between a 'normal' time series and an anomalous one.

The simplest method to perform anomaly detection on time series data is to use STL decomposition. STL decomposition decomposes the original time series Y in three terms, that is:

- a trend component Y_t, which identifies the overall trend of the time series;
- a seasonal component Y_s, which identifies the seasonal effects (that is, an effect which is shown with a certain period);
- a residual component r, which is given by:

$$r = Y - (Y_t + Y_s).$$

It is important to underline that the last point implies an additive STL decomposition.

Ideally, outliers should be normally distributed, to represent pure white noise; however, this is not the case in many real case scenarios. Therefore, if an outlier is detected within the residuals, it would be probably reflected in a sudden spike (either negative or positive) in the time series.

Outlier detection may be performed naively, supposing residuals are normally distributed, and therefore considering a certain threshold (e.g. a canonical 3σ) above which all the samples are considered outlier, or more sophisticated methods can be used. One of such algorithms is the generalized extreme Studentized deviate test (Rosner, 1983), a statistical test which overcomes classical tests, such as the Grubbs test, as only an upper bound for the number of outliers must be specified.

3.6.1. ASM Terni Dataset Outlier Analysis

An interesting example of the use of such approaches is shown in the Figure 12, which shows the STL decomposition of energy inserted by a prosumer available in the ASM Terni dataset during the first week of 2017. A daily period T=96 is considered, as there is a new measurement each 15 minutes.

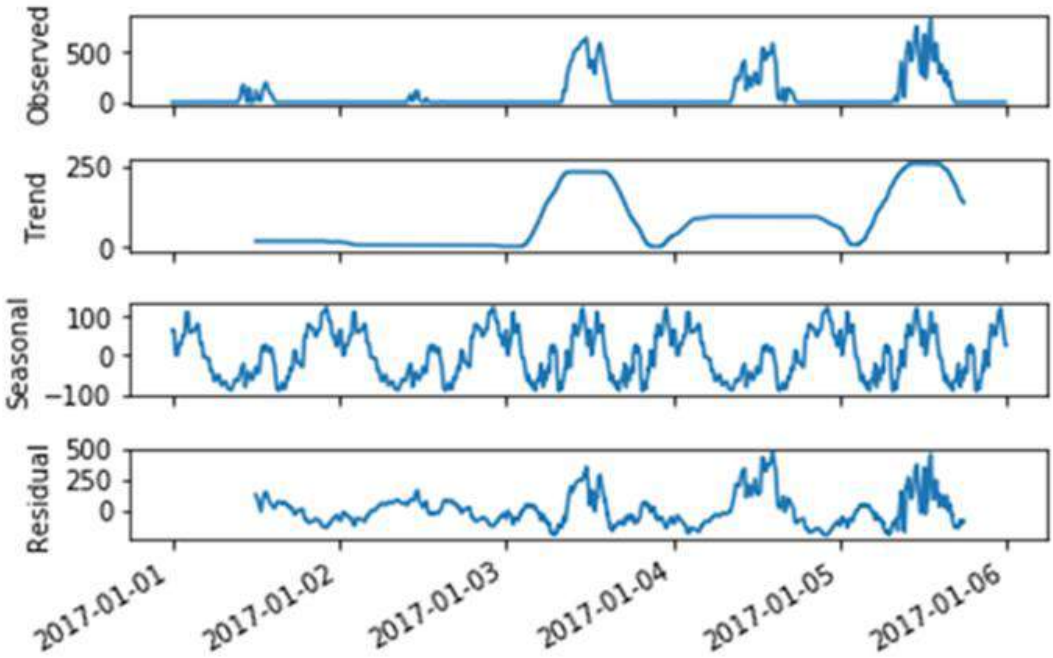


Figure 12: STL decomposition on the data acquired during the first week for a real prosumer sample (Source E@W based on ASM dataset elaboration)

It is important to underline how, in this specific case, no padding has been used; however, it should be clear that, as the trend is evaluated using a rolling mean whose window is equals to *T*, a padding procedure should be considered. As for the outlier detection, it is clear that no relevant spikes can be found within the residuals; this is also confirmed by the generalized ESD test (Rosner, 1983) , an algorithm technique has been used to make a statistical test, based on the data distribution of the overall dataset and to identify the values which are outside of a certain confidence range (outlier). Also, the most relevant effect that should be considered is a noticeable change in the trend, which can be found starting from the fourth day; this can be due to several combined effects, and more knowledge domain and/or metadata (such as the weather conditions) can be integrated in the analysis framework to achieve a more meaningful description.

Finally, it should be noted that this method relies on several assumptions that, in some cases, does not hold, such as the (approximatively) normality of the residual distribution. Furthermore, the additive hypothesis in the STL decomposition should be properly verified.

The STL decomposition tools are available in Python, specifically in the statsmodels library. As for the generalized ESD test, an implementation is available in the pyAstronomy package, which appears to have been discontinued.

3.7. Data Aggregation

Data coming from heterogeneous sources can be aggregated through proper data fusion techniques. Data fusion is part of the data preprocessing and data transformation area and allows to reduce volume data but increase its value by integrating data from a variety of sources to produce more meaningful and effective inferences and associations. Data fusion techniques could be applied to raw sensors data or to determine the analyzed data. In this context, data aggregation can be considered as a sub-component of data fusion that examines data to remove data redundancy.

Data Fusion can be classified based on a variety of attributes based on different dimensions, such as the level of abstraction or the relationship between data labels from one or more sensors (Castanedo, 2013) .

As regard the classification based on the abstraction levels, a methodology has been defined by Luo et al (R. C. Luo, 2002) that provide the following four abstraction levels: (1) signal level, (2) pixel level (for the image processing), (3) characteristic, (4) symbol.

A method for the classification on the base of the relations between the input data sources, was proposed by (Durrant-Whyte, 1988). These relations can be defined as:

a. Complementary: Information provided by different sources represent different parts of the same scenario and could be used to obtain more complete global information);

b. Redundant: two or more input sources provide information on the same target and could be merged to increase the confidence;

c. Cooperative data: Information is merged into new information that is more complex than the original information.

One of the best-known data fusion classification systems was provided by Dasarathy (Dasarathy, 1997), formalizing attributes through the following five categories:

1. **Data In-Data Out (DAI-DAO)** is the primary method of data fusion in the classification model. It processes the inputs and outputs raw data resulting in more reliable and accurate data. Data fusion at this level is conducted immediately after the data are collected from the sensors;

2. **Data In-Feature Out (DAI-FEO)** processes the raw data to extract features or characteristics that depict an entity in the environment;

3. **Feature In-Feature Out (FEI-FEO)** processes a collection of features to improve feature results. This process is also known as feature fusion, symbolic fusion, information fusion or intermediate-level fusion;

4. **Feature In-Decision out (FEI-DEO)** processes the features to acquire a series of decisions. Most classification systems that execute a decision based on sensor inputs are part of this classification category;

5. **Decision In-Decision Out (DEI-DEO)** fuses the input decisions to extract more efficient decisions.

Therefore, Data Fusion is a process of aggregation and integration of data from different sources, even heterogeneous, allowing the extraction of accurate and significant data for the domain of interest, eliminating at the same time unneeded and unusable data.

Another modality of classification used for Data fusion techniques is based on the application reference architecture (Castanedo, 2013), which can be centralized, decentralized or distributed.

The decision on which type of architecture is the best to use is based essentially on the application domain and on the specific business and technological requirements. Architectures based on decentralized and distributed logic are very similar to each other, but they differ in the way data is pre-processed.

In particular, in the decentralized architecture, data aggregation takes place at the level of each single node, while in the distributed architecture data are first pre-processed at the level of a single node and then aggregated in a centralized manner.

Therefore, the distributed architecture differs from the centralized architecture because it foresees that the pre-processing takes place at the single node level thus reducing the communication costs (sending data already pre-processed) and processing (the data are directly aggregated without further pre-processing steps).

4. Initialization and Prosumers' Profiling

The consumption data of customers has the potential to give great insights of significant importance for utilities and policy makers. Valuable insights can be derived by the knowledge of typical consumption curves of different consumer groups and understanding the main drivers of consumption. This knowledge can assist decision makers in the electricity utility industry in developing DSM (demand side management) schemes. Knowledge on the way different demographic groups consume electricity is valuable to study the effect of energy policy on different population groups. For the purpose of the EDREAM service to be developed, the process for prosumers profiling is described as in Figure 13

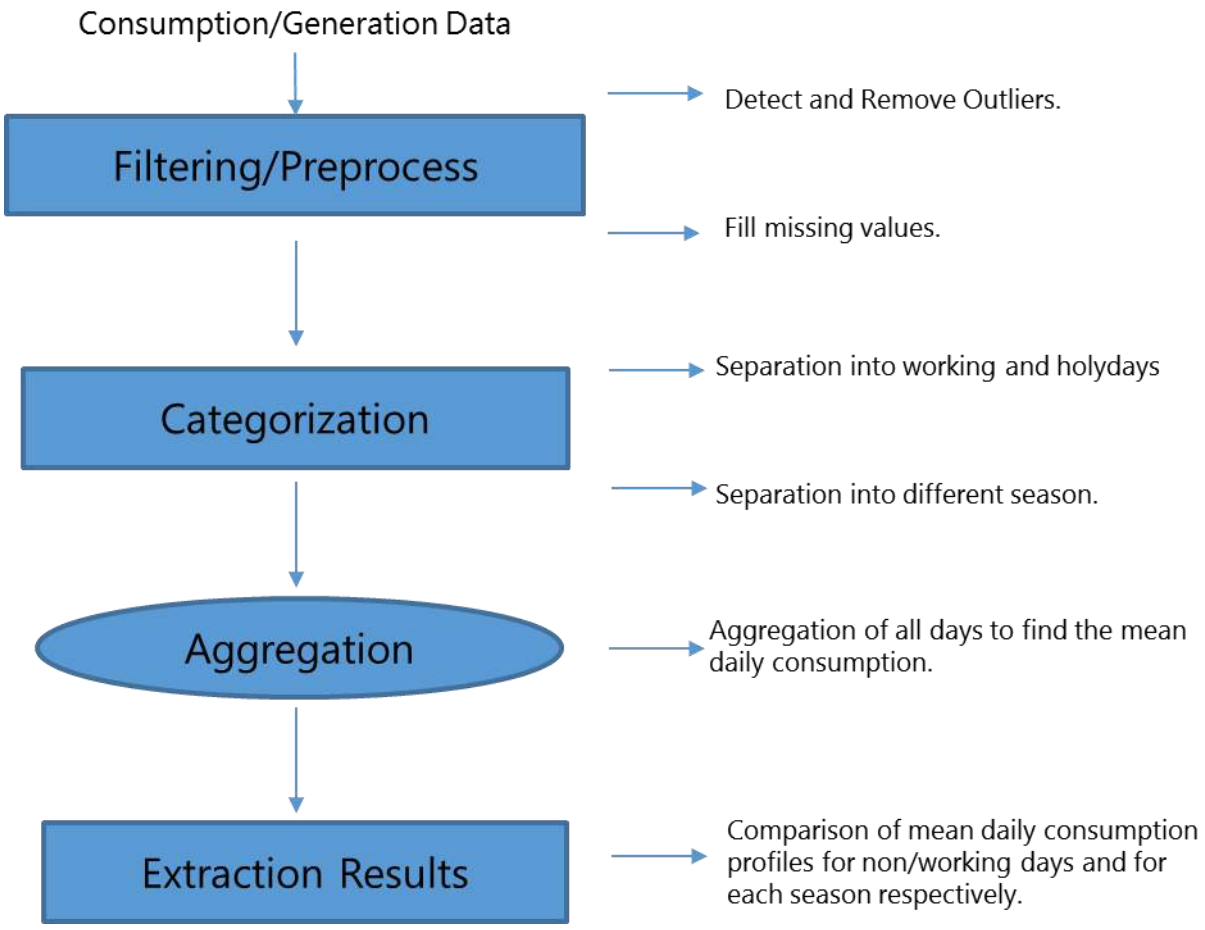


Figure 13: Conceptual scheme of the profiling process

Filtering Process. In this first step the Load Profiling module will take advantage of the pre-processing tool described in previous chapter in order to detect possible anomalies from data ingestion. These anomalies could include null, invalid or duplicated data as well a preliminary outlier analysis. During this first step the pre-processing will be able to identify empty values (e.g. measurements with no associated quantities) that could be originated for different reasons (communication failure, sensing issues, etc.). The process to "reconstruct" the missing values is described in previous chapter.

Categorization. This step deals with the organization of the electrical quantities according to specific criteria. Considering the final goal of this component, categorization is devoted to separate values according to different time/temporal intervals. First of all, a separation between data corresponding to energy consumption in different seasons should be applied. The most of procedures divide data into the four seasons, but of course according to the actual weather trends, other Some procedures does not divide Basic categorization splits working days from

holidays in order to obtain profiles that could be easily compared; in some case it is even possible to separate Saturday from holidays, since this day could be a working day for some people. A more specific refinement could be applied to holidays, by separating in different categories Sundays and other similar holydays (for instance corresponding to a higher power consumption for domestic users), from vacation days (where domestic users are at minimum).

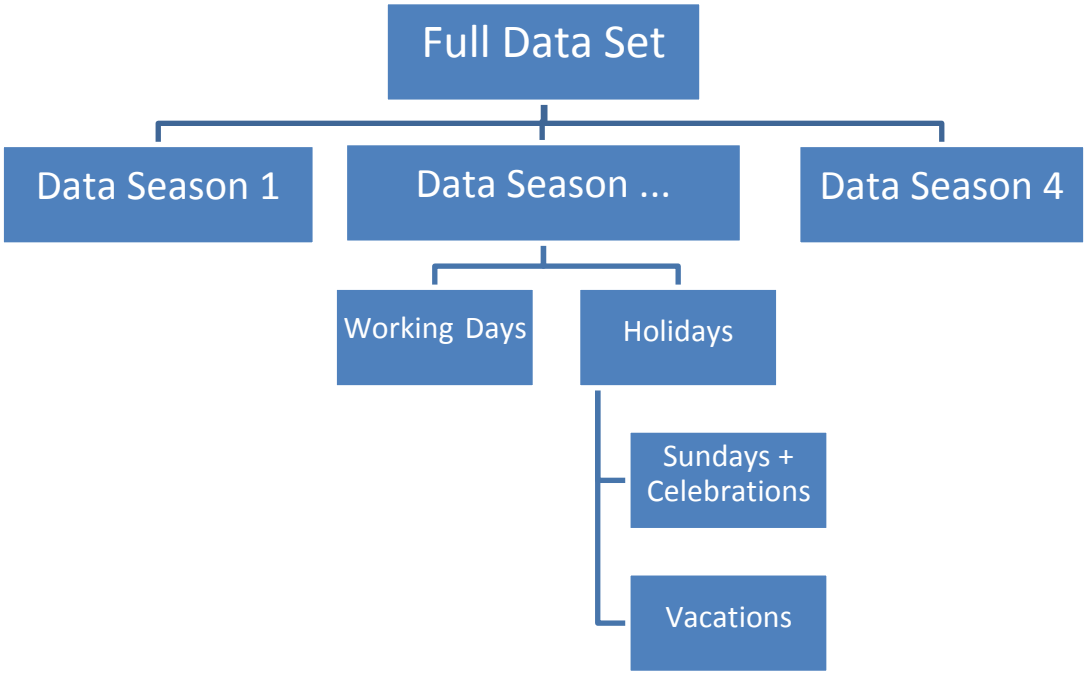


Figure 14: Representation of categorization of dataset

Aggregation. In this step data from different categories (seasonal and working/holidays) are aggregated in order to extract common profiles. Two main methods are adopted for aggregating profiles: Mean Absolute Deviation (MAD) and Root Mean Square Error (RMSE). MAD is basically the average of the absolute deviations from the mean of the considered time series, while RMSE describes the root deviation between observed data and estimated data. Th purpose is to acquire a unique profile for each customer in a given time categorization (season and/or working/holidays). This step strictly depends from the kind of clusterization required, so that aggregation could be planned or not.

Extraction Results. The final step is the comparison of different profiles calculated across the considered seasons and days (working/holidays) categorization. In this case the jointed evaluation of prosumer habit during different seasons and in different days, should provide the required information about his energy usage.

As an example, the following analysis shown in Figure 15, focuses on a single residential user and was applied on data collected monitored over a span of 3 years, from 2014 to 2016. The dataset consists of electrical consumption data logged at 15-minute-long intervals. The core idea is to highlight user-specific various patterns regarding energy consumption, as these fluctuate among years, within the four seasons and, at the same time, between workdays, Saturday and Sunday as well (4 seasonal categories and 3 daily categories). Basically, each box represents the mean energy consumption for 1 year and one specific daily category; seasonal trends are depicted with different colours.

Figure 15: Comparison of four seasons hourly customer profiles based on different days categories in 2014, 2015 and 2016 (source CERTH)

Finally, it is possible to rebuild profiles in a unique time-series for each customer, where the different seasons are depicted in a one continuous trend for a better understanding of yearly consumptions. In Figure 16 there is a comparison of electricity consumption for different seasons. As it is shown the hourly mean electricity consumption receives higher values during winter and summer. As a matter of fact, the specific user is using air condition for his cooling and heating. During the years 2014, 2015 the electricity consumption is higher in winter than in summer, in contrast with year 2016, in which summer has higher values. The reasons may that the winter was milder, or the user decided to use different applications for his heating.

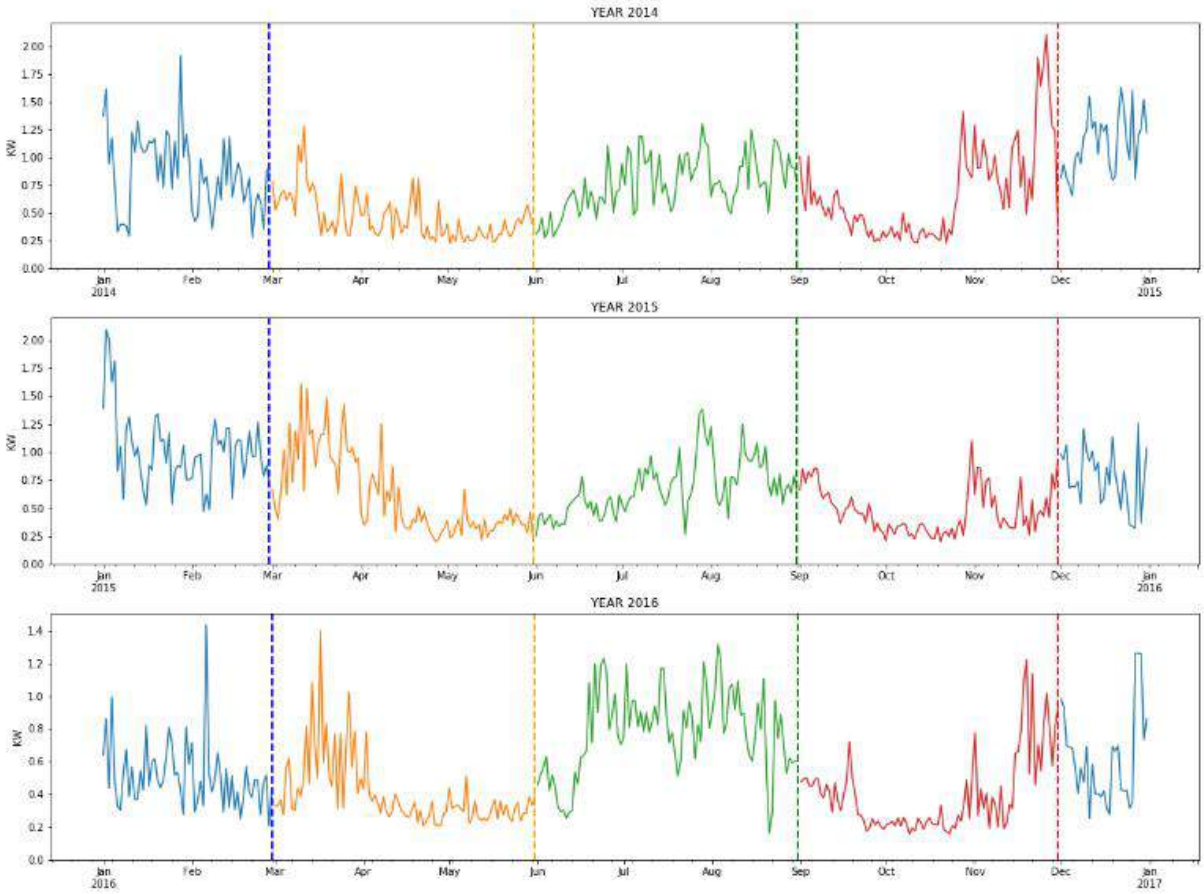


Figure 16: Profiling of energy customer time series for 2014, 2015 and 216; seasonal profiles in different colors (source CERTH)

5. Clustering

The clustering process can be considered the core of the big data layer, where the most valuable information is extracted from the received data. It can be a useful tool in order to support the DSOs, aggregators and others energy operators for characterization of their portfolio. Considering eDREAM objectives, the benefits derived from this tool are twofold:

- flexibility evaluation, that means helping energy market operators to quickly identify portions of customers with given flexibility potentials;
- portfolio assessment, that is a general assessing of prosumers for proper market participation.

In Figure 17 the overall clusterization process is shown; it is based on a generic application for project purposes. The importance of pre-processing is fundamental at this stage, in order to filter and categorize data so that the algorithm can process them (see chapter 3 for details about pre-processing). The second step deals with the attribute selections, where the proper feature must be defined to extract valuable information from clusterization. In the third step the algorithm for the clusterization must be chosen according to the objective of the clusterization and the properties of data to cluster. There are a lot of methods and categories of algorithm and some of them are described in followings. After the calculation and iterative computation, the number of the cluster must be validated and consolidated. Finally, the information is available for operators ready to be extracted or interpreted.

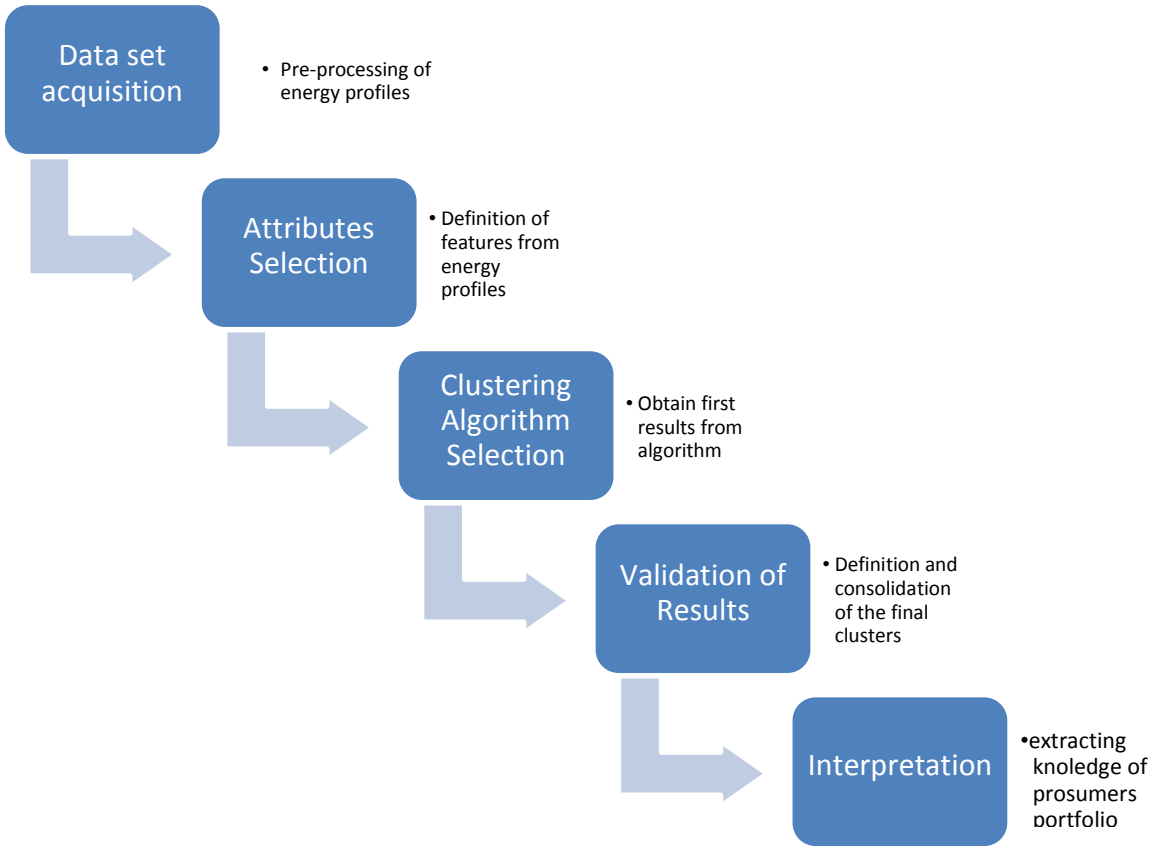


Figure 17: Steps of clusterization procedure (source Atos based on (Halkidi, 2001))

5.1. Attributes' selection

Since the smart meters roll out, a huge amount of energy profiles was suddenly available within an hour sampling or even in minutes. In addition to this first condition, the transaction to a new model market and the adoption of

demand response mechanism have required a relevant improvement in customers profiling and related pattern recognition. So, it is now important to identify the proper set of customers that could better participate in a given market scenario or in a specific DR programme. The clusterization could be very helpful in this way, but of course the right conditions and indexes, must be taken into account to obtain the demanded clusters. It is important to choose the appropriate set of attributes in order to extract valuable information from data sets and find out a valuable clustering of prosumers.

With the smart grids and smart meters revolutions a lot of studies have been published in order to define the most relevant indicators for customers characterization. Moreover, when considering fast event and flexibility potentials of prosumers, the attributes do not fit into the classical records anymore.

In (Azaza M., 2017) a review of the most relevant features for smart meters data clustering is carried out and reported inTable 1:

Feature	Description
Total Consumption Power	Total consumed power during period of the study
Mean load	Mean load´ during defined time period
Load Factor	Ratio of the average load during predefined time period to the maximum load during that period
Base Load	Mean load from 2 am – 5 am
Morning Maximum	The maximum consumption observed between 6 am – 10 am
Midday Load	Mean observed between 12 am and 2 pm
Evening Consumption	Mean observed load during evening peak relative the mean load during the rest of the day

Table 1: Profile indicators as input features for clustering loads pattern. Source: (Azaza M., 2017)

Load Profiling module and pre-processing activity would be helpful in defining and extracting those features, nevertheless those are only the basic ones that can be useful for a more general purpose clusterization. Other specific features useful for assessing flexibility among the customers could be:

- **Peakness**: that is the relation between the evening peak and the consumption throughout the rest of the day.
- **Rise time:** described by the time in the afternoon when the consumption starts to increase, usually associated to the arriving home time of its inhabitants after work.
- **Decline time**: is the time in the evening when the consumption starts to decrease rapidly, usually associated to people's bed time when most of the appliances are turned off.
- **Off-peak consumption**: is about the presence of significant consumption during the off-peak time, usually during night and early morning when the majority of the population is at sleep.

Moreover, considering the specific needs and the business of the U.K. pilot of KiwiPower, it is possible to define a set of parameters for back-up generators. When considering programmable generation some specific voices must be taken into account according to the participation in the different energy markets (ancillary, reserve, day-ahead,

etc.). These parameters have a direct influence on the choice of assets selection when answering a market petition. Table 2 reports those parameters that can be useful to set constraints and filters during the clustering of generators.

Parameter	Description
Asset Type	Generator, CHP, Chiller, Battery etc.
Asset Response Time	The time it takes the asset to respond to a signal to dispatch.
Maximum Turndown	Percentage of assets nominal power
Maximum Dispatch Duration	In seconds or minutes the maximum time a site can be dispatched for.
Fuel Capacity	How much fuel is required to run an asset, if required.
Manual/Automatic	The method in which an asset can be dispatched.
Number of Assets	How many assets are located on a site.
Site Availability	When a site can potentially participate in DR.

Table 2: Specific parameters for clusters of back-up generators

5.1.1. Attributes based on Pilots datasets

It would be possible to plan the clustering process by using two or more attributes or features for the first set of data received from ASM (137 smart meters data with 15 minutes frequency sampling). The dataset holds the values of energy absorbed and produced by low voltage prosumers connected to ASM distribution grid and it is useful only for a preliminary evaluation of the tools and services to be developed in Task 4.2. The following attributes have been defined for the analysis of this dataset and their possible adoption will be evaluated in the next version of the deliverable D4.2 where actual tool development will be tested in a more extended dataset.

First of all, it is important to clarify the following points deriving from load profiling and pre-processing sections:

- Organizing and dividing the data set in different kind of customers (domestic, shop, industries, etc)
- When considering domestic customers, it would be better separate single-phase ones (≤6.6 kW for ASM data) from three-phase ones (>10kW)
- selecting the time interval in the day when the maximum consumption is generally absorbed
    - according to ASM dataset, it is possible to identify 14:00-21:00 interval (it is also possible to take into account the whole day, but it would likely affect the dispersion of records from the centres of the clusters) – defining the index *n* as described below;

      $t_i \in \boldsymbol{T}$ where $\boldsymbol{T} = [t_1, \dots, t_n]$ is the set of the energy measurements in a day (t_i - t_{i-1} = 15');
- separating working days from holidays - defining the index *r* as described hereby:

  $d_k \in \boldsymbol{D}$ where $\boldsymbol{D} = [d_1, \dots, d_r]$ is the set of the monitored days.
- Identifying customer with missing and/or false data, that could bring to a fork:

- omitting the customers with too much missing measures;
- identifying the number of customers to be clustered - defining the index *m* as described below:

$u_j \in \boldsymbol{U}$ where $\boldsymbol{U} = [u_1, \dots, u_m]$ is the set of the considered customers.

Under these assumptions we are able to define the following 3 attributes.

Attribute 1: Time instant of energy peak

The first attribute we are going to investigate is the series of the instant time when each customer consumes the higher amount of energy, that could give light over the peak power consumptions in a given day time (Dent, 2015) (Dent I., 2012). The procedure for the proposed calculation is:

1. finding the maximum value of absorbed energy *Ea*, for each customer in all the considered daily time Intervals:

$$E_{MAX_{i,j,k}} = max\left(Ea_{t_i}\right)_{j,k}$$

and the time instant t_i when the peak occurred

$$t_{M_{j,k}} = t(E_{MAX_{j,k}})$$

2. calculating the distance *x* from the initial instant time t_1 of the daily Interval to the instant of the peak power

$$x_{j,k} = d[(t_{1_{j,k}}), (t_{M_{j,k}})]$$

3. calculating the mean $\overline{x}_j$, for the whole set of days in the year, between the distances $x_{j,k}$, of the maximum absorbed energy for each customer

$$\overline{x}_j = \frac{\sum_{k=1}^{r} x_{j,k}}{r}$$

4. calculating the standard deviation of the sequence $x_{j,k}$

$$\sigma_j = \sqrt{\frac{\sum_{k=1}^{r}(x_{j,k} - \overline{x}_j)^2}{r}}$$

and obtaining the sequence of *m* standard deviations of the distances of peak power instant for each customer *j*

5. normalizing (min-max normalization) the standard deviations

$$\sigma_j^* = \frac{\sigma_j - min(\sigma_j)}{max(\sigma_j) - min(\sigma_j)}$$

The sequence σ_j^* represents the first attribute for the clusterization.

The process can be replicated for the time instant associated to the second (or even lower) highest power measurement out of the neighbourhood of the maximum, in order to detect other peak instants during the day.

This attribute basically describes time instant or the interval where maximum energy consumption or production uses to be metered; it can be adopted both for generators and loads, thus it is good for prosumers.

Attribute 2 Energy peak

Under the aforementioned assumptions, the second attribute should be the quantity of the absorbed energy during peaks (Dent I., 2012) (Dent, 2015). It is calculated in a very similar way to the previous one through the proposed following procedure:

1. finding the maximum value of energy absorbed (generated) *Ea*, for each customer in the considered daily time intervals

$$E_{MAX_{i,j,k}} = max\left(Ea_{t_i}\right)_{j,k}$$

obtaining a matrix where each $E_{j,k}$ element, represents the maximum value of absorbed (consumed) energy for each customer (rows) for each day (columns)

$$E_{MAX} = \begin{bmatrix} E_{MAX_{1_1}} & \cdots & E_{MAX_{1_r}} \\ \vdots & \ddots & \vdots \\ E_{MAX_{m_1}} & \cdots & E_{MAX_{m_r}} \end{bmatrix}$$

2. calculating the mean $\overline{E}_j$ for each days of the year, among the daily maximum values of absorbed (consumed) energy, for each customer

$$\overline{E}_j = \frac{\sum_{k=1}^{r} E_{MAX_{j,k}}}{r}$$

3. calculating the standard deviation σ_j of the sequences $E_{j,k}$

$$\sigma_j = \sqrt{\frac{\sum_{k=1}^{r}(E_{j,k} - \overline{E}_j)^2}{r}}$$

4. obtaining the sequences of the m standard deviations of maximum values of absorbed energy, for each customer *j*
5. normalizing (min-max normalization) the standard deviations

$$\sigma'_j = \frac{\sigma_j - min(\sigma_j)}{max(\sigma_j) - min(\sigma_j)}$$

The sequence σ'_j is the second attribute for the clusterization.

This attribute tries to quantify the amount of the energy peak consumed or produced by a prosumer, regardless the instant (in which part of the day it uses to happen) and its duration. The clusterization between both attributes could bring to a distribution of the clusters that can be easily plotted in a bi-dimensional graph:

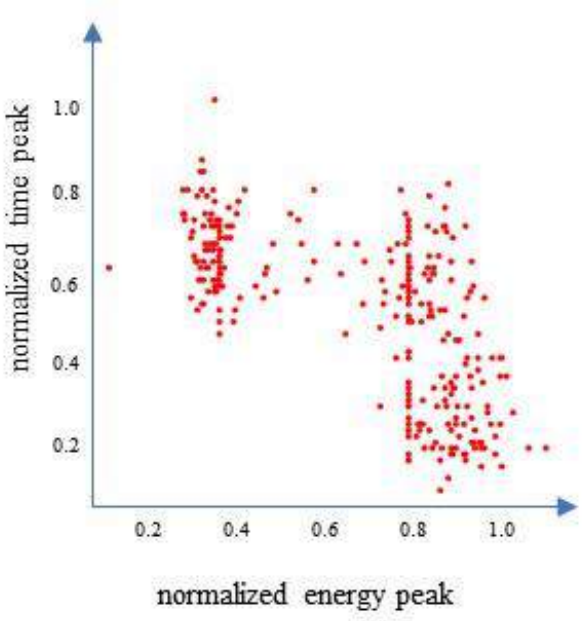


Figure 18: Graphical representation of 2-attributes clusterization

Attribute 3 Peak duration

This third attribute will help in understanding the duration of the energy peak absorbed or consumed by the prosumers. Starting from Attribute 1, it is possible to set a coefficient ε to define an interval of energy values in the neighborhood B around the $E_{MAX_{j,k}}$ so that $B = (E_{MAX_{j,k}}, E_{MAX_{i,j,k}} - \varepsilon)$. Thus, a time interval T is defined around the time instant $t_{M_{j,k}}$ when the energy peak occurs. The procedure is defined as follows:

1. for each prosumer *j* and each day *k* of the dataset, it is possible to identify a set of $T_{j,k}$.
2. calculating the mean $\overline{T}_j$ for each days of the year, among the daily intervals of energy peak absorbed (consumed), for each customer

$$\overline{T}_j = \frac{\sum_{k=1}^{r} T_{j,k}}{r}$$

3. calculating the standard deviation σ_j of the sequences $T_{j,k}$

$$\sigma_j = \sqrt{\frac{\sum_{k=1}^{r} (T_{j,k} - \overline{T}_j)^2}{r}}$$

4. obtaining the sequences of the m standard deviations of maximum values of absorbed energy, for each customer *j*
5. normalizing (min-max normalization) the standard deviations

$$\sigma'_j = \frac{\sigma_j - min(\sigma_j)}{max(\sigma_j) - min(\sigma_j)}$$

The sequence σ'_j is the second attribute for the clusterization. According to the initial value ε to be set, it is possible to consider the size of the time interval *T*. Considering the three of attributes at the same time into a unique

clusterization (see Figure 19 as possible visualization output) it will be possible to understand when energy peak use to occur, their magnitudes and how long they persist.

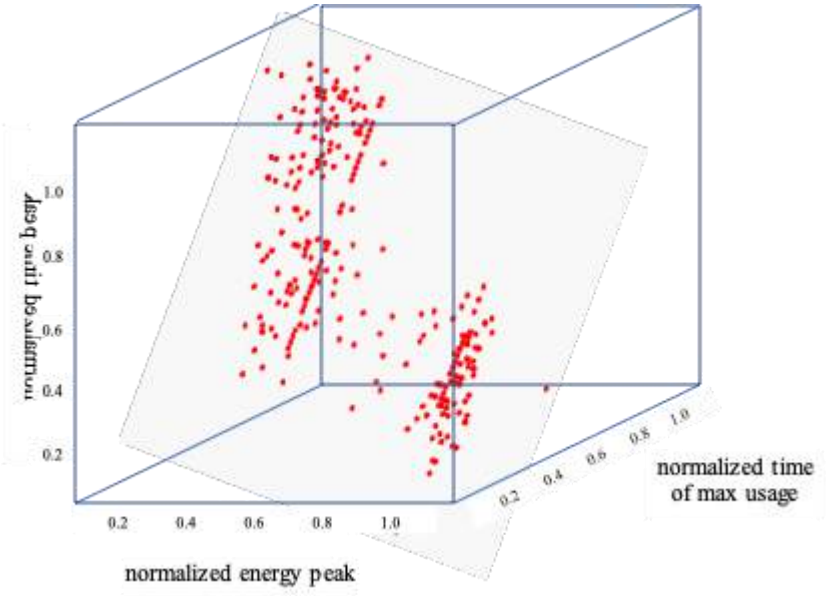


Figure 19: Graphical representation of 3-attributes clusterization

5.2. Algorithm's selection

It is possible to find a huge number of clustering techniques in literature, since data mining tools were becoming popular in many disciplines and for a lot of applications. Even the categorization of the different methodologies is not always unambiguous, but they basically depend by the point of view the categorization is carried out. According to (Halkidi, 2001), clustering algorithms can be classified according to:

- The type of data used for input;
- the way the algorithm evaluates the similarity between data points.
- The basic theory a clustering algorithm is based on (e.g. fuzzy theory, statistics).

According to these three fundamentals criteria it is possible to identify the following categories of clustering techniques (Halkidi, 2001) (Jin, 2017):

- Partitional Clustering: a class of iterative algorithm that divide the observations in a number of portions and, by re-calculating the centroids of theses portions, assign data points to each cluster.
- Hierarchical Clustering: a family of algorithms that tries to merge or separate clusters in order to create a tree of hierarchical relationships between clusters;
- Density-Based Clustering: these methods try to condensate group of data points within a given epsilon distance;
- Model-based Clustering: this category of models is independent of distance metrics and it "is based on fitting a probability distribution over the clusters" (Jin, 2017).

Starting from these four main categories, a large variety of techniques and methods can be found in literature, trying to mix different algorithms and leverage advantages and strengths of two categories with new formulations.

K-means Algorithm

K-means clustering (MacQueen, 1967) is an unsupervised learning technique, basically adopted when no categories or groups have been previously defined. This algorithm is aimed at finding groups in the data, the variable K is generally associated with the number of groups. Among the multitude of clusterization techniques, K-means belongs to the Partitional Clustering, a category of algorithms that basically tries to directly divide and detach the data into a set of separated clusters. Without entering much detail, partitional methods try to identify an integer number of partitions that would satisfy a given constraint or function. This function possibly will highlight the local or global structure of the data and through an iterative procedure (Halkidi, 2001).

The key concept of this method is to preliminary assume a number of clusters K, with associated centroids for each one of them. The process for k-means clusterization is drawn in Figure 20. There is no a unique rule for centroids identification, but as a general assumption they should be positioned as much as possible distant from each other. In this case experience of data scientist could be helpful because the selection of different centroids position will affect the final results. Otherwise some evaluation techniques can be used in order to find the initial number of clusters that better solve the optimization; elbow method and silhouette index are reported as a couple of the most popular techniques for k-means initialization (further details on evaluation indexes can be found in Section 5.4. Once the centroids are assigned and located, the distance of each element of data set from nearest centroid is calculated and all datapoints are associated to the closest centroid. Once all elements have been associated to a centroid, a preliminary partition of data set is done. In the following step k centroids must be re-calculated for the previously identified clusters and the Euclidean distance of each element from centroids must be re-calculated as well in an iteration process. At each iteration, the centroids may vary their position and the different element of data set can be assigned from cluster to cluster. The loop will keep on running until no variation in centroids positions is calculated. “This produces a separation of the objects into groups from which the metric to be minimized can be calculated” (M. Matteucci, n.d.). Thus, basically the final goal of this algorithm is the minimization of the objective function:

$$E = \sum_{j=1}^{k} \sum_{i=1}^{k} \left\| x_i^{(j)} - c_j \right\|^2$$

where:

- $x_i^{(j)}$ is the i-th element of j-th cluster
- c_j is the j-th centroid, and
- $\left\| x_i^{(j)} - c_j \right\|^2$ is the distance between data point and cluster centre.

Each centroid of a cluster is a collection of feature values which define the resulting groups. Examining the centroid feature weights can be used to qualitatively interpret what kind of group each cluster represents.

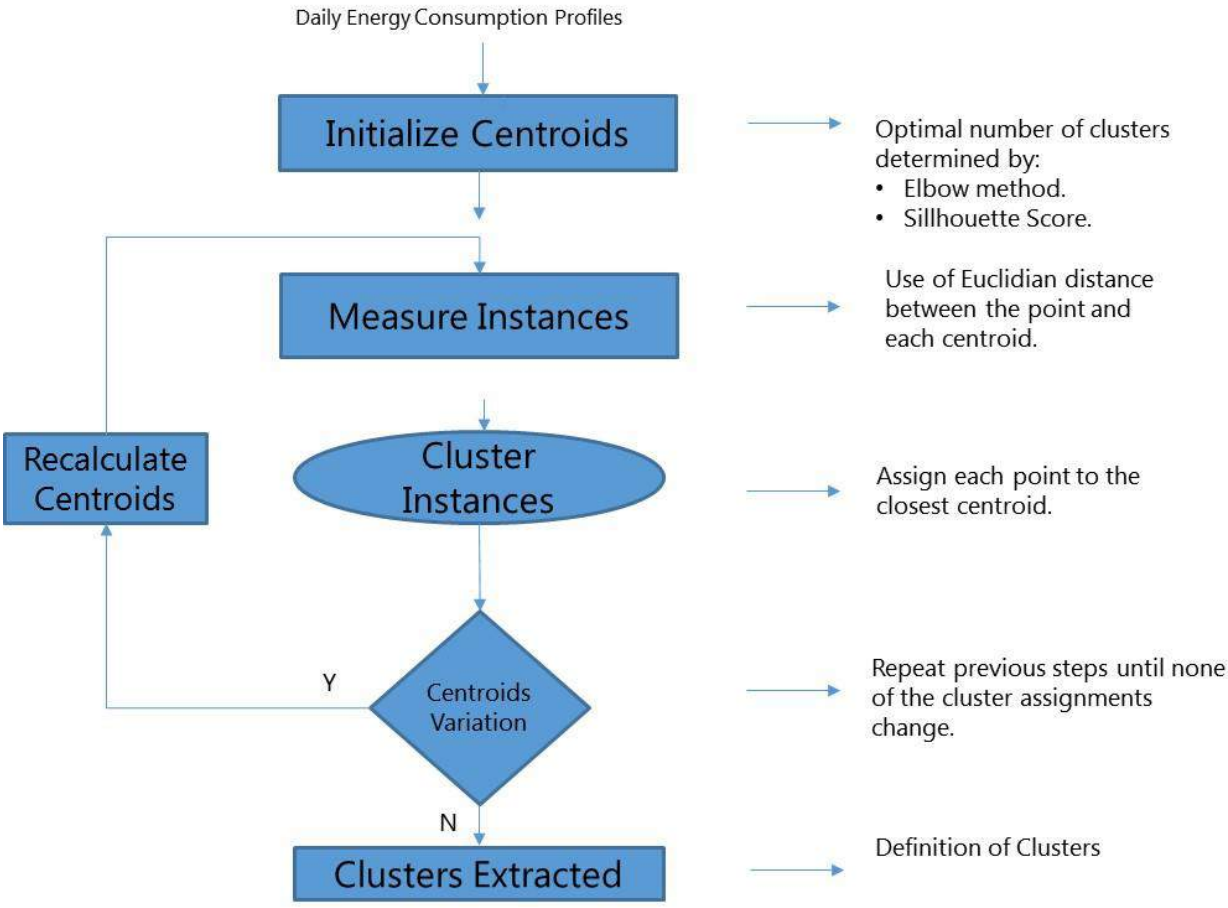


Figure 20: Process of clusterization with k-means

K-means is a generic algorithm that can be adopted in different domains and scenarios and its main advantages are:

- it is the simple to implement and to run;
- It works really well with large datasets.
- it can computationally be faster than other techniques (when k is small).
- it is able to identify well-circumscribed clusters;

But, on the other hand, it could have some weaknesses too: (M. Matteucci, n.d.)

- the initialization of the cluster could be a critical limit;
- It could happen that the set of elements next to c_j is empty, so that c_j cannot be updated. This event should require additional development;
- The results depend on the metric adopted to quantify $\left\| x_i^{(j)} - c_j \right\|^2$.
- The results depend on the value of k.

For the aforementioned reasons, an appropriate pre-processing is strongly needed for an effective use of this algorithm and compensating the described weaknesses. In this case we are using some evaluation indexes for

assessing the right number of k of algorithm initialization. Moreover, the adoption of deep learning techniques is also considered for the minimization of the Euclidean distance.

Density-based Spatial Clustering of Applications with noise (DBSCAN) Algorithm

DBSCAN (Ester et al. 1996) belongs to the category of the unsupervised data mining techniques. Essentially the algorithm groups together points that are close to each other (points with many nearby neighbours), marking as outliers points that appear alone in low-density regions. DBSCAN defines, as clusters dense groups of points. The idea behind DBSCAN is that if a particular point belongs to a cluster, it means that next to it will appear many other points as well.

Two parameters are necessary for DBSCAN algorithm (do Prado, 2017):

- **eps**: the minimum distance between two points. It means that if the distance of two points is lower or equal to this value (eps), these points are marked as neighbors. The **eps** neighborhood of a point:

$$N(p) = \{qeD|dist(p,q) <= Eps\}$$

- **minPoints:** the minimum number of points to form a dense region. For example, if we set the minPoints parameter as 5, then we need at least 5 points to form a dense region.

Given eps and minPoints categorize the objects into three exclusive groups.

- A point is considered as a **core point** if the corresponded number of points within **eps** is greater than a predefined number of points (**minPoints**)**.** These are the points at the interior of a cluster.
- A **border point** has fewer than **minPoints** within **eps,** but belongs to the neighborhood of a core point.
- A **noise point** is any point that is not core point nor a border point.

Each core point forms a cluster together with the points that are reachable within its *eps* radius. Two points are considered "*directly density-reachable*" if one of the points is a core point and the other point is within its **ε** radius. Larger clusters are formed when directly density-reachable points are chained together.

Unlike some other clustering techniques, DBSCAN does not require all data points to be assigned to a cluster. The DBSCAN algorithm repeats the following process shown in Figure 21 until all points have been assigned to a cluster or are labeled as visited. Some advantages of DBSCAN are:

- The ability to discover clusters of arbitrary shapes (spherical, elongated, linear) and noise.
- Working with spatial datasets.
- No need to predefine the number of clusters.

The minor disadvantage of DBSCAN is that it is sensitive to parameters.

- If the **eps** value is too small, the largest part of the dataset will be not clustered. On the other hand, if the value is getting high values, clusters will merge and the majority of the data points will end up in the same class. The decision of **eps** value should be based on the distance of dataset (k-distance graph could be used), but in general small eps values are preferable.
- Considering the parameter minPoints, a general rule is that it can be derived from the number of dimensions (D) in the dataset, as minPoints >= D+1. Larger values are usually better for data sets with

noise and will form more significant clusters. The minimum value for minPoints must be 3, but as larger the dataset is, the larger the value of minPoints should be.

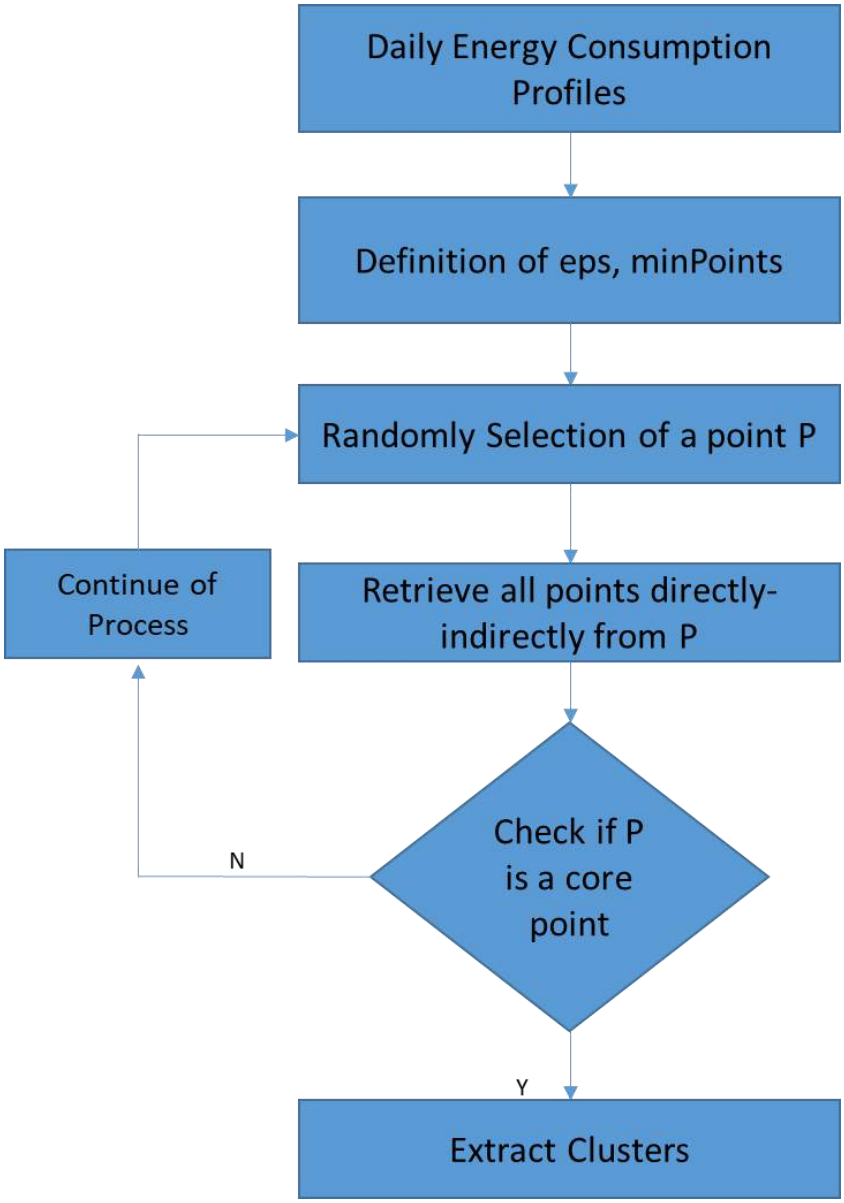


Figure 21: Process of clusterization with DBSCAN

More methodologies to be proof are:

- **Mean Shift Clustering:** is another unsupervised algorithm method based on kernel density estimation (KDE) usually Gaussian kernel. It works with a sliding-window-based algorithm that attempts to find dense areas of data points. It is a centroid-based algorithm meaning that the goal is to locate the center points of each group/class, which works by updating candidates for center points to be the mean of the points within the sliding-window. These candidate windows are then filtered in a post-processing stage to eliminate near-duplicates, forming the final set of center points and their corresponding groups. (Seif, 2018) The advantages are the simplicity and the parameter dependency, kernel bandwidth, which controls the algorithm´s result. On the other hand, it is slow in implementation.

- **Expectation-Maximization (EM) Clustering using Gaussian Mixture Models (GMM)**: is a sort unsupervised clustering algorithm which try to solve the K-means problems when cluster centers are very close together or clusters are not circular. In order to solve these problems Gaussian Mixture Models (GMM) are implemented. It obtains more flexibility than K-means; assuming that values are Gaussian distributed, but it needs two parameters to be defined: the mean and deviation. The optimization algorithm uses to find both parameters is Expectation–Maximization (EM) (Seif, 2018).

5.3. Parametrization and numbers of clusters

One of the most important aspect of the clusterization procedure is the initialization phase, where the dimension of the data set and the initial number of k centroids is set. This is a critical point especially when using k-means, where the number of clusters should be set a priori, and it determines the number of groups of points. Centroids are set in a random position and in the first calculation only the distance of each data point from the centroid is calculated. Further iterations are devoted to re-positioning the locations of centroids, but they do not recalculate their number.

To figure out this limit of partition based algorithm, the different indices are frequently adopted to verify the distance of the data point within a cluster, to its centroid; there can be a large variety in literature and five of them have been selected and described in Section 5.4, due to their typical application in electric measurements analysis. When analysing the output of a clusterization, the evaluation with different indices would consolidate the result, giving a measure of the coherence of clusters.

In our first tests of clusterization of the prosumers data set from ASM, the algorithm has been executed in loop iterations, changing the number k of the centroids to each iteration (i.e. from 2 to 10). The output of each clusterization was evaluated applying at least two of the five indices. The result was an array of values with the calculation of indices for each initialization k value. In this way it is possible to choose the value of k that originates the best final result, by means the best value of associated index (see Table 3). Moreover, it could happen that two or more values of k (when they are very close and the data set is very large), could have very similar evaluation results. The adoption of different indices should help the post-assessment phase when selecting the optimal number of k. Keeping in mind that each index has a specific correspondence with calculated Euclidean distance the optimal value of k can be easily choose. This procedure could require the human supervision, but it can be easily converted in unsupervised task assigning basic rules of priorities.

	k_1	…	k_n
index 1			
index 2			
Index …			

Table 3: Number of centroids k, vs evaluation indices results

Some criticalities may occur when a dataset is too large and there could be problems for those data to be ingested by the algorithm. In this case some machine learning technique can be adopted in order to smooth the problem. Feature extraction is a machine learning technique aimed at identifying features in a measured dataset, these features are supposed to be informative and non-redundant. Therefore, it is possible to select a subset of features containing all the relevant information from the input data, in order to perform the task in a more agile, easy and

fast way through the reduced dimensionality. This technique is detailed in the following subsections and two main procedures to be adopted are described.

5.3.1. Features Extraction

Sometimes datasets hold many variables that, in order to process them, would require so memory as well as computer power, and not all of them are relevant or useful for the problem to be solved. Therefore, the feature extraction has a great relevance in this case. Feature extraction select those variables or attributes of the dataset that allow describing the complete dataset with enough precision; that mean, those relevant or useful variables, that used as input data for an algorithm, optimize the calculation by reducing the number of needed resources and obtaining better accuracy. In some cases, features extraction is confused with reduction of the so-called "curse of dimensionality" (Sheety, 2019), which reduce the number of variables in a dataset creating new variables by their combination.

One features extraction problem, when an algorithm selected is training with fewer inputs, is overfitting. This may be the case when the algorithm`s accuracy fit good for training data, but it decreases for test data or new values. Another problem is related with the dimensionality; Hughes Phenomenon shows that as the number of features increases, the classifier's performance increases as well until we reach the optimal number of features. Adding more features based on the same size as the training set will then degrade the classifier's performance (Shetty, 2016).

The three general classes algorithms of extraction features are:

- **Filter Methods:** each variable of dataset is applied by a statistical measure, once values are obtained, those variables with the highest value are maintained and those with the lowest values are deleted. Some examples which apply this method are: Chi squared test, information gain and correlation coefficient scores.
- **Wrapper Methods:** are based on the features` combination between all values available, the better combination with a specific algorithm will be selected.
- **Embedded methods:** These methods host an internal mechanism that selects the attributes which improve the algorithm while it is being created. The most characteristic algorithm within the embedded methods is the regulation (or penalization) method, since in the optimization of a predictive algorithm additional constraints are added that decrease the number of coefficients, obtaining a less complex model. (Brownlee, 2014)

In addition to these methods, with the arise of popularity of neural networks, new algorithms have appeared, one of the most powerful one in the field of feature extraction is the autoencoder model.

Autoencoder

Autoencoder algorithm is implemented with artificial neural networks, generating new data from the knowledge acquired through the compression of input variables, capturing the most relevant characteristics of these; and subsequently, it reconstructs the output based on the information achieved. One of the great advantages of this algorithm is that not affected by noise. The autoencoder is divided into two parts:

- **Encoder:** the number of input variables is compressed, applying the following coding function

$$h = f(x)$$

- **Decoder:** try to reconstruct the inputs based on the previously information collected. The function is applied

$$r = g(h)$$

The ensemble encoder-decoder that forms the autoencoder is governed by the following equation:

$$d(f(x)) = r$$

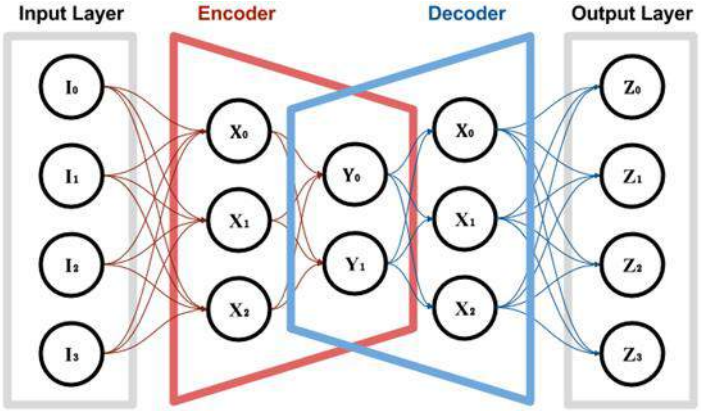


Figure 22: Autoencoder diagram (Source: (Zucconi, 2018))

STL decomposition

It is a robust and versatile time series decomposition algorithm. It applies in seasonal series with seasonality where stationary component can change over time. Its performance consists of taking a period of time (window) equal to stationarity and measuring the moving average centred by applying the additive or multiplicative model. It can handle any type of seasonality, although sometimes the use of the Fourier transform is required to detect this stationarity. The formula that applies is shown below, with the three components on which depends: seasonal, trend and rest.

$$Y_v = T_v + S_v + R_v v = 1, \ldots, NwithN = datapoints$$

The advantages it presents compared to traditional algorithms, SEATS and X11 decomposition methods are: (Cleveland, 1990)

- Seasonal components may change over time, within a range that can be controlled by the user; in addition to controlling the smoothness of the trend cycle;
- It is robust against extreme points

On the other hand, the algorithm presents better results for the additive decomposition compared to the multiplicative one; and does not allow the variation of the calendar automatically

Figure 23 corresponds to the application of an autoencoder without STL decomposition, where it is observed that for a time series data, it has only obtained two clusters with a much more prominent group, group 0, than the other group. It is reflected that the extraction of features helps in clustering, but the noise and stationarity that is intrinsically introduced in the algorithm does not allow creating more clusters.

```
For n_clusters = 2 The average silhouette_score is : 0.7251398885202318
For n_clusters = 3 The average silhouette_score is : 0.7239253602490284
For n_clusters = 4 The average silhouette_score is : 0.6954202339368055
For n_clusters = 5 The average silhouette_score is : 0.5087722410022836
For n_clusters = 6 The average silhouette_score is : 0.5263901686289619
```

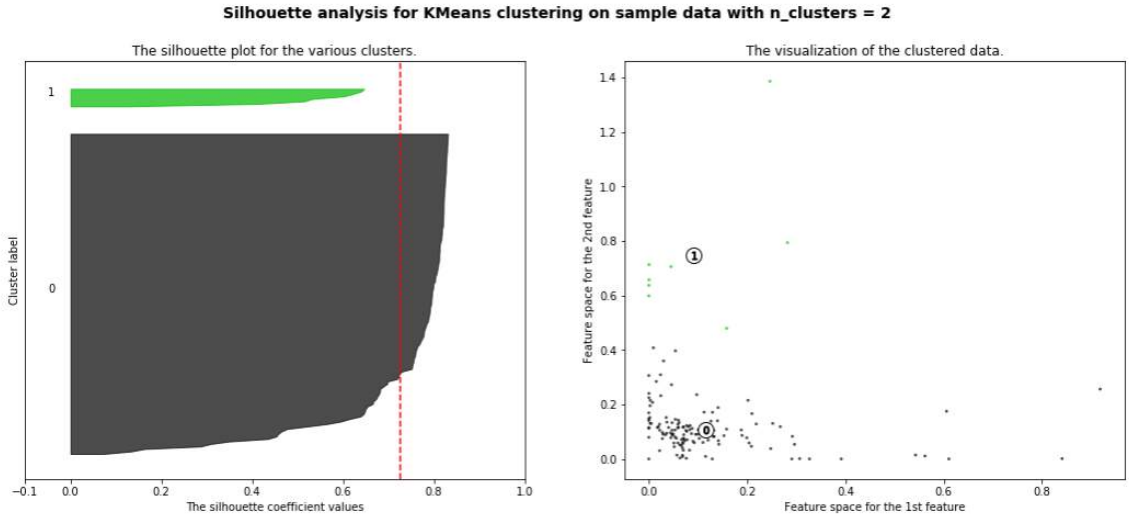


Figure 23: Clustering without Autoencoder and STL

On the other hand, Figure 24 reflects the application of autoencoder with STL, where the clusterization is much more differentiated. The STL decomposition algorithm has divided the seasonal trend of each user allowing the autoencoder to extract the most relevant characteristics, as can be seen by having many more clusters and discrete.

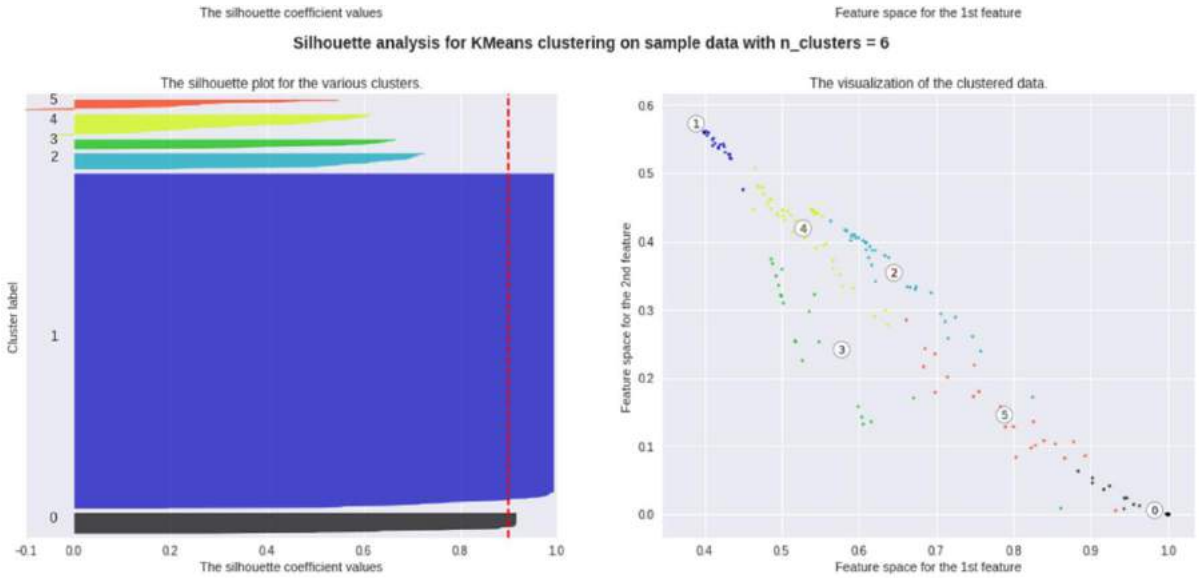


Figure 24: Clustering with Autoencoder and STL

As mentioned in section 2, the Big Data Tool Engine layer is composed with different and independent microservices but related to each other. Each of these microservices is composed of one or several tools, with different objectives such as analysis or pre-processing.

With this microservices configuration and different tools, the Big Data Clustering at Multiple Scales module is structured. Different customers data can be analyzed with the aim of clustering, applying the k-means tool along with the rest of the proposed algorithms according to the accuracy they offer; or by applying the tool that contains

the Autoencoder and STL decomposition algorithms. Besides that, it is possible to apply both tools and sort out the results with the best accuracy.

5.3.2. Principal Component Analysis and Recursive Feature Elimination

Two of the most common techniques of feature extraction are **Principal Component Analysis** (PCA) and **Recursive Feature Elimination** (RFE).

Principal Component Analysis

PCA is a fast and flexible unsupervised method for dimensionality reduction in data. The purpose of PCA is to reduce the dimensionality of the dataset, transforming the old dataset to a new one that contains most of the information of the old one, but with lower dimensions. PCA uses the eigenvalues and eigenvectors from the covariance matrix, which is computed before, to extract the principal components. If there are *n* observations with *p* variables, then the number of distinct principal components is min(*n-1*, *p*). This transformation is defined in such a way that the first principal component has the largest possible variance and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components. The resulting vectors are an uncorrelated orthogonal basis set.

Mathematically, the transformation is determined by the p-dimensional vectors of weights or coefficients $w_k = (w_1, \dots, w_p)_k$ that map each row vector x_i of X to a new vector of principal component scores tk(i) = x(i) * w(k) for i = 1, …, n, k = 1, …, l.

The first principal component contains the maximum variance of the information of the dataset. X is a n x P centred matrix (n observations and p features). The first component is given by:

$$MaxVar(Xw1)$$

$$Constraint: w_1^T w_1 = 1$$

The meaning of the constraint is to avoid picking arbitrarily large values for the vector **w1**. If that constraint is absent, it arbitrarily picks a large value for the vector **w1** in order to maximize the objective function.

The second principal component is defined by the following equations:

$$MaxVar(Xw2)$$

$$Constraint: w_2^T w_2 = 1$$

$$Constraint2: w_1^T w_2 = 0(Orthogonal)$$

One more constraint is added to make sure that both the components are orthogonal.

The figure below shows a simple example of PCA transformation.

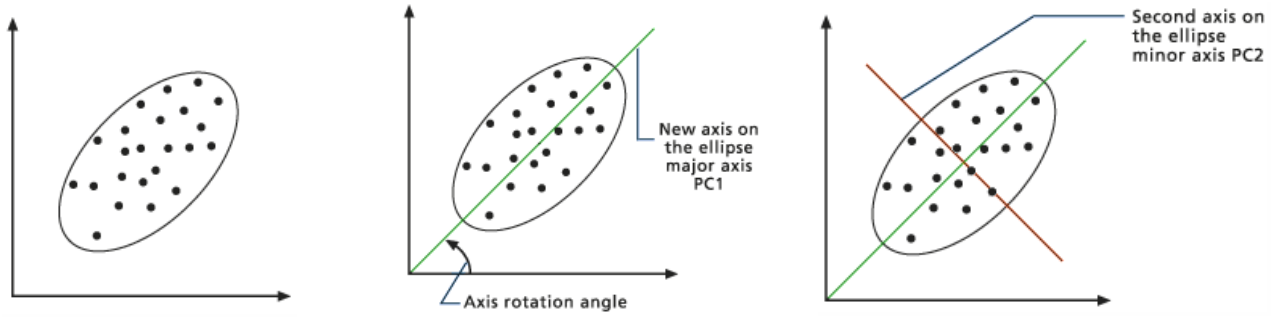


Figure 25: elliptical boundary of the points (left), first principal component axisPC1 (center), second principal component axis PC2 (right). Source: (Arcgis, 2016)

An ellipse is calculated to bound the points in the scatterplot. After the major of ellipse is determined and becomes the new x-axis, the first principal component (PC1), depicts the greatest variation because it is the largest transect that can be drawn through the ellipse. The direction of PC1 is the eigenvector, and it is magnitude is the eigenvalue. The angle of the x-axis to PC1 is used in the transformation as the angle of rotation. A line vertical to PC1 is calculated, which is the second principal component (PC2) and the new transformed y-axis. The new axis describes the greatest variance that is not described by the first principal component. Using the eigenvectors, the eigenvalues, and the calculated covariance matrix of the input, a linear formula is extracted, which defines the shift and the rotation. This formula is used for the transformation of each cell value relative to the new axis.

Recursive Feature Elimination

The Recursive Feature Elimination (RFE) belongs to the feature selection methods. RFE fits a model and removes the weakest feature (or features) until the specified number of features is reached.

It uses the model accuracy to identify which attributes contribute the most to predicting the target attribute. RFE attempts to eliminate dependencies that may exist in the model.

RFE requires a specified number of features to keep, however it is often not known in advance how many features are valid. To find the optimal number of features cross-validation is used with RFE to score different feature subsets and select best scoring collection of features. An example of a cross-validation for determination of the number of features is given in Figure 26.

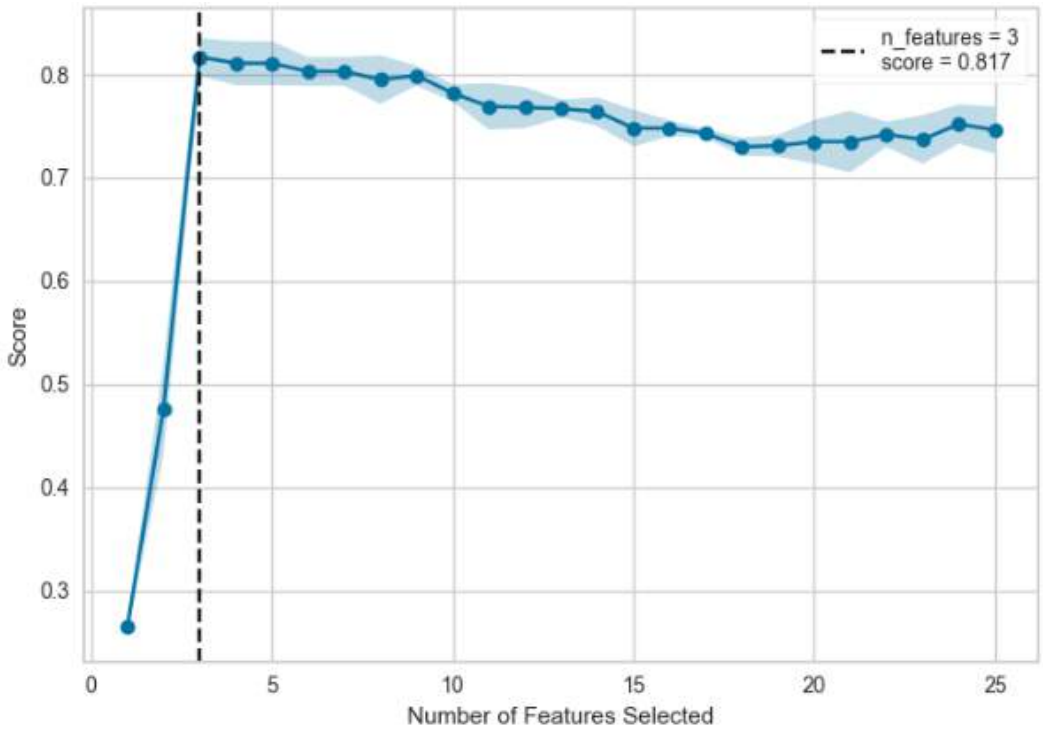


Figure 26: RFECV for SVC (Source: (scikit-learn, 2019))

As it can be shown, the curve jumps to an excellent accuracy when the three informative features are captured, then gradually decreases in accuracy as the non-informative features are added into the model. The shaded area represents the variability of cross-validation, one standard deviation above and below the mean accuracy score drawn by the curve.

5.4. Evaluation index

The number of clusters n_c in which the data are categorized with the clusterization it is generally unknown "*a priori*". This basically means that the number of clusters would change when clustering the very same data with different algorithms. Evaluation analysis is helpful when assessing the performances of one or more techniques and it is generally intended to measure the consistency of the proposed clusters. This kind of analysis is helpful to achieve two main goals:

- understand the optimal number of clusters for a given algorithm;
- compare the performances of different clustering techniques.

The analysis assesses the optimal n_c through the maximization of the intra-cluster similarities and/or the minimization of the inter-cluster similarities. That means it is possible to use evaluation index to understand the proper number of clusters for a given algorithm or the same performance of clusterization using different algorithms.

Among the large variety of methods and indexes available in literature, the most relevant ones for energy and power applications have been selected.

Elbow Method

It is one of the most common evaluation method and adopted for general purpose clusterization. This method is somehow vague or ambiguous, in the sense that it can provide a roughly approximation of the best number of clusters. It is a graphic method based on the calculation of the within-cluster sum of squared errors (SSE) and plot them in a bi-dimensional graph. For example, if a k-mean clusterization would run for ten times considering k (clusters number from 1 to 10), it is then plotted a 10 points graph. The point corresponding to the curve's slope variation (the "elbow") it is generally assumed as the optimal number of clusters; for the sample of Figure 27, it corresponds a value of 2 clusters.

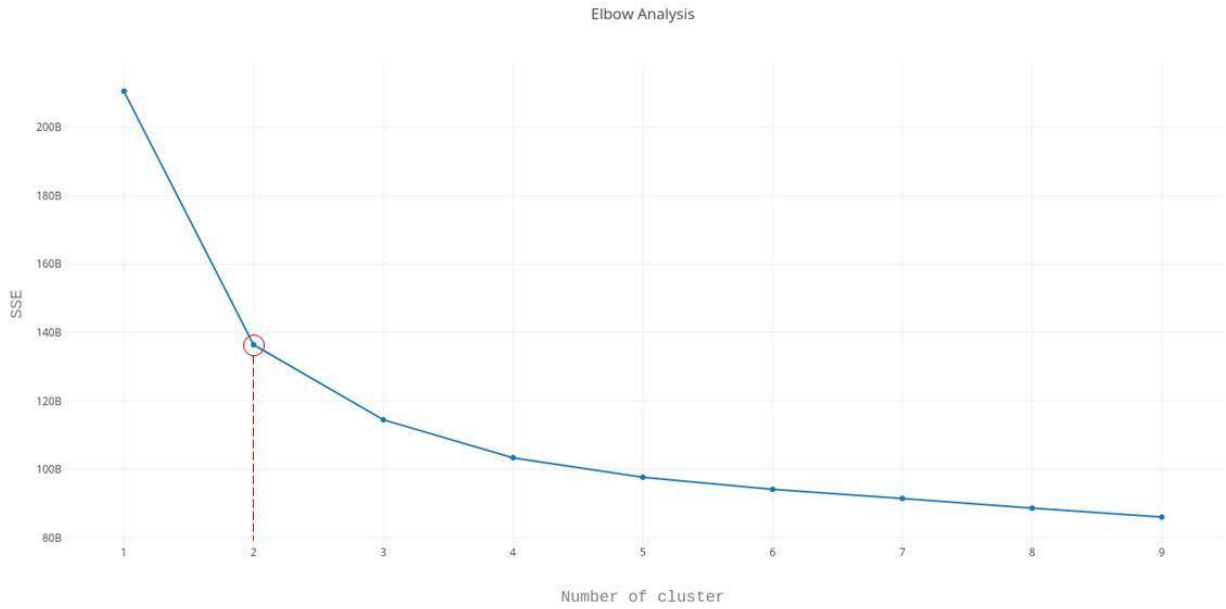


Figure 27: Sample of Elbow plot (Source: Atos elaboration on ASM dataset)

Mean Index Adequacy (MIA)

The MIA gives a value which relies on the amount by which each cluster is compact; if the members of a cluster are close together, the MIA is low

$$MIA = \sqrt{\frac{1}{K}\sum_{k=1}^{K}\sum_{r} d^2(r_{(k)}, C_{(k)})}$$

where C is the set of clusters centers and r_k is the k-th member of a given cluster (Dent I., 2012) (Dent, 2015).

Cluster Dispersion Indicator (CDI)

The CDI calculates the intra-cluster consistency (distance among records in the same cluster as for the MIA) and inter-clusters distance (distance between members of different clusters). Thus, it is able to give back at the same time, the measure of the compactness of each cluster and the differences among clusters.

$$CDI = \frac{1}{\hat{d}(C)}\sqrt{\frac{1}{K}\sum_{k=1}^{K}\hat{d}^2(R_k)}$$

"where C is the set of clusters centers and R_k is the set of members of k-th cluster" (Dent I., 2012), (Chicco G., 2003).

Silhouette Index

"Silhouette analysis can be used to study the separation distance between the resulting clusters. The silhouette plot displays a measure of how close each point in one cluster is to points in the neighbor clusters and thus provides a way to assess parameters like number of clusters visually. This measure has a range of [-1, 1]." (Scikit-Learn, n.d.)

$$Sil = \frac{1}{N}\sum_{c_k \in C}\sum_{\boldsymbol{x}_i \in \boldsymbol{c}_k}\frac{b(\boldsymbol{x}_i, c_k) - a(\boldsymbol{x}_i, c_k)}{max\{a(\boldsymbol{x}_i, c_k), b(\boldsymbol{x}_i, c_k)\}}$$

where

$$a(\boldsymbol{x}_i, c_k) = 1/|c_k| \sum_{\boldsymbol{x}_j \in c_k} d(\boldsymbol{x}_i, \boldsymbol{x}_j) =$$

and

$$b(\boldsymbol{x}_i, c_k) = \min_{c_l \in \boldsymbol{C}/c_k}\left\{1\Big/|c_k| \sum_{\boldsymbol{x}_j \in c_k} d(\boldsymbol{x}_i, \boldsymbol{x}_j)\right\}$$

Values $a(\boldsymbol{x}_i, c_k)$ and $b(\boldsymbol{x}_i, c_k)$ are measures of cohesion and isolation, respectively. Entities with a Silhouette width close to 1 are well clustered while those with a lower width can be considered intermediate (Fernandes, 2017) (Rousseeuw, 1987), (Dent I., 2012), (Chicco G., 2003) and (Alves, 2018).

Davies-Bouldin Indicator (DBI)

The Davies-Bouldin Index, as described by authors themselves, "indicates the similarity of clusters which are assumed to have a data density which is a decreasing function of distance from a vector characteristic of the cluster" (Davies, 1979).

The DB index is defined as:

$$DB_{nc} = \frac{1}{nc}\sum_{i=1}^{nc} R_i$$

where R_i is the maximum value of R_{ij} and R_{ij} is the similarity measures between two clusters (C_i and C_j) and it shall satisfy well-defined constraints (such as $R_{ij} > 0$, $R_{ij} = R_{ji}$, and others defined in (Davies, 1979) and (Halkidi, 2001)).

According to the authors, DBI does not depend by the number of the clusters or by the adopted method to perform the clusterization, thus basically the objective is to minimize the index in order to find the best clusterization solution.

6. Profiles Segmentation

Segmentation component is responsible for recognizing the load profiling customer´s pattern that had been clustered in Big Data Clustering at multiple scale module and assigning new customer to one of them.

In order to achieve this allocation of new consumptions, the segmentation module has different possibilities to be implemented. For this case, Artificial Neural Networks (ANN) have been chosen. The application of these algorithms provides a series of advantages that allow to classify the profiles of new clients without the need to group them again, once trained.

Big data layer provides clusters, ensembles of customers based on its common characteristics. Each of these clusters is assigned a label, so all customers belonging to that cluster will have the same label. This allows to provide the algorithm, the input data for its training. The objective of this training is to allow the new data generated in the pilots, as well as those generated in real time, to be labelled and assigned to one of the existing clusters.

A neural network is a series of algorithms that recognize underlying relationships in a set of data, based on interconnections neurons, enables them to learn as more input data they receive. In addition, they save information in a self-organizing way, responding to failures in a tolerant way, with great flexibility and almost in real time. This implementation avoids the scalability and stability problem to the extent that the algorithm can be retraining every so often with the objective of sustaining the accuracy in an optimal range.

Neural networks are an ensemble of layers organized in interconnected nodes which contain a non-linearity activation function responsible for transmitting a signal from one neuron to another. The input layer collects each customer data and the cluster assignment from big data clustering element. Hidden layers process via system of weighted connections the input to an output layer where the result is obtained. These weights are being adjusting in learning process with backpropagation algorithm. Backpropagation is a supervised learning process that occurs with each epoch (each time the network is presented with a new input) which is used with gradient descent optimization algorithm in order to adjust the weight values by calculating the gradient of loss function. Once the neural network is trained until the satisfactory accuracy is attained, it ought to be profited from new customers as an analytical tool. New input data, customer profile data, runs through the network which predicts the cluster it belongs to.

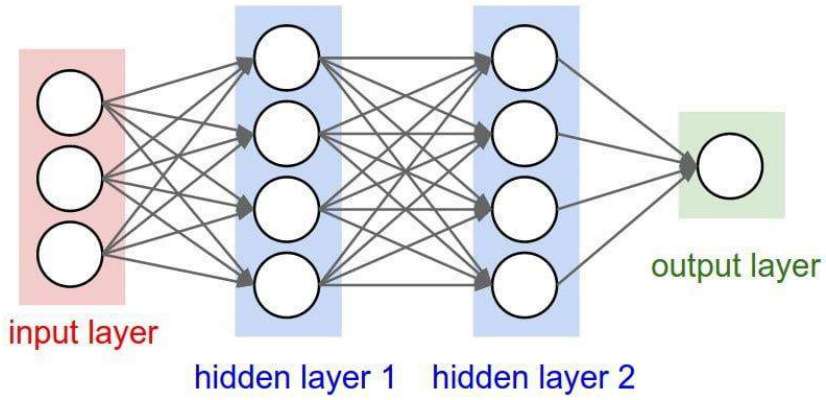


Figure 28: ANN Representation (Source: (Rosebrock, 2017))

Networks based on the *Adaptive Resonance Theory* (ART Neural Networks) serve to classify patterns in an unsupervised way, which means that, the network forms groups according the resonance similarities that it recognized and, in the case, that no groups correspond to the existing groups, they create a new one. (Calderón,

2003) . Among the large variety of ART types, Neural Networks ART1 and ART2 have been selected for segmentation purpose.

ART1 serves for the recognition and classification of the entry pattern. The network detects if the information belongs to a known category, calculating the similarity percentage between the input and the stored prototype; check the classes and, if there is no such category, creates a new one. It constituted by two layers: one input layer with N neurons and another output layer with M neurons

ART2 network is an extension of the ART1 network which the main difference is that it supports real values; like the previous network, it serves to classify patterns in unsupervised way. Both networks, ART1 and ART2, possess the same architecture (Calderón, 2003).

7. Pilot Application

eDREAM project pilots are carried out by ASM and KIWI companies, located in Italy and United Kingdom respectively. In the second version of the deliverable associated to this task 4.2, the methodologies and tools described in this document will be applied and tested at pilots' sites. Here it follows a brief introduction to the pilots.

The ASM's electric grid is characterized by a large number of distributed renewable energy sources embedded in the MV and LV distribution networks: 1 biomass plant, 5 hydro power generators and 1,234 photovoltaic (PV) units are currently connected directly to the MV and LV distribution networks reaching the total installed capacity of around 70 MW. In this regard, it is worth pointing out that, based on this energy mix shown in Figure 29, 200 GWh of the 400 GWh absorbed yearly, are produced by DER systems connected to the MV/LV grid of ASM, 70 GWh of which are from intermittent RES.

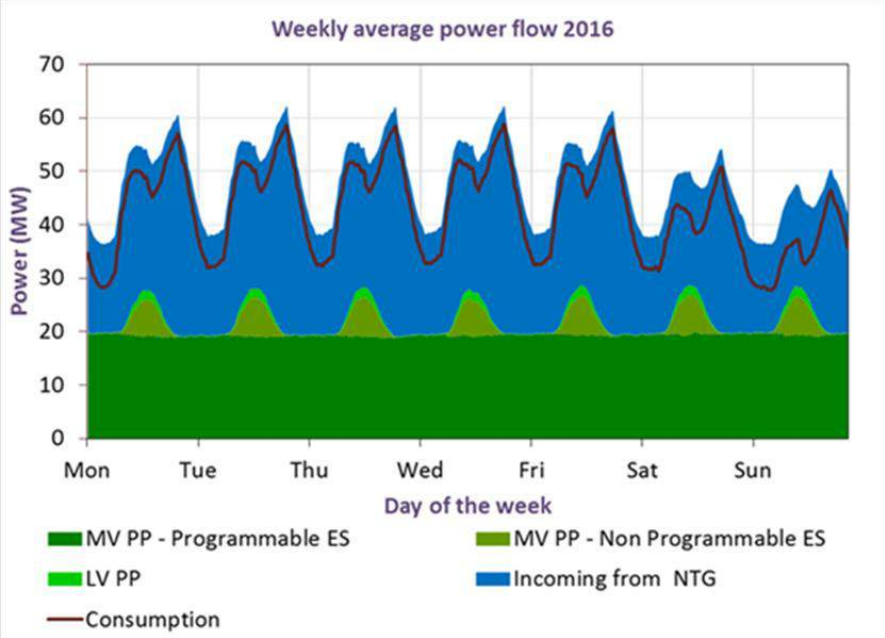


Figure 29: Average weekly profile of consumption and production in the ASM power network

ASM's power distribution network is connected to HV grid through 3 substations. There are also 6 sorting MV/MV substations and 615 MV/LV substations. MV cable and overhead lines are long about 622 km, whereas LV lines length is about 1418 km. Nowadays the energy customers are about 65,500, 98% of which have an electronic meter. In 2015 about 50% of the total consumption was covered by RES. In 2015 the local power network received renewable energy from 1,062 power generation plants (mainly PV arrays) using renewable sources, such as sunlight, water and biomass. In 2015 the total electric power generated from RES was 110GWh approximately (30MW from PV). Moreover, in 2015 a thermoelectric plant produced 74,9 GWh from waste material. It is worth pointing out that among 1,062 plants connected to the ASM power network, about 770 PV arrays (72.5%) are classified as domestic being installing at residential and commercial premises.

This peculiarity of ASM system perfectly fits with the purpose of eDREAM project due to the high penetration of RES in the low voltage sections and the potential flexibility that a large number of prosumers could theoretically offer. In 2010 the deployment throughout the power network of smart meters with four data acquisitions per hour (each 15 minutes) was completed, reaching all the 65,500 end users. The application of profiling, clusterization (with k-mean algorithm) and segmentation techniques will be demonstrated on a dataset of about 1500 prosumers (connected to LV and MV grid) and equipped with the smart meter with 15 minutes frequency sampling. Data are

gathered by using specific software tools: Id-Spectrum, produced by Ericsson, which allows for visualization of all the data collected from smart meters (SM) as well as for error detection in AMR measurements and their validation through specific algorithms; Terranova, a data management software integrating Id-Spectrum billing functions and allowing for remote operation (e.g. smart meter disconnection, power rating changes, calculation of penalties for reactive imbalance). Some functionalities offered by these software tools were specifically implemented for ASM TERNI. Moreover, statistical data analyses were carried out using Microsoft Excel; standard deviation from reference values was figured up, thereby supporting the analysis of economic flows.

KiWiPOWER will provide a data set for testing purposes from Commercial and Industrial customers, allowing the validation of the load profiling, big data clustering at multiple scale (with k-means algorithm) and customer segmentation modules at the U.K. pilot site. Also, KiWiPOWER will have access to a hundred of residential users data sets (minute by minute data). A different portion of KiWiPOWER Pilot will be devoted to the validation of the clusterization via DBscan technique and to the interaction of the big data layer with the linked components "*Multibuilding DR characterization*" (for the aerial scanning through thermal and lidar cameras) and "*VPP and active Microgrid Flexibility Profiling*".

Figure 30: Aerial view of the building to be scanned

In this section of the U.K. pilot, a strong interaction with the Task 3.4 "*Aerial survey techniques for DR potential estimation*" is foreseen, because in this task an aerial scan of the building with thermal, optical and lidar cameras will be done through a drone and in parallel, measurements from smart meters will be analyzed in order to estimate the potential capacity for a possible DR participation of those customers. Images and data processed in T3.4 will then ingested into the Big Data Clustering at multiple scale module and clustered through DBScan tool.

8. Conclusions and next steps

This document provides information for the development of the load profiling and customers clusterization tools in the eDREAM modular platform. Three modules of the "*Load Profiling*", "*Big Data Clustering at Multiple Scale*" and "*Customer Segmentation*" have been described in relation to the overall architecture of platform and with particular regards of WP3 modules. For each one of these components, the methodology and the tools to be adopted have been described with the purpose of achieving a stable and scalable solution.

A first exploration analysis has been performed with a reduced sample dataset from a pilot partner, in order to test pre-processing, profiling and clusterization; thus, first results are shown for methodology proofing. Finally, pilot sites have been described with their main features for the profiling and clustering applications.

During the following months of the task the algorithms and the software tools will be fully developed. In the same time, the methodologies will be implemented with the data from pilots. *Load Profiling* will be accomplished and tested on dataset from ASM pilot in order to verify its real application with measurements of real prosumers; also it will be adopted with forecasted data to extract trends over specific time frames in the future. Big Data Clustering at Multiple Scale will be developed with the two algorithms described in section 5: K means will be devoted to Italian pilot application, while DBScan will be designed for U.K. pilot applications. During this phase it will be important to ensure the correct operation of the module with the input data from "Load Profiling", "VPP Generation Modelling & Forecasting" and "Multi-building DR characterization through thermal optical and LIDAR Information fusion". Finally, "Customer Segmentation" will be developed through ANN applications and it is expected to work in parallel with the clustering module when analyzing large set of prosumers.

References

Alves, G. (11 de December de 2018). *Unsupervised learning with K-means*. Obtenido de medium.com: https://medium.com/infosimples/unsupervised-learning-with-k-means-3eaa0666eebf

Amudhavel J., S. D. (2015). Big Data Scalability, Methods and its Implications: A Survey of Current Practice. *In Proceedings of the 2015 International Conference on Advanced Resesearch in Computer Science Engineering and Technology (ICARSET '15)*, (p. 5). New York, NY, USA. doi:https://doi.org/10.1145/2743065.2743121

Arcgis. (2016). Obtained from http://desktop.arcgis.com/en/arcmap/10.3/tools/spatial-analyst-toolbox/how-principal-components-works.htm

Azaza M., W. F. (2017, December). Smart meter data clustering using consumption indicators: responsibility factor and consumption variability. (ELSEVIER, Ed.) *Energy Procedia, 142*, 2236-2242. doi:https://doi.org/10.1016/j.egypro.2017.12.624

Box, G. E. (2015). *Time series analysis: forecasting and control. John Wiley & Sons.*

Brownlee, J. (6 de October de 2014). *An Introduction to Feature Selection*. Obtenido de https://machinelearningmastery.com/an-introduction-to-feature-selection/

Brownlee, J. (17 de August de 2018). *A Gentle Introduction to SARIMA for Time Series Forecasting in Python*. Obtained from https://machinelearningmastery.com/sarima-for-time-series-forecasting-in-python

Calderón, J. L. (March de 2003). *ART Teoría de la Resonancia Adaptativa*. Obtenido de https://es.slideshare.net/mentelibre/teora-de-resonancia-adaptativa-art

Cassandra. (2019). Obtained from http://cassandra.apache.org/

Castanedo, F. (2013). A review of data fusion.

Chicco G., N. R. (2003). Customer characterization options for improving the tariff offer. *Power Systems, IEEE Transactions on,, 18*(1), 381-387.

Cleveland, R. B. (1990). *STL: A seasonal-trend decomposition. Journal of official statistics, 6(1), 3-73.*

Cygnus. (2019). Obtained from https://github.com/telefonicaid/fiware-cygnus

Dasarathy, B. V. (1997). *"Sensor fusion potential exploitation-innovative architectures and illustrative applications," Proceedings of the IEEE, vol. 85, no. 1, pp. 24–38.*

DASK. (2018). Obtenido de https://docs.dask.org/en/latest/

Davies, D. a. (1979). A Cluster Separation Measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 1*(2), 224–227.

Dent I., C. T. (2012). "An approach for assessing clustering of households by electricity usage. *the 12th Annual Workshop on Computational Intelligence.* Heriot-Watt University, Edinburgh, U.K. Obtenido de arXiv:1409.0718

Dent, I. (2015). *Deriving knowledge of household behaviour from domestic electricity usage metering.* Nottingham, U.K.: PhD thesis, University of Nottingham. Obtenido de http://eprints.nottingham.ac.uk/27972/1/thesis_master.pdf

do Prado, K. S. (2017). *How DBSCAN works and why should we use it?* https://towardsdatascience.com/how-dbscan-works-and-why-should-i-use-it-443b4a191c80.

Durrant-Whyte, H. F. (1988). "Sensor models and multisensor integration". *International Journal of Robotics Research, vol. 7, no. 6,* , pp. 97–113.

Ester M., K. H. (1996). A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. *KDD, Vol.96, N. 34;* , 226-231.

Fernandes, M. P. (2017). Segmentation of Residential Gas Consumers Using Clustering Analysis. *Energy, 10*(2047). doi:https://doi.org/10.3390/en10122047

Fiware. (2019). Obtained from https://www.fiware.org/community/smart-energy/

Gandomi, A., & Haider, M. (2015). Beyond the hype: Big data concepts, methods, and analytics. *International Journal of Information Management,, 35*(2), Vol. 35, 137-144.

George E. P. Box, G. M. (2015). *Time Series Analysis: Forecasting and Control.* (Wiley, Ed.)

Hadoop. (2019). Obtained from https://hadoop.apache.org/

Hale, J. (2018). *Deep Learning Framework Power Scores 2018*. Retrieved from https://towardsdatascience.com/deep-learning-framework-power-scores-2018-23607ddf297a

Halkidi, M. B. (December de 2001). On Clustering Validation Techniques. (K. A. Publishers, Ed.) *Journal of Intelligent Information Systems, 17*(2-3), 104-145. doi:https://doi.org/10.1023/A:1012801612483

Ishwarappa, & Anuradha, J. (2015). A brief introduction on Big Data 5Vs characteristics and Hadoop technology. *Procedia Computer Science, 48*, 319–324;. Retrieved from https://ac.els-cdn.com/S1877050915006973/1-s2.0-S1877050915006973-main.pdf?_tid=ec0e11d3-746d-42e7-859e-f3d

ISO/IEC. (2001). *ISO/IEC 9126-1:2001, Software engineering -- Product quality -- Part 1: Quality model.*

Jain, A. (2016, September 17). *The 5 Vs of Big Data.* Retrieved from www.ibm.com: https://www.ibm.com/blogs/watson-health/the-5-vs-of-big-data/

Jin, L. L. (2017). Comparison of Clustering Techniques for Residential Energy Behavior Using Smart Meter Data. *AAAI Workshops - Artificial Intelligence for Smart Grids and Buildings*, (págs. 260-266). San Francisco, CA USA. Obtenido de https://www.aaai.org/ocs/index.php/WS/AAAIW17/paper/view/15166/14673

Karafiloski A., M. E. (2017). Blockchain solutions for big data challenges: A literature review. *IEEE EUROCON 2017 -17th International Conference on Smart Technologies*, (pp. pp. 763-768.). Ohrid. doi:doi: 10.1109/EUROCON.2017.8011213

Keras. (2019). *Keras: The Python Deep Learning library*. Obtained from https://keras.io/

Lee Jay, B. B.-A. (2014). Recent Advances and Trends of Cyber-Physical Systems and Big Data Analytics in Industrial Informatics. *Int. Conference on Industrial Informatics (INDIN)*. Porto Alegre, Brazil.

M. Matteucci. (n.d.). *A Tutorial on Clustering Algorithms*. Retrieved from Polimi: https://home.deib.polimi.it/matteucc/Clustering/tutorial_html/kmeans.html

MacQueen, J. (1967). Some Methods for Classification and Analysis of Multivariate Observations. *n Proceedings of 5th Berkley Symposium on Mathematical Statistics and Probability,*, *1 - Statistics*, págs. 281-297. Barkeley, U.S.A.

Marr, B. (2015, March 19). *why only one 5 Vs big data really matters*. Retrieved from www.ibmbigdatahub.com: https://www.ibmbigdatahub.com/blog/why-only-one-5-vs-big-data-really-matters

Michael M., M. J. (2007). Scale-up x Scale-out: A Case Study using Nutch/Lucene. *2007 IEEE International Parallel and Distributed Processing Symposium*, (págs. 1-8). Rome, Italy. doi:doi: 10.1109/IPDPS.2007.370631

Numpy. (2019). Otained from https://numpy.org/

Orion. (2019). Otained from (https://github.com/telefonicaid/fiware-orion)

Pandas. (March de 2019). *Python Data Analysis Library*. Otained from https://pandas.pydata.org/index.html

Percentage points for a generalized ESD many-outlier procedure. Technometrics, 2. 1.-1. ((1983)). *Rosner, B.* .

Pfeil, M. (29 de October de 2010). *Why does Scalability matter, and how does Cassandra scale?* Obtained from https://www.datastax.com/dev/blog/why-does-scalability-matter-and-how-does-cassandra-scale

PyAstronomy. (2019). *Welcome to PyAstronomy*. Obtenido de https://www.hs.uni-hamburg.de/DE/Ins/Per/Czesla/PyA/PyA/index.html

Python. (2019). Obtenido de https://www.python.org/: https://www.python.org/

R. C. Luo, C.-C. Y. (2002). “Multisensor fusion and integration: approaches, applications, and future research directions,” IEEE Sensors Journal, vol. 2, no. 2, pp. 107–119.
Rosebrock, A. (2017). *Moral Robots*. Obtenido de https://moral-robots.com/resources/a-simple-neural-network-with-python-and-keras/
Rosner, B. (May de 1983). Percentage points for a generalized ESD many-outlier procedure. (L. o. Taylor & Francis, Ed.) *Technometrics, 25*(2), págs. 165-172. doi:10.2307/1268549
Rousseeuw, P. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 53-65.
Sagiroglu S., S. D. (2013). Big data: A review. *2013 International Conference on Collaboration Technologies and Systems (CTS)* (págs. 42-47). San Diego, USA: doi: 10.1109/CTS.2013.6567202.
scikit-learn. (2019). *scikit-learn Machine Learning Python*. Obtenido de https://scikit-learn.org/stable/
Scikit-Learn. (n.d.). *Selecting the number of clusters with silhouette analysis on KMeans clustering*. Retrieved from https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html
SciPy.org. (2019). Obtenido de https://www.scipy.org/
Seif, G. (5 de February de 2018). *The 5 Clustering Algorithms Data Scientists Need to Know*. Obtenido de https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68
Shetty, B. (16 de January de 2016). *Curse of Dimensionality*. Obtenido de https://towardsdatascience.com/curse-of-dimensionality-2092410f3d27
StatsModels. (2017). *Welcome to Statsmodels's Documentation*. Obtained from https://www.statsmodels.org/stable/index.html#
Telefonica Investigación y Desarrollo. (2019). *iotagent-node-lib*. Retrieved from
https://github.com/telefonicaid/iotagent-node-lib
TensorFLow. (2019). *An end-to-end open source machine learning platform*. Obtenido de https://www.tensorflow.org/
tsfresh. (2019). *tsfresh*. Obtenido de https://tsfresh.readthedocs.io/en/latest/index.html
Wiki SciPy. (2019). Obtenido de https://en.m.wikipedia.org/wiki/SciPy
XGBoost. (2016). *XGBoost Documentation*. Obtenido de https://xgboost.readthedocs.io
Zucconi, A. (14 de 03 de 2018). *Introduction to Autoencoder*. Obtenido de https://www.alanzucconi.com/2018/03/14/an-introduction-to-autoencoders/