UCLouvain

École polytechnique de Louvain

Machine learning for electricity consumption forecasting in energy communities.

Author: **Lambert Misselyn**
Supervisor: **Axel Legay**
Readers: **Mathias de Schietere de Lophem, Laurent Francis**
Academic year 2022–2023
Master [120] in Computer Science

Abstract

The climatic consequences linked to the exploitation of fossil fuels invite us to rethink our mode of energy production and turn to renewable energies. An energy community is a group of people, who collectively manage their production, consumption and eventually their storage of electricity. This type of renewable energy project could develop in the years to come as EU Member States are required to follow the recommendations of the "Clean Energy for All Europeans" package adopted in 2019. WeSmart is a Belgian company based in Brussels that sets up and manages energy communities. Thanks to smart meters, they can collect data from their users and develop monitoring tools to help members of energy communities to consume more efficiently. The data collected also enables them to develop a user optimisation and recommendation service. The idea is to provide community members with personalised advice on how to maximise their consumption of community-generated electricity. To achieve this, it may be useful to be able to forecast the community's electricity production and each member's consumption in the near future.

Acknowledgement

I would like to thank my supervisor, Axel Legay, for giving me the opportunity to carry out this work by proposing me this personalised thesis subject, which has enabled me to develop my knowledge in a field that I am passionate about.

I would like to express my sincere gratitude to Mathias de Schietere de Lophem, who provided me with invaluable advice and feedback throughout my work. They played a decisive role in the completion of this work.

I also give my thanks to Laurent Francis for accepting to read my work.

I wish to acknowledge Malagi Legast for introducing me to Professor Legay, and whose excellent thesis has been a source of inspiration for me.

I am deeply thankful to family, my friends and my girlfriend for their unconditional support and their confidence in my abilities. Their presence throughout my academic journey has helped me to overcome all the challenges.

I am grateful to the Catholic University of Louvain-La-Neuve for giving me the opportunity to increase my knowledge and for providing me with the necessary resources to conduct my research.

Contents

Introduction

Energy communities offer an innovative way of producing and consuming electricity, notably thanks to renewable energies. However, managing energy flows efficiently in this kind of decentralised network can be challenging. Smart meters enable renewable production units to be better integrated into the distribution network by collecting consumption and production data from each user. Being able to predict household electricity consumption would help energy communities to allocate energy flows efficiently between participants.

Belgian company WeSmart offers management and support services for renewable energy communities. It is currently developing a recommendation tool to help users better manage their electricity consumption. The aim of this thesis is to study the prediction of household electricity consumption, with a view to advising members of energy communities on the best time to consume electricity.

Chapter 1

Context

This master thesis is conducted in collaboration with WeSmart, a startup that provides a digital platform to help in the creation and management of local energy communities.

1 Concepts

1.1 Energy communities

The climatic consequences linked to the exploitation of fossil fuels invite us to rethink our mode of energy production and turn to renewable energies. Many individuals and companies have already invested in production units such as photovoltaic panels, thus creating a decentralised electricity production network. However, without proper organisation it may seem complicated to efficiently manage the energy generated by a multitude of little producers. As we will see in this document, energy communities have the potential to solve this problem.

An energy community is a group of people, who collectively manage their production, consumption and eventually storage of energy. It can take many forms, going from a group of neighbours or business owners, to a larger-scale project that could includes an entire city. The infrastructure that provides energy to the community can also take different forms, however in the context of this work, we will primarily consider renewable energy production units such as photovoltaic panels and eventually wind turbines. We will also only consider communities with a limited geographic scope, therefore, for the rest of this document, energy communities will refer more specifically to local renewable energy communities.

The people who own decentralised production units are what we call prosumers. They can consume electricity from their own production and from the distribution network, but they can also inject the electricity they don't consume directly into the grid. Actually, there is no need to be part of an energy community to be a prosumer. The major difference for the members of such communities is that they can sell the surplus of their production to other members of the same community, generally at a lower price than the one charged by conventional suppliers.

This kind of decentralised energy systems could have an important role to play in the future of energy. The proximity between the producers and consumers reduces potential energy losses due to the transport of electricity. It also increases the resilience and flexibility of the grid by supporting large power plants. Such infrastructures can be subject to breakdowns and can sometimes encounter difficulties in meeting the needs of the population during consumption peaks that are difficult to predict [1][2].

The security of energy supply is an important issue for the European Union. It is still highly dependent on fossil fuels importations which represent more than 50% of of energy used in the territory. Investments in renewable energy have proven to be effective in reducing dependence on imported fossil fuels [3]. Energy communities appear as good candidates to integrate these investments and improve the energy security of the population. It can relocate energy production close to consumers, reduce dependence on fossil fuels and conventional suppliers, and provide affordable energy.

The reluctance of the public to build new infrastructures is an important obstacle to the development of renewable energies. An experimental survey conducted in Austria, Germany, Italy, and Switzerland revealed that the probability of a renewable energy transition project being accepted by citizens was three percent higher if it was a local energy community incorporating photovoltaic panels. Consequently, it could help foster public acceptance of renewable energy transition projects [4].

However, energy communities also face limitations and challenges. The integration of this new way of producing and consuming electricity is not without its difficulties. The profitability of energy communities for consumers but also for the electricity system depends on the political and economic framework applied. In this article [5], the author analyses regulatory schemes, in particular feed-in tariffs, and net metering. Feed-in tariffs oblige the electricity grid operator to buy all the energy produced by the prosumer. Net metering involves the installation of a bi-directional meter that allows the meter to run in reverse when energy generated

by the prosumer is sent to the grid. In this way, the consumer only pays for the net energy consumed (total energy consumed - total energy supplied to the grid). The analysis shows that in the case of feed-in tariffs, prosumers have strong interest in maximising the energy they inject into the network, but this is far less profitable for the electricity system in terms of cost and impact on the grid. Net metering, on the other hand, seems to be less negative for the electricity system, but it still seems to be correlated with a reduction in revenue in the energy sector.

The intermittency of the renewable energy sources can also be a limiting factor for energy communities relying on PV or wind turbines. Electricity production from these facilities depends entirely on weather conditions, and the peak production of PV often appears in the middle of the day, when the consumption is quite low. Improper planning of distributed generation systems may overload the distribution network and can cause instability of the voltage profile [6]. Proper integration of renewable energies into the grid requires upgrading the transmission infrastructures, which can represent a substantial investment [7]. Storing surplus energy could be a solution to the intermittent nature of these generating units. Domestic batteries, electric car batteries as well as pumped hydroelectricity energy storage systems could be used for this purpose [8], however, this only solves part of the problem. Storing energy is one thing, but determining when to store it, and when to send it to the grid, is essential for this solution to be effective.

1.2 Smart grids

Smarts grids are electrical networks managed by information and communication technologies. It is based on smart meters. These are devices able to measure precisely the amount of electricity that is consumed or fed into the grid. This precious data can be transmitted through internet connection and indicates in real-time the load of the network and eventual failures to the operators. It can also receive some data for monitoring and control purpose, offering a lot of possibilities for its users and new market actors such as aggregators[9].

Another great advantage of collecting data from smart meters is that it can be used to train machine learning models to forecast consumption and production tendencies. Being able to predict population power consumption is very useful for electricity producers, who can adapt their production accordingly and limit losses[10]. In the same way, forecasting prosumers electricity production could help to prevent instabilities of the voltage profile of transmission lines, manage efficiently energy storage systems, and advise energy community members on the best time to use electricity [11].

In Europe, smart meters are currently being rolled out. A study conducted by the European Commission's Directorate-General for Energy estimates that around 77% of European consumers will have an electricity smart meter by 2024. Thanks to a more efficient use of the grid, a better balance between electricity production and demand, a more effective integration of prosumers and a dynamic energy pricing, it is expected that smart-grids could reduce annual electricity bills of smart-meter owners by an average of 253€ and save energy between 5.42%-7.85%. However, it is important to note that these figures are indicative and may vary depending on the terms set by the states and the degree of consumer involvement [12].

1.3 Aggregators

Although energy communities can be created by citizens' initiative, this may require certain knowledge and skills that can be difficult to acquire without the necessary background. Aggregators are enablers for energy communities, they are third-party entities that help with creation and maintenance of communities. They have two main functions. Firstly, they manage the community's energy flows. The data collected from smart-meters can help them to balance the generation and load on the grid. This is often achieved by providing recommendations to community members such as time slots to optimise self-consumption or minimise costs. The second one is to supervise energy trades between the members of the same community on the one hand and between the community and the main supplier on the other hand [13].

The local energy market of energy communities follows a peer-to-peer design. Each member can be both a seller and a buyer. Peers can decide to manage all transactions themselves, using a trading platform for example. However, this strategy struggles to allocate energy flows efficiently. A central entity is desirable to maximise the social welfare of the community [14].

It is important to note that electricity exchanges between members of an energy community are often virtual. Members do not necessarily have to be interconnected by an additional electricity network. In practice, electricity produced by a prosumer, if not directly consumed by that prosumer, will take the shortest route to an ordinary consumer connected to the main grid, who may not be a member of the community. If a member of the energy community wants to buy this surplus production at an attractive price, he will actually consume the electricity from the traditional supplier, but pay the price set by the community's local market, whereas the average consumer will consume the surplus production, but pay the supplier's price. So it comes down to the same thing. There are also infrastructures called microgrids where electricity is exchanged directly between members, but for the purposes of this work we will only consider virtual exchanges.

2 Legal Ground

2.1 European directives

Following the adoption of the international treaty on climate change in 2015, the the European Parliament and the Council revised its directives on the promotion of renewable energy. The text of the new directives, which came into force in 2018, aims to establish the rules and principles to be followed by Member States in their support schemes for renewable energy. It states that the energy communities should be able to benefit from the assistance provided by the state in the same way as the larger participants, and also demonstrates the will to reduce administrative barriers and complexity to the creation of energy communities [15]. Following this, the European Commission issued another text in 2022 setting out recommendations for speeding up the permits-granting procedures for renewable energy projects and notably energy communities as shown by the 8^{th} and 9^{th} recommendations of the text:

- *Member States should stimulate the participation of citizens, including from low and middle-income households, and energy communities in renewable energy projects, as well as take measures to encourage passing the benefits of the energy transition on to local communities thus enhancing public acceptance and engagement.*
- *Member States should implement simplified permit-granting procedures for renewable energy communities, including for the connection of community-owned plants to the grid and reduce to a minimum production licensing procedures and requirements, including for renewables self-consumers [16].*

The "Clean Energy for All Europeans" package adopted in 2019 aims to provide Member States with a direction and legal framework for the energy transition. Among the eight new laws, the text on the design of the electricity market demonstrate a clear desire to make citizens more responsible for their energy consumption, and recognise energy communities as an energy-efficient and cost-effective way of meeting citizens' needs while delivering on their commitment to the European green deal [17].

2.2 Belgian legislation

As a member of the European Union, Belgium is committed to implementing the directives and recommendations it draws up. However, since the regionalisation of Belgium in 1980, energy legislation has been the responsibility of the regions.

The Brussels-Capital Region was the first to approve in March 2022 a project aimed at defining a legal framework, authorising and facilitating the creation of energy communities. As regards the Walloon region, the decree presented on 5 May 2022 was approved in March 2023. And the decree has also been accepted by the flemish region. This text clearly defines the legal status of renewable energy communities, and establishes its rights.

Art. 4.2° *"renewable energy community" means a legal entity: which is based on open and voluntary participation and is autonomous; whose shareholders or members are : natural persons, local authorities as defined by the Government, including municipalities, small or medium-sized enterprises whose main commercial or professional activity is not participation in one or more energy communities; which is effectively controlled by the participants located in the vicinity of the production facilities which it owns or over which it has a right of use; whose main objective is to provide environmental, economic or social benefits to its participants or to the local areas in which it operates, rather than to generate financial profits.*

Art. 35undecies § 1. *Within the meaning of the matters governed by this Decree, an energy community has the right to carry out the following activities: 1° generating electricity ; 2° supplying electricity ; 3° self-consumption of the electricity produced by its installation(s), after storage where appropriate, at the location of its production installation(s); 4° share between its members the electricity produced, either by the installations it owns, or by the installations over which it has a right of use likely to give it the status of producer, or by self-generation installations owned by its members and fed into the network; 5° practise aggregation ; 6° participating in flexibility services ; 7° storing all or part of the electricity it receives from the grid or which it has produced itself; 8° provide recharging services for electric vehicles; 9° provide energy efficiency services or other energy services; 10° sell the electricity it produces that is not self-consumed and not shared in accordance with 4° and, in the case of electricity from renewable energy sources, where applicable by means of a renewable electricity purchase contract or peer-to-peer trading.*[18]

3 Current energy context

By 2021, energy prices had already risen as a result of the economic upturn following the end of the COVID19 pandemic. Russia's invasion of Ukraine in 2022 prompted the United States and Europe to impose economic sanctions and reduce imports of Russian fossil fuels, which greatly accentuated the rise in prices. Although these measures mainly concern natural gas imports, electricity prices have also been heavily impacted. Even today, on European markets, the price of

electricity is determined by the marginal production cost of the last kWh produced. And this marginal production cost is often the one of gas-fired power stations. This is why the price of electricity is in reality strongly linked to the price of gas [19].

Since the energy communities have their own electricity market, prices are less dependent on the wholesale markets and are therefore less sensitive to economic disruptions at global and European level. Energy communities can therefore contribute locally to the energy security of the population.

4 WeSmart

WeSmart is a Belgian company based in Brussels that sets up and manages energy communities. It acts as an aggregator and is involved in several key stages in the creation and maintenance of an energy community:

- Identifying participants: WeSmart is responsible for identifying and contacting potential participants. This also makes it possible to identify the production units available and to determine the different profiles present in the community.
- Technical and economic feasibility: Using the data collected on the participants, a study is carried out to determine whether the project is viable, and to estimate the benefits that could be enjoyed by the members of the community.
- Creation and launch: WeSmart takes care of the administrative aspects and makes its dedicated platform available to participants.
- Management and optimisation: Billing, and an optimisation service to help participants consume at the best time.
- Awareness: Making participants aware of their role within the community is important because the effectiveness of an energy community also depends on the degree of involvement of the participants.

In addition to the services related to establishing an energy community, WeSmart provides a platform that facilitates the management of your own energy community and participation in energy projects. Through the utilisation of data collected from each participant's smart meter, WeSmart delivers a monitoring service that empowers consumers to visualise their cost savings achieved through community collaboration. Furthermore, the community manager has the capability to oversee key metrics, including electrical self-sufficiency, KW/h pricing, average production capacity, and daily energy consumption.

4.1 What are the needs ?

WeSmart is currently developing an optimisation and recommendation service for its platform. The aim is to provide users with personalised advice to maximise the community's self-sufficiency and therefore the benefit(s) for each member. By optimising consumption of the electricity produced by the community, each member can make the most of clean and affordable energy. It also limits the negative impact of decentralised production units on the distribution network by reducing congestion during production peaks.

To achieve this, it may be useful to be able to forecast the community's electricity production and each member's consumption in the near future with sufficient accuracy. Once this information is known, it is possible to build a personalised and precise recommendation tool that allows the consumption curve to be shifted off-peak hours. Ideally, the consumption curve should approximately follow the community's power generation curve without exceeding it as shown in the figure below 1.1. Once again, these recommendations only make sense if they are actually applied by the members of the community. Another challenge is to determine the number of recommendations to send out to get the desired number of people to actually apply them. This problem will not be discussed in this work, but could be the subject of a future improvement.

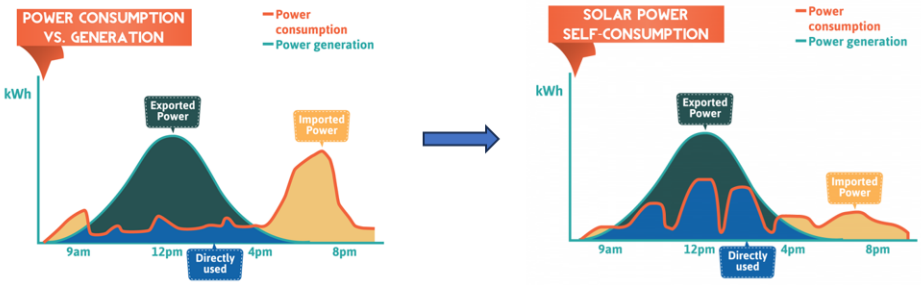


Figure 1.1: Shifting consumption curve [20]

A recommendation algorithm could analyse the predicted patterns of electricity demand and production, and determine the best time frames for consumption. Users could then be notified of the most opportune moment to carry out the most electricity-intensive tasks, such as running the tumble dryer, charging the electric car, or activating electric heating systems. In this way, we can balance the flow of energy produced and consumed by the community, so that members can make the most of clean, inexpensive energy. Such a tool could also provide personalised recommendations based on individual requests and needs. Its integration into smart homes would enable certain tasks to be automated during working hours. The potential applications are numerous and welcome at a time when energy prices are tending to rise.

The annual solar energy output of a photovoltaic system can be estimated with a simple formula.

$$E = A * r * H * PR$$

Where

- E is the energy produced in Kilowatt-hours.
- A is the total area of the production units.
- r is the yield of the panel as a percentage; this is specific to each model of panel and is supplied by the manufacturer.
- H is the solar radiation on tilted panels.
- PR is the performance ratio, which lies between 0.5 and 0.9 and is used to estimate losses due to external parameters such as temperature, shade and dust, as well as losses caused by the transformation of direct current into alternating current [21].

A more precise estimate, on a daily scale for example, can be obtained using simulation tools, or by implementing prediction models based on meteorological data.

Household electricity consumption can be more complex to predict, given the large number of factors that can come into play. The aim of this thesis will be to analyse the ability of different prediction algorithms to forecast future electricity consumption patterns based on historical data.

Chapter 2

Methodology

1 Problem definition

The aim of this thesis is to study the feasibility of a tool for predicting the electricity consumption of members of energy communities. For my research, WeSmart provides me with eight consumption profiles from an energy community in operation in Tournai, Belgium. It comes in the form of Excel files, each containing the quarter-hourly electricity meter readings of an anonymous consumer covering at least one year between 2020 and 2021. To this we add data on the weather conditions in the same geographical area as the community, on the same dates.

For now, WeSmart does not have access to real-time electricity meter readings. It would therefore be pointless to make predictions in a too short timeframe, as these predictions would quickly become obsolete because they are out of date. We therefore consider that the tool must be capable of predicting a week's worth of future data.

This tool would be part of the WeSmart recommendations service. By comparing the predicted consumption and production values, the algorithm is supposed to suggest users with the best time frames to perform intensive electricity-consuming tasks. As these tasks generally require a time window of one hour or more, we consider that a granularity of one hour for the prediction tool is sufficient.

The requirements are the following : this tool must be capable of producing a solution for any consumption profile, provided that the data format is respected. It must be as accurate as possible, and reliable over time.

2 Project plan

The first step in this project is to search the scientific literature for existing solutions to similar problems. The IEEEXplore digital library contains numerous articles on computer science and electrical and electronic engineering. The main articles analysed for this project come from this database.

The next step is to prepare the data sets. First of all, we concatenate the consumption data with the meteorological data. The most practical way to do this is to create a csv file, which stands for "Coma separated values". This is the type of file most commonly used for data analysis, and they are easily interpreted by Python libraries such as Pandas. The next step is to check the integrity of these data sets and compensate for any missing data and noise that may have been introduced by the measuring equipment. The data is then cleaned of outliers. Once the files are clear, we will try to extract as much useful information as possible for our problem. We will then filter it to retain the most relevant information.

The next stage of the project consists of implementing some of the solutions analysed in the literature review. The aim is to find the one that best satisfies our needs by analysing the advantages and disadvantages of each and analysing their performance on our data.

To ensure that the solution is reliable over time, we will create an algorithm capable of detecting when the prediction deviates too far from the actual values, and rework the solution accordingly to restore a suitable level of accuracy.

Chapter 3

State of the art

Now that the problem has been defined, the next step is to gather the knowledge needed to implement a concrete solution. The main objective is to explain people's electricity consumption using their consumption history and past weather data for the area in which they live.

Intuitively, when it comes to predicting continuous values, we think of regression techniques, unlike classification, which is used to predict the class to which the data belongs. Then, as we are trying to predict a result on unseen data using a data history, we move towards supervised regression techniques, as opposed to unsupervised regression, which is used to find patterns on unlabelled data.

We are given a set of input variables, also referred to as features, denoted X, and an output variable, denoted Y characterising the electricity consumption in Wh. The model is trained on these data in order to find the optimal mapping relationship between the input variables and the output variable. This modelling, denoted $f(X)$, can take various forms, but the objective remains to minimise the error between the actual values Y and the modelled values Y'.

$$find\ f(X) = Y'\ s.t\ error(Y, Y')\ is\ minimum$$

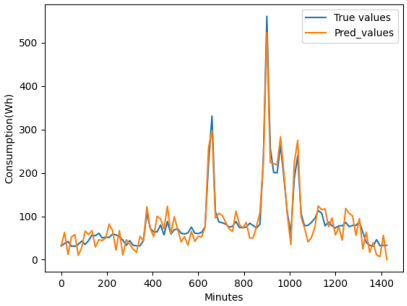
The measurement to assess the error between the actual values and the estimated ones can also take different forms. This will mainly depend on the problem. Training a machine learning model is therefore an optimisation problem in which we seek to minimise an error function.

1 Electricity consumption forecasting

With the development of smart grids, energy prediction models are raising interest. Indeed, there are many articles in the scientific literature dealing with the prediction of energy production from renewable [22]–[24]. There are also many articles dealing with the prediction of power load, which is the portion of electricity used for heating systems and electric motors [25]–[27]. A few articles describe models for predicting aggregate electricity consumption over a given geographical area and over a period of varying length [28]–[30]. On the other hand, there are relatively few cases on electricity consumption by individual households.

Forecasting a household's electricity consumption is a complex time-series problem. Firstly, because like many real machine learning applications, the information gathered by the sensors may contain noise, redundant data, outliers, and some values may also be missing. Beyond that, a household's electricity consumption is highly volatile and difficult to anticipate. Predicting power consumption at a national level, for example, already seems a more convenient task. On a large scale, variations tend to be less abrupt, and atypical behaviour is strongly attenuated by the general trend. The prediction of power load, although complex, also seems comparatively straightforward. There is already a greater chance of observing similar trends over time and correlations with certain meteorological data as it only concerns a portion of electricity use. For example, we can expect a certain correlation between the evolution of the power drawn by electric heating systems and the outside temperature.

The time interval used for the prediction can also be a determining factor in the complexity of the problem. Indeed, forecasting a citizen's electricity consumption every quarter-hour over the course of a day will produce an error greater than if we consider the average consumption hour by hour or over 12-hour periods.



Prediction interval	MAPE
15 minutes	0.29
1 hour	0.12
12 hour	0.015

2 Supervised regression models review

It is difficult to determine in advance which model will be the most appropriate for solving a given problem. There are many models capable of solving supervised regression problems, each with a different way of working. The complexity of the problem, the volume of data, and the computing resources available are factors to be taken into account when choosing a model. Neural networks are typically used on large volumes of data, and require greater resources than linear regression. However, we cannot state in advance that one model will produce a more accurate solution than another. It is therefore advisable to test different solutions to find the most suitable one. In the scientific literature, several regression models are regularly used to deal with energy forecasting problems. This section will be dedicated to the review of some of these models.

2.1 K-Nearest Neighbours

The K-nearest neighbours algorithm is quite simple. The idea is to determine the value or the class of an unseen data point from a number of nearest data samples. The K parameter defines the number of neighbours used for the computation, and can be tuned according to the problem. The distance calculation can be performed by any distance measurement function, the default choice being the Euclidean distance :

$$d_{x_1,x_2} = \sqrt{\sum_{i=1}^{D} (x_{1i} - x_{2i})^2}$$

Where D is the dimension of the problem determined by the number of features in the dataset.

For regression, we simply take the average of the K nearest points. We can choose to involve the points uniformly, or to assign them a weight that is inversely proportional to their distance. In this way, the closest points among the K samples will be given a higher weight and will have a greater impact on calculating the average compared to points with a lower weight.

Different algorithms exist for calculating nearest neighbours. Brute force, as its name suggests, simply calculates the distance between the point whose value is to be determined and all the other points in the training dataset. The complexity of this algorithm is $O(D * N^2)$ where N is the number of data samples, and D is the number of dimensions. However, this solution quickly becomes inefficient when N is large.

To overcome this problem, a tree-based data structure called *K-D Tree* was proposed by Jon Louis Bentley in 1975. This data structure efficiently encodes the distances between the samples of the training set. It limits the number of distance calculations thanks to a simple principle. If point A is very distant from point B and point B is very close to point C, then by transitivity, point A is very distant from point C. The complexity of this algorithm is given by $O(D * Nlog(N))$. In practice, this algorithm is very effective for problems with dimensions not exceeding 20. Beyond that, it can become inefficient.

The *Ball Tree* is another data structure based on a binary tree in which each node corresponds to a ball of dimension D. Each ball is defined by a centre C and a radius r such that each point within the node lies within the sphere formed by the parameters C and r. The calculation of the distance between an unseen data point and the centre C of a node is sufficient to determining the boundaries of the distance between this point and all the samples included in this node.

This algorithm has a complexity of $O(Dlog(N))$ and can handle high-dimensional problems more efficiently than the *K-D Tree*. However, the construction of the tree is comparatively more costly. The choice of the best Nearest neighbours algorithm will then depend on the nature of the problem.[31]

2.2 Support Vector Machine

A standard linear regression algorithm seeks to draw a line denoted $f(x)$ through a cloud of points x_i such that the sum of the distances between $f(x_i)$ and y_i is minimal. For a multivariate regression with n features, the function is written as :

$$f(x) = w_0 + w_1 * x_1 + w_2 * x_2 + ... + w_n * x_n$$

The vector of input variables and the vector of weights are respectively denoted by :

$$\vec{x} = [1, x_1, x_2, \ldots, x_n]$$
$$\vec{w} = [w_0, w_1, w_2, \ldots, w_n]$$

The goal is to find $\vec{w}$ such that :

$$Min \sum_{i=1}^{m} (y_i - \vec{x}_i.\vec{w})^2$$

Where m is the size of the dataset.

The *Support vector machine* algorithm is widely used in machine learning for classification tasks. However, it can also be quite effective for regression. It particularly differs from classical linear regression by introducing an acceptable margin of error denoted ϵ. The objective is to minimise the L2-norm of the vector of coefficients under the error constraint :

$$Min\ \frac{1}{2}\ ||\vec{w}||_2$$

Such that

$$|y_i - \vec{x}_i.\vec{w}| \leq \epsilon \quad \forall i \in m$$

The epsilon parameter can therefore be tuned to obtain the desired accuracy. However, it is not always possible to find a solution such that all the data points x are within the acceptable error margin. For all values, a slack variable denoted ξ which corresponds to the deviation between the value and the ϵ margin is introduced. These variables make it possible to accept an additional margin of error, but the objective is still to minimise them. The optimisation objective becomes :

$$Min\ \frac{1}{2}\ ||\vec{w}||_2 + C\sum_{i=1}^{n} |\xi_i|$$

Such that

$$|y_i - \vec{x}_i.\vec{w}| \leq \epsilon + |\xi_i| \quad \forall i \in m$$

The hyper-parameter C defines the penalty applied to samples outside the margin defined by epsilon. It is also called the regularisation parameter. Intuitively, for large values of C, the model will try harder to adapt to the training data, which can lead to overfitting. A grid search can be performed on a specific problem to find the optimal value of C. [32], [33]

Some regression problems cannot be efficiently explained by a linear model. It is possible to extend the SVM regression model for non-linear functions by introducing a kernel.

A kernel is a mathematical function that transforms a low-dimensional input space into a higher-dimensional space. For SVM the 'Radial Basis Function' kernel is the default choice. This kernel can be expressed as the following function:

$$K(X_1, X_2) = exp(-\frac{||X_1 - X_2||^2}{2\sigma^2})$$

It computes the similarity between two distinct data points. A value close to 1 means that the distance separating the two points is very small, and so their similarity is high. Conversely, when the value is close to 0, the distance separating the two points is high, and their similarity is low.

The sigma σ parameter is used to determine the distance at which points are considered to be similar or dissimilar. A large value for sigma means that this distance may be relatively large, in which case the model will be less effective at capturing the complexity of the data, and will behave rather like a linear model. A small value for sigma, on the other hand, means that the distance must be very small for two points to be considered similar. In this second case, the model will tend to overfit the data.[34]

2.3 Random Forest

The *Random forest* algorithm belongs to the family of ensemble methods. The aim is to use the prediction of several base models built from a given algorithm to produce the result of the final model. This serves to improve the ability of the Machine Learning method to generalise the prediction and therefore reduce overfitting. Ensemble methods can be divided into two sub-families:

- Bagging methods: Several models are built and trained on the problem independently. The predictions of these models are then averaged. This solution generally offers a better result than each of the basic estimators.
- Boosting methods: Estimators called weak learners are added sequentially to the main model. A weight is assigned to the predictions of each weak learner so that poor predictions receive a high weight. The next weak learner will then focus on the poorly learned training data in an attempt to compensate for this shortcoming.

Random forest is part of the bagging ensemble family. It is based on *Decision Trees* which are supervised learning methods for classification and regression. Their principle is to create a binary tree using simple decision rules based on the features of the dataset. The leaves of the tree contain the predicted values of a target variable. A decision tree can be thought of as a sequence of "if", "then" and "else" conditions. The depth of the tree influences the complexity of the decision rules and therefore the adaptation of the model. In other words, the deeper the tree, the more complex the decision rules become, allowing the model to fit the data better [36].

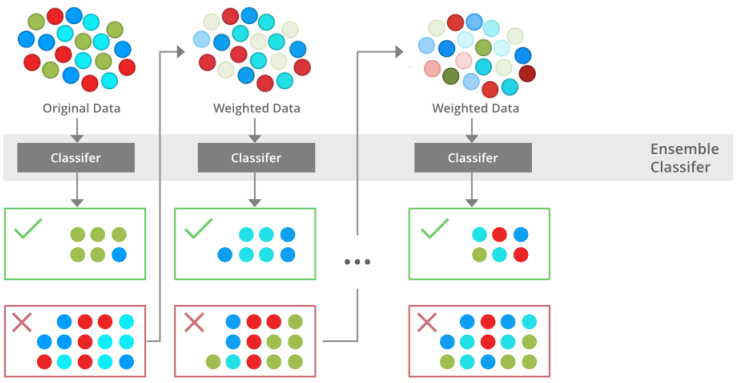


Figure 3.1: Illustration of the boosting method [35].

Random forest uses the perturb-and-combine principle. This means that the basic decision trees are constructed by introducing randomness. The final prediction is then calculated as the average of the predictions from the multiple decision trees. The randomness introduced in this algorithm comes from two sources :

- Firstly, each basic decision tree is built on the basis of a bootstrap sample. This is a resampling technique that consists of repeatedly drawing samples from the dataset with replacement (duplicates allowed).
- In addition, when building decision trees, the best split can be found from all the features in the training dataset, or from a randomly selected subset of these features.

The aim of introducing randomness into the model is to reduce its variance. The variance of a Machine Learning model defines how the model varies depending on the training dataset. If the predictions differ greatly from one training set to another, then the variance is high, which may also indicate overfitting. As tree decisions generally have a high variance, randomising and averaging the predictions significantly reduces the variance of the final model. However, this can be accompanied by an increase in bias. Bias error is defined as the inability of a model to correctly capture the relationship between the features and the variable of interest. We therefore need to find the right trade-off between bias and variance to obtain the most accurate predictions possible on unseen data.

It is possible to tune the number of basic estimators used to build the final model. The performance of the model tends to increase with the number of estimators up to a certain threshold, which is specific to each problem. The execution time also increases with the number of estimators, hence the utility of identifying this

threshold. A second important parameter to be tuned is the maximum number of features randomly selected to find the best split in the decision trees. A small number will result in a large reduction in variance, but will tend to increase the bias error. Again, the optimal value for this parameter will depend on the nature of the problem.[37]

2.4 Gradient Boosted Trees (XGBoost)

XGBoost is a library that provides a framework for machine learning models using the gradient boosting principle. In our case, we are using the gradient tree bosting method. Like random forest, this model belongs to the ensemble learning method family and uses decision trees as weak learners. However, it belongs to the boosting methods sub-family. This algorithm works as follows:

The objective is to train a model $F(x) = y'$ minimising a loss function $L(y, y')$. Let's consider the Mean Squared Error as a loss function.

$$L(y, y') = \frac{1}{n} \sum_{i=1}^{n} (y_i - y'_i)^2$$

The model is initialized with a simple function that returns the mean of the y values for a given training set of size n.

$$F_0(x) = \frac{1}{n} \sum_{i=1}^{n} y_i$$

The residual error of each point for a given test set of size m can be calculated as follows:

$$r_i = y_i - F_0(x_i) \quad \forall i \in m$$

We then train a decision tree $t_1(x)$ on this residual data. The aim of this new model is not to predict the values of y, but to reduce the error of the previous model. The 'boosted' model is given by :

$$F_1(x) = F_0(x) + t_1(x)$$

We can then calculate the residual error r_i for each point of the test set for this new model, then introduce a new decision tree $t_2(x)$ to reduce these errors, and build the boosted model $F_2(x) = F_1(x) + t_2(x)$ and so on.

A learning rate α is introduced to regularise weak learners by shrinkage. This learning rate is used to control the convergence of the model. In practice, a small learning rate significantly improves the generalisation capacity of the model. The generalised model takes the following form:

$$F_k(x) = F_{k-1}(x) + \alpha * t_k(x)$$

Where k is the number of estimators, which also defines the number of iterations. A model with a small learning rate (<0.1) will require a larger number of estimators to converge. The larger k, the better the model will fit the training data.

2.5 Autoregressive integrated moving average (ARIMA)

ARIMA is a statistical model used on time series data. A dataset is considered to be a time series if it is ordered chronologically and data are separated by fixed time interval. It is auto regressive, which means that the prediction of the values of the variable of interest is calculated from the past values of this same variable. This model is therefore used when it is assumed that future values are very likely to resemble past values. The lag order denoted p defines the number of previous time-steps considered for estimating the current value. The equation of the auto-regressive block of the statistical model is noted :

$$Y_t = \beta_1 + \phi_1 * Y_{t-1} + \phi_2 * Y_{t-2} + ... + \phi_p * Y_{t-p}$$

Where Y_t is the value we are trying to explain from the constant β_1 and the p previous values $(Y_{t-1}, Y_{t-2}, ..., Y_{t-p})$ weighted by $(\phi_1, \phi_2, ..., \phi_p)$. The weights are computed statistically, and represent the respective degrees of correlation between the weighted value and the current value.

Moving average means that prediction errors from previous periods are exploited to improve the current prediction. The number of errors considered is given by q, which defines the size of the sliding window. The equation is given by :

$$Y_t = \beta_2 + \omega_1 * \xi_{t-1} + \omega_2 * \xi_{t-2} + ... + \omega_q * \xi_{t-q} + \xi_t$$

Where $(\xi_{t-1}, \xi_{t-2}, ..., \xi_{t-q})$ are the errors of the q previous predictions weighted by $(\omega_1, \omega_2, ..., \omega_q)$, and β_2 is a constant.

Combining these two equations gives the ARMA statistical model:

$$\begin{gathered} Y_t = (\beta_1 + \beta_2) + \\ (\phi_1 * Y_{t-1} + \phi_2 * Y_{t-2} + ... + \phi_p * Y_{t-p}) + \\ (\omega_1 * \xi_{t-1} + \omega_2 * \xi_{t-2} + ... + \omega_q * \xi_{t-q} + \xi_t) \end{gathered}$$

This model only operates on stationary data. A time series is said to be weakly stationary if it meets the following conditions:

- The expectation is constant over time, in other words, the average of the values over a given period is approximately equal to the average of the values over the whole series.
- The variance is constant over time.
- There is no correlation between the variable of interest and time.

It is possible to make data stationary by transforming each data point Y_t by the difference between the next point $Y_t + 1$ and this point Y_t. Sometimes this operation, called differencing, has to be performed several times for the data to be effectively stationary. The number of differencing operation required to make the data stationary defines the third parameter of the model, which is called the differencing order.

$$diff\ order = 1 : Z_t = Y_{t+1} - Y_t$$
$$diff\ order = 2 : Q_t = Z_{t+1} - Z_t$$
$$...$$

The Arima model is called integrated because it predicts the difference between the time steps rather than the values of the series itself.[38]

To sum up, the ARIMA statistical model has three parameters:

- Lag order (p): number of previous observations considered relevant for explaining the current value.
- Differencing order (d): number of times the data had to be differenced for them to become stationary.
- Size of the sliding window (q): defines the number of errors in previous predictions that are considered relevant for improving the current prediction.

2.6 Multi-Layer Perceptron

The *Multilayer perceptron* belongs to the family of deep learning algorithms, a branch of machine learning that uses artificial neural networks to process complex data. It was inspired by biological neurons, with the idea of creating a model capable of learning the characteristics of the data by itself.

To understand how this model works, it is essential to explain how an artificial neuron also called a *perceptron* operates. This is the unit of artificial neural networks. It consists of a linear function and an activation function.

Considered the perceptron k, the inputs $(x_1, x_2, ..., x_m)$ are the data provided to the model weighted by the vector k $(w_{k1}, w_{k2}, ..., w_{km})$. The input x_0 is generally fixed at $+1$ and the weight w_k0 represents the bias of the model and can be seen as the intercept of the linear function. The sum of these weighted inputs and the bias is then fed to an activation function to produce the output y_k [39].

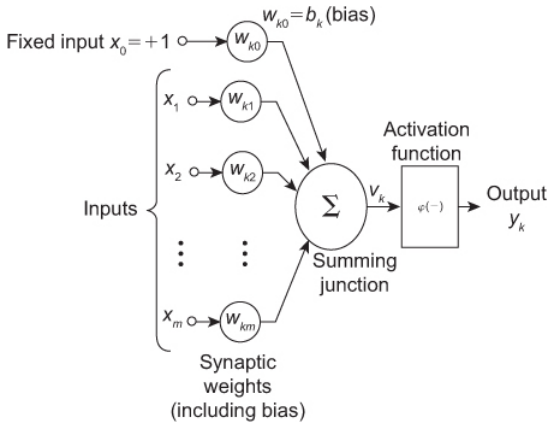


Figure 3.2: Illustration of a perceptron

The *Multilayer perceptron*, as its name suggests, has several layers. It is structured as follows:

- The input layer contains the problem's data. It can be represented by the vector $(x_1, x_2, ..., x_n)$ containing n values. It simply feeds them to the next layer.
- A hidden layer made up of m artificial neurons. Each neuron has its own weight vector of size n and its bias. Each neuron computes the sum of the values received from the previous layer, weighted by its weight vector, plus its bias. It then passes the result to its activation function and the output is fed to the next layer. The m outputs of the hidden layer with a sigmoid activation function are given by the following formula :

$$\forall j \in m : y_j = \sigma((\sum_{i=1}^{n} w_{ij} * x_i) + b_j)$$

- Other optional hidden layers similar to the previous one can be added to the model, each with a certain number of neurons.

- The output layer is made up of one or more activation functions, depending on the number of outputs needed for the problem. It receives the outputs from the previous layer, and computes the final output(s) [40].

There are several activation functions with different objectives. A neural network is often made up of several layers. The input layer does not use an activation function. The hidden layers typically use non-linear and differentiable activation functions.

Non-linearity is necessary for learning complex relations and for capturing the complexity of training data. Linear activation functions would render hidden layers useless, as the composition of linear functions necessarily results in a linear function, which is not suitable for complex problems [41], [42].

Differentiability is necessary to be able to use the backpropagation technique and adjust the weight vectors when training the model. The activation functions most commonly used in hidden layers are :

- *sigmoid* : $\sigma(x) = \frac{1}{1+e^{-x}}$: maps any input to a value between 0 and 1.
- *Hyperbolic tangent activation function* : $tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$: shape very similar to *sigmoid*, but it maps any input to a value between -1 and 1.
- *Rectified Linear Unit*: $ReLu(x) = max(0, x)$: simply returns the input if it is positive, 0 otherwise.

The choice of activation function for hidden layers depends on the type of neural network. However, *ReLu* is widely used and is preferred in neural networks with many layers, because the *sigmoid* and *tanh* functions tend to saturate when the value passed to the function is large. This means that for highly positive or negative values, the output value remains about the same. They are therefore less sensitive to variations in the weight vectors and reduce the ability to converge, which is not the case with *ReLu*.

The output layer uses different activation functions depending on the type of problem. For a classification problem, the activation function can take different forms, for example :

- The sigmoid returns a probability. It can be used for binary classification: a certain input will be classified with $label_1$ if the probability is greater than 0.5, $label_2$ otherwise.

- Softmax : $\sigma(x_j) = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}} \quad \forall j \in (1, ..., K)$: is used for multiclass classification. It defines a probability distribution for K mutually exclusive classes. A certain input will be classified with the label k which has the highest probability among the K labels. This type of classification involves the use of several neuron in the output layer (one per class).

In the case of regression, the activation function must provide a continuous value. In this case, we simply use the linear function f(x) = x, which simply returns what it receives as input.

The main objective of the model remains to find the weight vectors that minimise the error between the predicted values and the actual values of the variable(s) of interest. This weight vector learning mechanism is achieved by the Backpropagation algorithm. For MLP, the objective is to minimise the mean squared error.

In the first iteration, the weight vectors of the artificial neurons are randomly generated and the biases are set to zero. Each time the data makes a forward pass from the input layer to the output layer, the gradient of each parameter is calculated by deriving the loss function with respect to this parameter, weighted by the negative learning rate α. The weight and bias gradients are given by :

$$\triangle w_j = -\alpha \ \frac{\partial L}{\partial w}$$

$$\triangle b_j = -\alpha \ \frac{\partial L}{\partial b}$$

$$L = Mean\ squared\ error = \frac{1}{n} \sum_{i=1}^{n} (y_i - y_i')^2$$

Once the gradients have been calculated, the weights and biases of each neuron are updated:

$$w_{j+1} = w_j - \triangle w_j$$

$$b_{j+1} = b_j - \triangle b_j$$

The forward pass and the backpropagation phase are executed iteratively until the weights have all converged.

2.7 Deep hybrid learning

In the scientific literature, deep learning algorithms have been proven to be highly effective for solving problems on real data. They are particularly appreciated for their ability to process unstructured data, and their ability to extract relevant features from the data without needing to perform feature engineering beforehand. On the other hand, they often require a large amount of data to obtain a satisfying result, and are fairly greedy in terms of hardware resources and computing time.

Classical machine learning algorithms are very effective for solving problems on structured data where the characteristics have been extracted beforehand. They generally require less data and are less expensive in terms of computation.

Deep hybrid learning therefore consists of merging a deep learning algorithm with a machine learning model to produce a solution that is expected more accurate and less costly in terms of computation. We can also merge two deep learning models with different structures to try to overcome the shortcomings of each of the two base models. [43]

Convolutional neural network - Long short term memory

The deep hybrid learning algorithm reviewed is a CNN-LSTM that has been proposed for a *Short-Term Individual Household Load Forecasting* problem [44]. The idea is to use the properties of the convolutional neural network to extract features of interest and, at the same time, filter out the noise in the input data. On the other hand, Long-Short-Term memory is used to capture patterns in temporally correlated data. It is useful for finding short and long-term dependencies.

Initially, CNN is used for image recognition. Images are encoded in the form of matrices. But it is possible to process any data, as long as it can be represented in matrix form. This is the first step in implementing this hybrid model. All the features of the problem are encoded on a single vector.

The part of the convolutional neural network that extracts features is designed as follows :

- Convolutional layer: In this layer, a filter that we call a kernel is used to extract specific patterns from the input matrix. This kernel takes the form of a small matrix (smaller than the input matrix). This kernel is slided across the input matrix from left to right and from top to bottom. We successively calculate the scalar product between the kernel and the selected portion of the matrix. This operation produces a new matrix called the feature map. Each

entry in this new matrix is then passed to a non-linear activation function, typically ReLu [45].

- Pooling layer: the objective of this layer is to reduce the dimension of the feature map in order to reduce the computation load. Once again, a filter is applied across the matrix received as input. With a MaxPooling layer, only the maximum value of the portion of the matrix selected by the filter is retained. With an AveragePooling layer, the average of the values included in this portion is retained. A new matrix is thus created, the size of which depends on the size of the filter [46]. Here the authors of the article have chosen to use a Maxpooling layer

The backpropagation phase updates the kernels rather than the weights and biases, as it is the case for the Multiayer perceptron for example. After the feature extraction layers, there is a dropout layer. This is used to reduce the model's over-fitting by randomly selecting a certain number of neurons which will be deactivated during the feedforward phase. The next step is the sequence learning, which is performed by LSTM layers.

Long short term memory is a particular case of Recurrent neural network. The idea of the RNN is to include the output of the previous iteration in the computation of the current output in the neurons of the hidden layers.

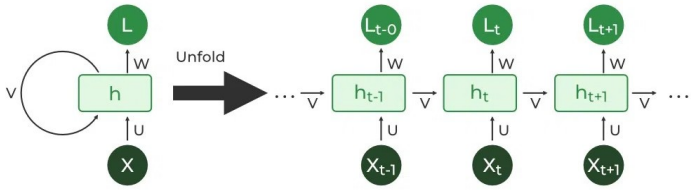


Figure 3.3: Illustration of a neuron in a RNN

This model has proved to be very effective for certain problems, particularly timeseries problems. However, it can suffer from exploding gradient and vanishing gradient problems. These problems appear during the backpropagation phases of the training. The vanishing gradient prevents the model from converging because the gradient becomes negligible over the iterations. Conversely, a gradient that increases very sharply can lead to divergence. The use of the ReLu activation function can reduces the risk of vanishing gradient, however, it is not applicable to all types of problem, and it does not eliminate the risk. Long Short-term memory is used to overcome these issues [47].

LSTM greatly reduces the risk of vanishing gradient, but does not prevent the problem of exploding gradient. The idea is to provide the RNN with a short-term memory that can last for many iterations. This is provided by "gates" that manage the influx of data into and out of the cells. The hidden layer is structured as follows:

Let x_t be the current input vector, h_{t-1} be the output vector from the previous time step (previous hidden state), W, U and b are respectively the weight matrices and the bias which are specific to each gate and cell, and which must be learned during training of the model.

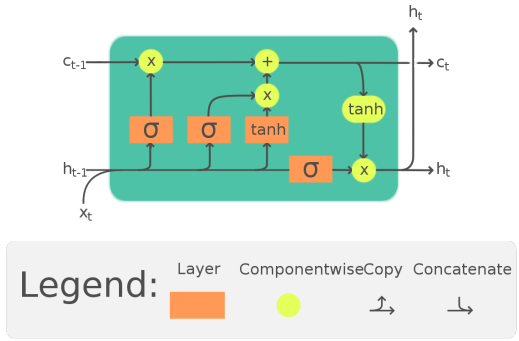


Figure 3.4: Illsutration of a single cell inside a LSTM hidden layer [48].

The Forget gate quantifies the information from the previous time-step that is kept. This function gives a vector of results between 0 and 1, with 0 = forget everything, and 1 = keep everything.

$$f_t = \sigma(W_f * x_t + U_f * h_{t-1} + b_f)$$

The Input gate quantifies the importance of the current information.

$$i_t = \sigma(W_i * x_t + U_i * h_{t-1} + b_i)$$

The current information to be quantified by the input gate is given by :

$$n_t = tanh(W_c * x_c + U_c * h_{t-1} + b_c)$$

Once this information has been quantified, it is added to what is known as the cell state. It is this state which contains the information retained from previous time-steps. The update of the cell state is given by the quantity of old information to keep plus the quantity of new information to add.

$$c_t = f_t \odot c_{t-1} + i_t \odot n_t$$

Here we use the Hadamard product rather than the classic matrix multiplication. It takes two matrix of the same dimension and simply returns the element-by-element product of the corresponding components of the two matrices. The output gate also return a vector with values between 0 and 1:

$$o_t = \sigma(W_o * x_o + U_o * h_{t-1} + b_o)$$

Finally, the output vector of the layer is given by:

$$h_t = o_t \odot tanh(c_t)$$

The output vector or hidden state, is therefore computed from the current output o_t and the current state of c_t, which contains information from previous time-steps.

After the feature extraction block based on a CNN and the sequential learning block based on an LSTM, there is a second dropout layer and the fully-connected layer. This block receives the output of the previous layer and computes the final output.

The first layer of this last block is used to flatten the output of the previous layer. It receives a matrix and produces a large vector. In this case, the fully-connected layer already receives a vector, so the flattening layer is not necessary. This vector is therefore sent to the first layer of the fully-connected layer, which simply passes the data to the output layer.

The number of neurons m in the output layer defines the number of time-steps predicted. For example, three neurons can be used to predict electricity consumption over the next three hours. For each neuron, the output layer computes the weighted sum of the received vector plus the bias:

$$\forall j \in m \quad y_j = (\sum_{i=1}^{n} w_{ij} * x_i) + b_j$$

This result is then passed through an activation function, which is simply the linear function f(x) = x, as we are in a regression case.

To analyse the performance of this model, the authors used the Mean absolute percentage error metric. This loss function is commonly used because it is intuitive and easy to interpret in terms of relative error, as it provides a percentage result. It is calculated as follows:

$$MAPE = \frac{100}{n} \sum_{i=1}^{n} \left| \frac{y_i - y_i'}{y_i} \right|$$

The dataset contains electricity consumption data for 69 households in Australia, all of which have an electric boiler. On average, this model shows a $MAPE = 40.38\%$. This may seem like a large error, but in fact these results are better than most of the state-of-the-art results. Indeed, an LSTM model has been proposed for the same problem with a $MAPE = 44.06\%$ [49]. In this article, the author clearly demonstrates the difficulty of obtaining quality predictions when attempting to explain individual electricity consumption. He shows that on a sample of 29,000 people, the prediction error at national level is $1.97\% MAPE$. This error rises to 5.15% for a university campus, 13.8% for a village, and climbs to around 50% at the individual level.

Chapter 4

Data preprocessing

Data preprocessing is the first step in any machine learning problem. It consists of formatting, cleaning, transforming the data, to make it processable by an algorithm.

For this problem WeSmart gave me access to eight excel files, each containing quarter-hourly electricity meter readings from a member of a community. First of all, we need to check that the dates are correctly formatted and spaced. It is not uncommon for meters to introduce small errors such as duplicates or gaps. Duplicates can be safely removed. For gaps, we simply fill in the missing dates, and estimate the readings by interpolation. Some of the files contained long periods where the readings remained unchanged. Even during the holidays, buildings consume a minimum of energy, so there's a good chance that these long gaps are caused by power failures or other anomalies. In this case, it is not possible to generate data by interpolation. We will therefore first calculate the actual consumption of each consumer. To do this, simply calculate the difference between the reading at time t and t-1 for every point t. Then, where consumption is equal to zero over a long period, we simulate values based on data taken from the same dataset that is randomly disturbed.

Meteorological conditions can be interesting in explaining a household's electricity consumption because of the impact it can have on people's behaviour. For example, we can expect to observe an increase in consumption during hot weather among owners of air conditioning, or during the winter if the heating system is powered by electricity. Episodes of rain and wind could also be accompanied by such an increase, as people tend to stay at home during bad weather. We do not possess information on the heating, air conditioning or ventilation equipment, so it is possible that the consumption profiles do not vary in the same way when compared with external conditions. In a subsequent section, we'll explain how we can select the most relevant data for each profile.

Using the geographical coordinates of the energy community and an API freely provided by Open-Meteo, we can retrieve the meteorological variables of interest for a given location and on the desired dates. The query is the following:

```
https://archive-api.open-meteo.com/v1/archive?
latitude=xxxx&longitude=xxxx&start_date=2023-04-09&end_date=2023-04-23
&hourly=temperature_2m,relativehumidity_2m,dewpoint_2m,
apparent_temperature,pressure_msl,surface_pressure,snowfall,
weathercode,cloudcover,cloudcover_low,cloudcover_mid,cloudcover_high,
shortwave_radiation,direct_radiation,diffuse_radiation,
direct_normal_irradiance,windspeed_10m,winddirection_10m,windgusts_10m,
vapor_pressure_deficit,precipitation
```

This query returns all the variables in "JavaScript Object Notation format". It is easy to format this data using Python to convert it into a "comma-separated values" file we're interested in. This API only provides data minimum on an hour-by-hour basis. We therefore use interpolation to fill in the gaps and obtain quarter-hourly meteorological data. We concatenate this new dataset with each of the other eight files at the corresponding dates, to obtain datasets on which we can start working.

1 Management of outliers

Outliers are samples that contrast strongly with the rest of the dataset. They may be the result of a measurement error or simply the cause of high variability in the data. It is important to manage these deviant samples, as they have a negative impact on the performance of machine learning models. Intuitively, it is not interesting to train a model with values that have a very low probability of occurring in reality. This can give poor indications for prediction on unseen data by introducing unnecessary noise. It is therefore preferable to get rid of them.

Consider the dataset 4 as an example. The box-plot 4.1 is an intuitive way of representing the distribution of the data and observing outliers. The box covers the first quartile through to the third quartile. This means that there are 25% of the data below the box and 25% above. The orange line is the median, which separates the dataset such that 50% of the samples have a higher value and 50% a lower value. The graph shows that almost 75% of the data is below 100Wh. After calculation, only 0.27% are above 800Wh, representing 106 samples out of 38416. We can therefore consider, without risking losing important information, that all the data above 800Wh are outliers.

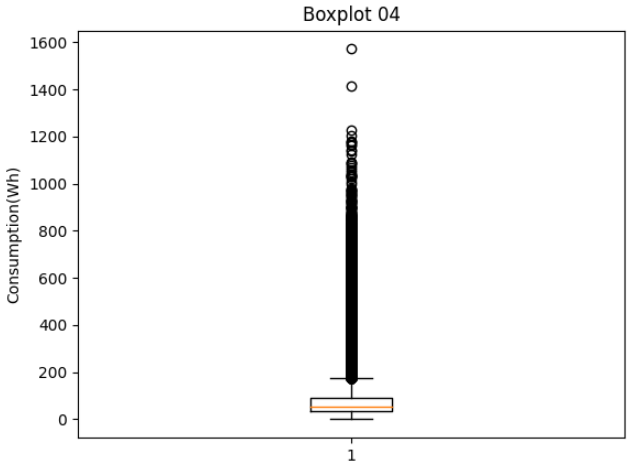


Figure 4.1: Box-plot of electricity consumption values from the fourth dataset.

There are several ways of managing outliers. The simplest would be to delete the rows containing those. However, this solution is not suitable for times series, as the dates are used as indexes, which would cause problems later on in the calculations if removed. The second solution would be to replace these outliers with the capping value, i.e. 800Wh. We're going to modify this solution slightly by introducing randomness to make it seem more realistic. The outliers will therefore be replaced by a random value between 780Wh and 800Wh.

After these modifications, the variance of the dataset decreased from 16108 to 15253, which is still consequent. The variance of a dataset is a statistical measure that evaluates the dispersion of the data. The higher the variance, the more distant the data are from the mean and therefore from each other. We cannot draw too strong conclusions about the variance of a dataset. However, intuitively, the more dispersed the data, the less likely it is that a large number of similar samples will be observed. These small numbers of samples will therefore be more difficult to predict, as they will have less impact during model training. We are not going to further reduce this variance to avoid denaturing the dataset. The aim of this operation is to prevent the introduction of absurd values in the training of the models. On the other hand, we want to study the ability of these models to predict values in real-life situations. It is therefore important to keep the data sufficiently realistic.

2 Feature engineering

Features are the variables fed to the Machine Learning model for training. It is the information from which we try to infer the values of the variable of interest. Feature engineering consists of creating this information from the raw data. It is important to extract as much information as possible from the data we have. Models are able to find links between data that are sometimes difficult or impossible for us to interpret, but they are nonetheless useful as long as they allow us to improve our predictions.

For the moment we have the dates of the meter readings and the meteorological data. We cannot use the dates in their current state, as they are the indexes to our dataset, so by definition they are all different. However, we can extract some useful information from them. The Python *datetime* class allows us to do this easily. The following characteristics are extracted:

- As the time of day cannot be interpreted as it stands, we will consider the number of minutes elapsed since midnight : [0, 15, 30, . . . , 1425].
- Day of the week : [1, 2, ..., 7].
- Weekend : [0,1].
- Day of the year : [1, 2, ..., 365].
- Week of the year : [1, 2, ..., 52].
- Month of the year : [1, 12].

Assuming that people develop electricity consumption habits over time, it may be interesting to add past consumption values to the features. In the context of this project, we estimate that there is a delay of around three days between the time when the data is read from the electricity meter and the time when we can add it to our training set.

For the first feature, we will try to capture consumption habits relating to the time of day in the near past (prev_4_d), and for the second one, we will try to capture habits relative to the day of the week (prev_4_w). This Python code can be used to compute these new features, using [3, 4, 5, 6] as timesteps for the first one, and [7, 14, 21, 28] for the second 4.2.

To obtain a more general result and to limit the influence of a value that would be exceptionally high or low, each past point is calculated as the average of the current point, the point just before and the point just after.

```
def past_consumption_feature(filename, timesteps):
    new_feature = []
    df = pd.read_csv(filename, index_col=['Datetime'])
    dates = [datetime.fromisoformat(d) for d in df.index]
    first_date = dates[0]
    dates.reverse()

    for date in dates:
        past_cons = []
        for t in timesteps:
            prev_step = date - timedelta(days=t)
            if prev_step > first_date:
                before = str(prev_step - timedelta(minutes=15))
                after = str(prev_step + timedelta(minutes=15))
                tmp = [df.at[before, 'Consumption(Wh)'],
                       df.at[str(prev_step), 'Consumption(Wh)'],
                       df.at[after, 'Consumption(Wh)']]
                past_cons.append(np.mean(tmp))
        new_feature.append(np.mean(past_cons))
    new_feature.reverse()
    df['new_feature'] = new_feature
    df.to_csv(filename)
```

Figure 4.2: Algorithm used to build prev_4_d and prev_4_w features [50].

Chapter 5

Feature Selection

The feature selection phase is essential in the construction of a machine learning model for high dimension problem. It consists of selecting a subset of relevant variables from all those that are available. The objective is to reduce the dimension of the problem to speed up the training phase and to get rid of features that are not interesting for explaining the variable of interest. The datasets currently contain 27 features. It would be far too costly in terms of computing time to train models with so many dimensions, and it could also introduce unnecessary noise. The curse of dimensionality is a phenomenon that affects many problems in mathematics and computer science when dealing with data in a large dimensional space. In machine learning, the size of the dataset must increase exponentially with respect to the number of features in order to achieve good performance. This is simply due to the fact that there must be enough data samples for each possible combination of features in order to train the model efficiently. Hughes' phenomenon is a good illustration of the curse of dimensionality in Machine Learning. It states that for a given dataset, the performance of a regression or classification model increases with the number of features until an optimal point is reached, beyond which the addition of features degrades performance [51].

1 Correlation

The first strategy consists of filtering the characteristics, keeping only those that have a relatively high correlation coefficient with the variable of interest compared with the others. There are several ways of assessing the correlation between two variables, the most widely known being Pearson's and Spearman's correlation coefficients.

1.1 Pearson

The Pearson correlation coefficient is a statistical measure of the linear correlation between two variables. It is calculated as the ratio of the covariance between the two variables over the product of their standard deviations :

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X * \sigma_Y}$$

The result is a coefficient between -1 and 1 showing the direction and strength of the linear relationship. A result close to 0 indicates an almost zero relationship, while a result close to -1 or 1 indicates a very strong negative or positive relationship between the variables. For each dataset, we select the 5 features with the highest Pearson correlation coefficient, beyond that, the values are close to zero and thus irrelevant. [52].

Dataset	Selected features				
1	prev_4d	prev_4w	dewpoint	diffuse_radiation	apparent_temp
2	prev_4w	prev_4d	weekend	day_of_week	relativehumidity
3	prev_4w	prev_4d	minutes	diffuse_radiation	shortwave_radiation
4	prev_4w	prev_4d	minutes	weekend	day_of_week
5	prev_4w	prev_4d	dewpoint	apparent_temp	temperature
6	prev_4w	prev_4d	windspeed	day_of_year	day_of_week
7	prev_4w	prev_4d	temperature	apparent_temp	diffuse_radiation
8	prev_4w	prev_4d	minutes	temperature	apparent_temp

Table 5.1: Selected features with Pearson's coefficients method.

The table shows that the same features recur frequently. The two features that are supposed to capture consumption habits according to the time of day and the day of the week shows the strongest linear correlation in all datasets with coefficients varying between 0.6 and 0.13. Then temporal features such as the time of day, the day of the week or the fact that it's the weekend show relatively high scores between 0.29 and 0.15. Finally, features relating to temperature or radiation show a weak negative correlation. This shows that consumption would tend to decrease at the same time as temperature and radiation increase. It is important to note that a correlation does not necessarily mean that there is a causality. However, these correlations support our intuition that people develop consumption habits over time, and that there could indeed be a link between electricity consumption and weather conditions.

1.2 Spearman

In statistics, Spearman's rank correlation coefficient is employed to assess the rank correlation existing between two variables. The ranking is simply the ordering of the observations of a variable from the smallest to the largest. This coefficient is calculated as the ratio of the covariance between the two rank variables over the product of their standard deviations. It is therefore defined as the pearson correlation coefficient between the rank variables. In doing so, the Spearman rank coefficient assesses whether there is a monotonic relationship between the two variables rather than a linear relationship.

$$r_s = \frac{cov(R(X),R(Y))}{\sigma_{R(X)} * \sigma_{R(Y)}}$$

It also provides a result between -1 and 1 indicating the direction and strength of the monotonic relationship. One notable difference is that the Spearman coefficient is robust against outliers, unlike the Pearson coefficient.

The features with the highest Spearman rank coefficients are very similar to those obtained with Pearson coefficients. However, the coefficients show higher values. This indicates that the relationships between these features and electricity consumption are monotonic and non linear. It is therefore highly likely that a linear model is not sufficient to explain electricity consumption based on these features [53].

2 Mutual Information

Mutual information is a non-negative value that measures the dependence between two random variables. In other words, it is a measure of the information that one variable gives about the other. Entropy is a mathematical function that measures the amount of uncertainty contained in a random variable. In our case, the random variables are the features in our dataset. The entropy of the feature X is given by :

$$E(X) = -\sum_{i=1}^{n} P(X = x_i) * log_2(P(X = x_i))$$

Where n is the number of different values the feature X can take. Conditional entropy gives us the amount of information contained in a random variable Y given X :

$$E(Y|X) = -\sum_{i,j} P(X = x_i \wedge Y = y_j) * log_2(P(X = x_i|Y = y_j))$$

The mutual information between the random variables X and Y is given by :

$$I(X;Y) = E(Y) - E(Y|X)$$

Mutual information can therefore be interpreted as the gain in information about the variable Y when X is known.

Once again, the best features selected using mutual information are similar to those selected using correlation coefficients, but with even fewer differences between the datasets. We still find the two features created from past consumption data. Then, in terms of temporal features, it's the time of day that systematically recurs. In terms of meteorological features, it's the perceived temperature and the dew point that appear the most, the dew point being the temperature below which the water vapour in the air condenses.

This feature selection strategy seems to be well suited to our problem, as it allows us to capture complex relationships between features and the variable of interest, unlike the two previous strategies, which were respectively limited to linear and monotonic relationships. However, this does not necessarily promise that this feature selection strategy will produce the best possible accuracy for each model.

3 Wrapper Method

Wrapper methods are "greedy" feature selection algorithms which test subsets of features against a evaluation criteria. The main difference compared with previous methods is that these algorithms look for the best possible combination of features for a given dataset and a specific model. This may be very useful for improving the accuracy of a particular model, however, it can be extremely costly in terms of computing time, with a complexity that is exponential in the worst case with respect to the number of features to be tested.

The forward selection algorithm is a simplified version of the greedy method. Instead of testing all possible feature combinations, the algorithm iteratively selects the feature that offers the best performance for a given model when added to the features already selected. The complexity is $O(n * m)$ where m is the complexity of the model and n is the number of variables, against $O(2^n * m)$ for a greedy method.

For a regression problem, the R^2 score is typically used as an evaluation criterion. It is widely used to assess the quality of a linear regression. It can be interpreted as the performance of the model compared to a simplified model that predicts the mean of the variable of interest for all inputs. The higher the R^2 score, the better.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - y_i')^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2}$$

It might also be interesting to use the forward selection algorithm with the negative *Mean Absolute Percentage Error* as evaluation criteria as it is the metric that we will try to optimise for our problem 6.1.

4 feature redundancy

Selecting the most relevant features is not always enough to obtain an efficient model. Indeed, redundancy in the selected features can also have a negative effect on the model's performance. Redundant variables signify that they both provide very similar information to the model, and are therefore highly correlated with each other. This tends to degrade the generalisation capacity of the model due to the over-representation of certain variables compared to others. This can lead to a loss of accuracy, as well as prolonging training and convergence times. To eliminate redundancy, we simply use one of the strategies described above on the features selected beforehand. This strategy is applied between each pair of features; if a pair obtains a high score, the feature that is least useful for explaining the variable of interest is deleted.

Chapter 6

Regression models

Before embarking on the review of machine learning models, it is useful to explain the choice of metric and method for evaluating their performance, as well as the tools used to tune their parameters.

First of all, we will use the *Mean Absolute Percentage Error* metric. This calculates the average percentage deviation of our prediction from the actual values, so the lower the better.

$$MAPE = \frac{100}{n}\sum_{i=1}^{n}|\frac{y_i - y_i'}{y_i}| \tag{6.1}$$

This metric has the advantage of being very intuitive, as it provides a percentage value that is easy for us to interpret. Furthermore, it allows us to compare the performance of a model on different datasets efficiently, as it is a relative measure.

The most widely used method for evaluating the performance of a machine learning model is the K-fold-cross-validation technique. This involves dividing the data set into K equal parts, and during K iterations, training the model on K-1 folds (in blue), and testing it on the remaining folds (in orange) 6.1. The average error over these K iterations is then calculated. The problem with this method is that the test fold falls between the training folds. However, for time series problems, we must necessarily select a testset that comes after the training set. Indeed, it would make no sense to train a model to predict past data. We therefore use an alternative version: the blocking time series split. This method consists of selecting a certain period for the training, 16 weeks in our case, and the period just after for the test, which in our case is 1 week. We then advance the training set by a certain period, 3 weeks in our case, and so on, for a predefined number of iterations.

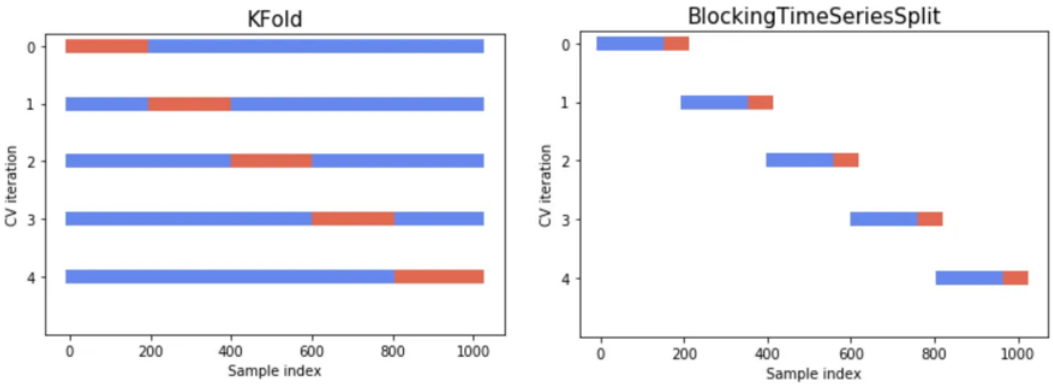


Figure 6.1: K-fold and Blocking Time Series Split illustration from [54]

Parameter tuning can be a complicated step when building a machine learning model. The Grid Search method makes it possible to test all the combinations of parameters provided beforehand, and to select the best one. These searches can take a very long time, depending on the model and the number of parameters, so it is important to choose the values inserted in the search grid carefully.

1 Linear Regression

The very first model implemented is a simple linear model. This is a good way of approaching the problem because it is simple and highly interpretable. To select features, Pearson correlation coefficients seem to be the most appropriate strategy, as it allows to select those that have a strong linear relationship with the variable of interest. However, experiments have shown that features selected with Spearman coefficients give better results for this model.

We will illustrate this model on an arbitrary portion of the dataset 01. We take the period from 24 November 2020 to 24 February 2021 for the training and the day of 27 February 2021 for the test. The five features with the highest Spearman coefficients are : prev_4d, prev_4w, dewpoint, direct_normal_irradiance, shortwave_radiation. However, these last two features are highly redundant, with a correlation coefficient of 0.94. We can therefore remove the last one from the model. Once fitted, the linear formula is :

$y = 17.59 + 0.47 * prev\_4d + 0.36 * prev\_4w - 0.57 * dewpoint - 0.008 * direct\_normal\_irradiance$

Here below, the graphical representation of the last three days of training and

the prediction for 27 February 2021 6.2. The training graph shows that the model struggles to fit the training data. The *MAPE* for this test is 32%, which is not too bad. However, on average, this model performs very poorly, with a *MAPE* of 63%.

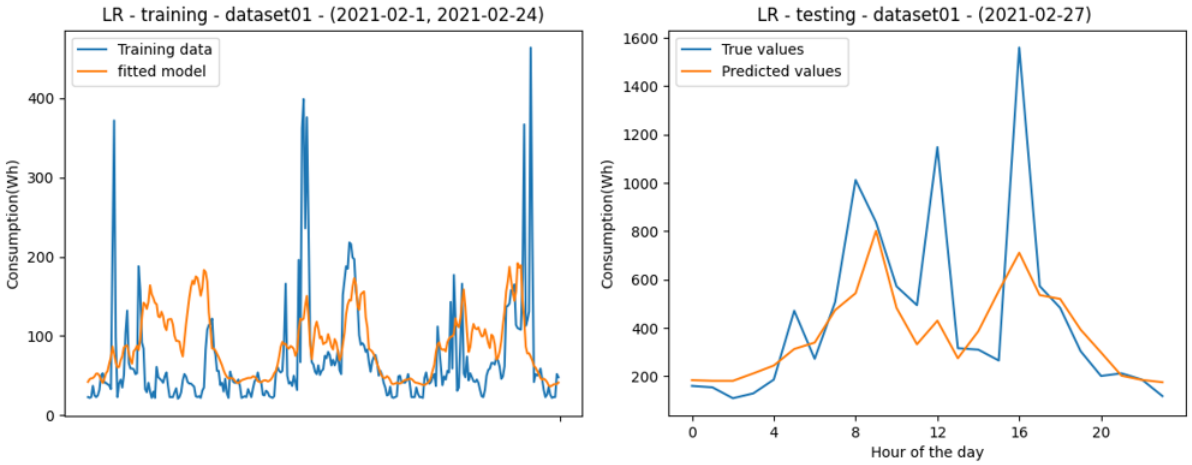


Figure 6.2: Training and testing visualisation of the linear regression model.

Dataset	MAPE
01	43%
02	43%
03	97%
04	88%
05	29%
06	76%
07	50%
08	73%
Average	63%

Table 6.1: Average Mean Absolute Percentage Error of Linear regression model.

2 K-Nearest Neighbours

Several parameters need to be tuned for KNN. The K number of neighbours used to approximate each entry point. The choice of algorithm used to calculate the nearest neighbours (brute force, ball tree, K-D tree). The distance measure used to evaluate the spacing between points, and the weighting strategy used to estimate a data point from the K nearest points (uniform or proportional to distance).

The best choice for these parameters depends on the nature of the problem. The strategy is therefore simply to test all the combinations with a GridSearch. The parameters found are the following:

- K neighbours = 40.
- Algorithm = brute force.
- Metric = Euclidian distance
- Weighting = uniform

When visualising the training graph, we can see that the adaptation of the model to the training data depends heavily on the weighting method used to estimate the input points. with uniform weighting, the general trend is fairly well reflected by the model, whereas with proportional weighting, the model largely overfits the training data, and this for any value of K.

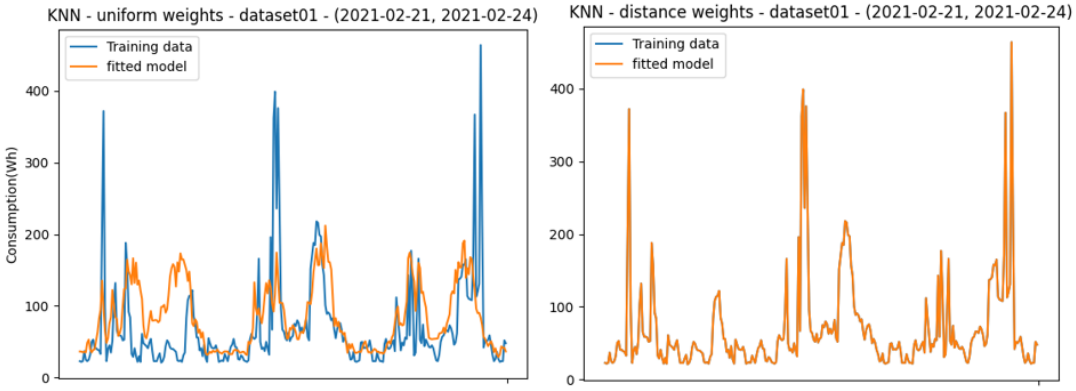


Figure 6.3: Training visualisation of KNN model.

In terms of testing, both strategies produce a MAPE of around 30% on our test. On the whole, the model performs poorly, with an average MAPE of 61%. The feature selection method used to obtain this result is the Spearman coefficients. However, the difference in error compared to the other methods is small.

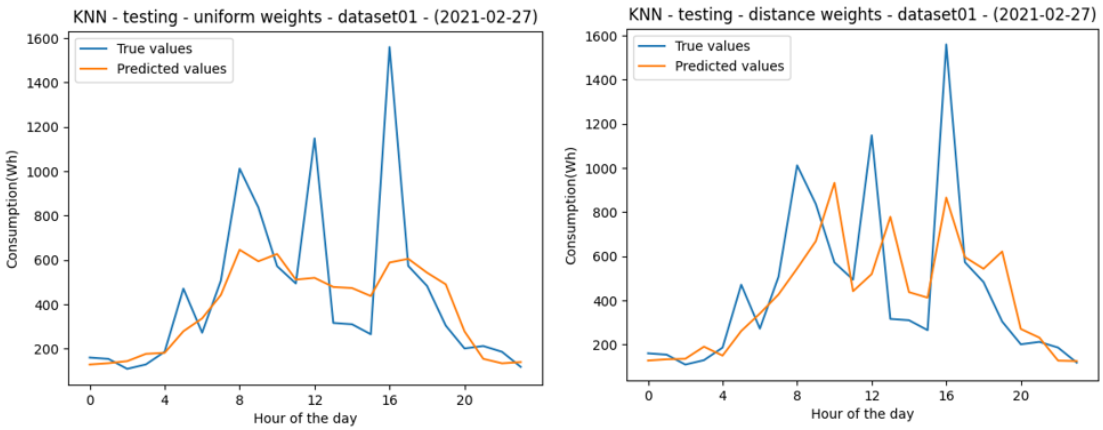


Figure 6.4: Testing visualisation of KNN model.

Dataset	MAPE
01	43%
02	49%
03	84%
04	85%
05	31%
06	73%
07	50%
08	70%
Average	61%

Table 6.2: Average Mean Absolute Percentage Error of KNN regression model.

3 Support Vector Machine

The parameters to be tuned for the support vector machine regression model are ; epsilon which defines the accepted error margin, C which defines the penalty applied to samples outside the error margin, the kernel type and gamma which is here calculated as :

$$\gamma = \frac{1}{2\sigma^2}$$

This parameter is used by the kernel to determine the influence distance of a single sample in the calculation of similarity between points. A small value for gamma

indicates that the influence distance of each point is large. This will improve the generalisation capacity of the model. Conversely, a large value will force the model to fit the data more closely.

The kernel used is the Radial basis function kernel, which is the default choice for this model. It is complex, and can combine several polynomial kernels of different degrees, which makes it very powerful. In addition, it does not require any additional parameter tuning unlike polynomial kernel. The optimal values for Epsilon and C are 0.4, 5 respectively and were determined by using GridSearch. The optimal value for gamma is calculated as follows:

$$\gamma = \frac{1}{n\ features * Var(X)}$$

This is the default formula proposed by scikit-learn's SVM class. It gives a value inversely proportional to the variance of the training data and the number of features. In this way, gamma will be small for highly dispersed data to prevent overfitting.

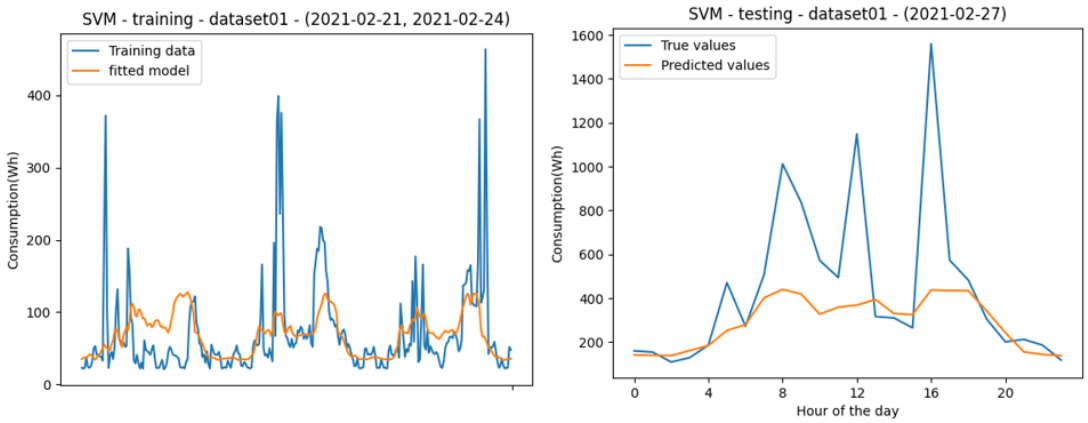


Figure 6.5: Training and testing visualisation of the SVM regression model.

The graphical representation of the fitted model shows that the curves are less sharp than for linear regression. Peaks of electrical consumption are barely, or not at all, fitted by the model. The result is a smoother prediction, which at first glance may appear less accurate. However, for this test, the MAPE is only 27%. This model also performs better overall, with an average MAPE of 38%. The feature selection method used was Pearson coefficients, but the difference in the error produced compared with the other methods proposed was negligible.

Dataset	MAPE
01	29%
02	36%
03	49%
04	45%
05	23%
06	49%
07	32%
08	38%
Average	38%

Table 6.3: Average Mean Absolute Percentage Error of SVM regression model.

4 Tree-based

4.1 Random Forest

The main parameters to be tuned for the random forest model are the following:

- The number of estimators, which defines the number of decision trees used for prediction.
- The maximum depth of each decision tree.
- The number of features randomly selected to find the best split.
- The number of samples used for bootstrap resampling.

The most significant parameter is the depth of the decision trees. When the trees are deep, the decisions to split the data become increasingly specific, leading to significant overfitting. For our problem, a depth of 6 is optimal. Then, beyond 150 estimators, the model's performance no longer improves. We can therefore consider that this number is sufficient to obtain good results, without the execution time being too long. The next two parameters make it possible to introduce randomness into the model to reduce overfitting. The number of randomly selected features is equal to 2, and the number of samples for bootstrap resampling is 85% of the training set.

For the same test as before, the model shows a MAPE of 29%, and in average, the error is 39%, which is not too bad compared to other models.

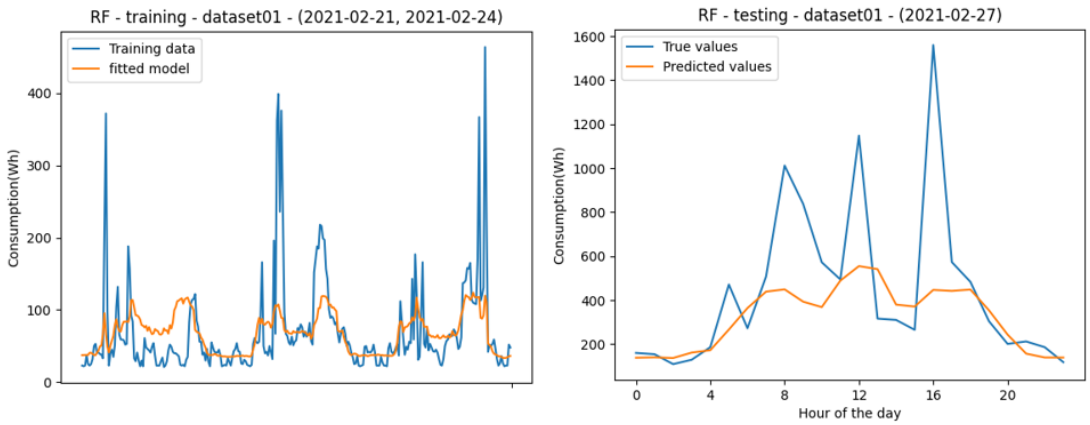


Figure 6.6: Training and testing visualisation of the Random Forest regression model.

Dataset	MAPE
01	30%
02	38%
03	46%
04	48%
05	25%
06	52%
07	31%
08	42%
Average	39%

Table 6.4: Average Mean Absolute Percentage Error of RF regression model.

4.2 XGBoost

Apart from the learning rate, the parameters to be tuned for the XGBoost model are the same as for Random forest. The most significant parameters are the learning rate and the number of estimators. A small value for the learning rate increases the convergence time, but improves the generalisation capacity of the model, while the adaptation of the model to the training data increases with the number of estimators. Tuning these two parameters is essential to find the right compromise. In addition, as with Random Forest, the depth of the decision trees has a significant impact on model fitting. Finally, we can tune the two parameters responsible for the randomness in the construction of each decision tree to reduce overfitting.

The optimal values of these parameters for this problem are as follows:

- Learning rate: 0.008.
- Number of estimators: 135.
- Maximum depth of decision trees: 4.
- Number of features randomly selected for each split: 2 out of 5.
- Bootstrap sampling : 95% of the training set.

This model produces a MAPE of 33% on the usual test, and 40% on average.

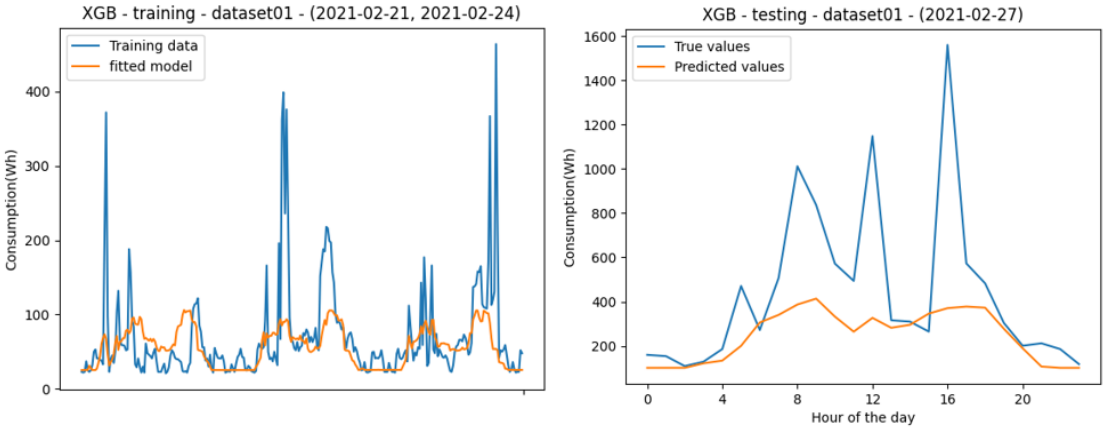


Figure 6.7: Training and testing visualisation of the XGB regression model.

Dataset	MAPE
01	30%
02	34%
03	56%
04	52%
05	28%
06	47%
07	29%
08	46%
Average	40%

Table 6.5: Average Mean Absolute Percentage Error of XGB regression model.

5 Multi-Layer Perceptron

The most sensitive part of building a neural network model is the choice of the number of hidden layers and their dimensioning. These choices depend on many parameters, such as the size of the training set, the number of features, the noise present in the data, etc. There are no precise rules for tuning hidden layers, but generally speaking, complex problems such as image or text recognition require a greater number of layers than simpler problems such as binary classification. Furthermore, the number of neurons in each layer should not exceed the number of neurons in the input layer, as this would increase the variance of the model and lead to overfitting.Another important parameter to tune is the learning rate, which is used to control the updating of the weight vectors during the backpropagation process. As with XGBoost, it is used to control model convergence.

After numerous tests and grid searches, no configuration was found that allowed the model to deliver compelling results. This model produces a MAPE of 32% on the usual test and 61% on average. The feature selection strategy used is the Spearman coefficients, and the chosen configuration is the following:

- Number of hidden layers: 3
- Number of neurons per layer: 10
- Activation function : Rectified Linear Unit
- Learning rate : 0.0001

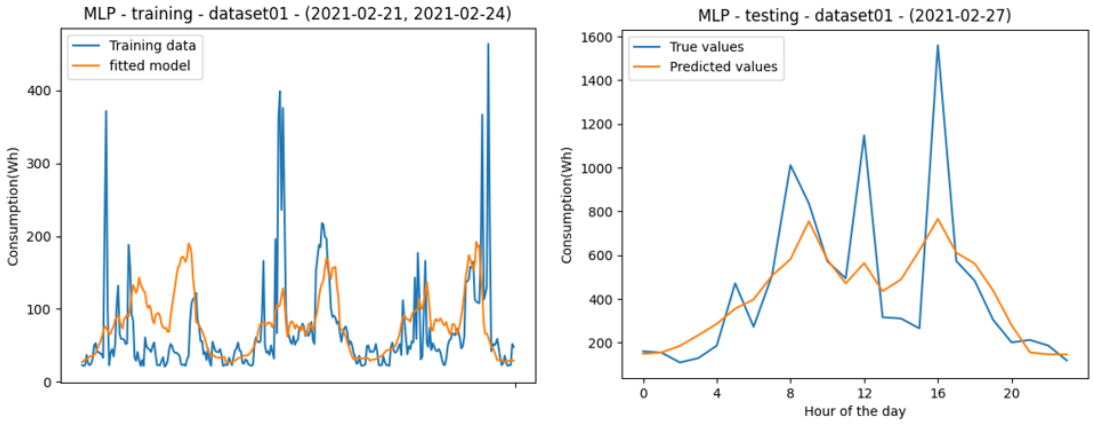


Figure 6.8: Training and testing visualisation of the MLP regression model.

Dataset	MAPE
01	37%
02	47%
03	97%
04	90%
05	29%
06	72%
07	49%
08	66%
Average	61%

Table 6.6: Average Mean Absolute Percentage Error of MLP regression model.

6 Remarks

Several comments can be made following these experiments. Firstly, for a given model, the choice of feature selection method does not significantly change the overall performance of the model. Indeed, if we take the example of the SVM model, Pearson's method provides the best performance, but the difference in MAPE compared with the other methods is around 0.1%, whereas for XGBoost, this difference does not exceed 1%. This can be explained by the fact that the features selected are often similar from one method to another. In addition, the two features "prev_4d" and "prev_4w" are selected almost systematically, with a rate of occurrence of 98.6% and 91% respectively for all methods combined. We can conclude that these are the two most relevant features for our problem.

Then, by analysing the training graphs for each model, it is easy to see that the more the model fits the data (which is reflected in a sharp curve and a large amplitude), the greater the prediction error. These characteristics are observed on the training graphs of the Linear regression, KNN and MLP models, and this is accompanied by a significant error. In other words, the more the model adapts to consumption peaks, the greater the prediction error. This is not surprising, as we saw in the section on managing outliers that very few samples make up these consumption peaks, so they are difficult to predict and tend to lead the models into error, as we can see from this graph of the linear regression test 6.9.

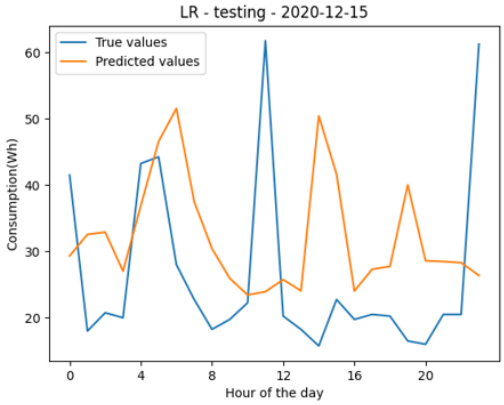


Figure 6.9: Test of the linear regression model showing a MAPE of 56%.

We have already removed the most aberrant samples from the data sets. We could not simplify the data any further, as this would have distorted the original problem. What remains is the tuning of the parameters of each model to control the adaptation of the models to the training data. The aim of parameter tuning is to find the optimum trade-off between variance and bias. These concepts have already been discussed in the review of machine learning models. Reducing the variance of a model improves its ability to generalise at the expense of bias. In other words, it makes the model less sensitive to the specificities of the training data, thereby increasing the error on the training data, but it also makes it possible to reduce the error on testing data.

However, finding the right trade-off does not always lead to a high-performance model. In the case of our problem, it is the robustness against outliers that will be decisive. A robust model is capable of capturing sufficient complexity in the training data while ignoring irrelevant samples. This is notably the case for algorithms based on decision trees. Their robustness derives from several characteristics. Firstly, at each branch, the model seeks to find the split that generates the greatest reduction in impurities by measuring the entropy, or Gini impurity index. These measures depend on the probability distribution of the data rather than the values themselves. Secondly, as long as the trees are not too deep, the model is less likely to capture non-pertinent data. In the case of the Support Vector Machine model, its robustness actually comes from its kernel. Tuning the parameter γ allows to define the influence distance of a single sample. In our case, it is inversely proportional to the variance of the training data. As a result, the influence of outliers is greatly reduced for highly dispersed data.

The ARIMA and CNN-LSTM models are used for time-series forecasting. They operate in a different way from those implemented above. A regression model is used to predict a variable from a set of other variables called features. A time-series forecasting model uses past observations of a variable to extrapolate future values for that same variable by learning patterns that recur over time. Exogenous variables can be added to the model to incorporate external factors during model training. These variables are called exogenous because they do not directly intervene in the learning of temporal patterns. These two models will not be implemented in this work, but may be the subject of a future study on this subject.

7 Final models

In this section, we have examined the effectiveness of multiple regression models for addressing our problem, aiming to identify the most suitable solution. To do this, the models were studied in such a way as to provide the best possible results on all the data sets available. The results show that three models stand out, namely : SVM, Random Forest and XGBoost.

Regression model	Average MAPE
Linear Regression	63%
K-Nearest Neighbours	61%
Support Vector Machine	38%
Random Forest	39%
XGBoost	40%
Multi-Layer Perceptron	61%

Table 6.7: Average Mean Absolute Percentage Error of each tested model.

We can now look for the best combination of model and parameters for each of the eight profiles. The feature selection method is set to Pearson coefficients, as we have seen that this choice has very little impact on model performance. The results are presented in the table below.

Dataset	Model	Parameters	MAPE
01	XGBoost	(0.0085, 165 4, 2, None)	29.1%
02	XGBoost	(0.008, 145, 4, 2, None)	33.6%
03	XGBoost	(0.0087, 60, 4, 1, 85%)	45.1%
04	XGBoost	(0.005, 110, 1, 1, None)	39.9%
05	SVM	(1, 0.05, 0.0001)	22%
06	XGBoost	(0.011, 70, 3, 2, 75%)	47.5%
07	XGBoost	(0.006, 170, 1, 1, 85%)	27.3%
08	XGBoost	(0.0075, 95, 1, 1, 95%)	38.2%
Average	-	-	35.33%

Table 6.8: final models, specific to each data profile.

Where parameters for XGBoost and SVM are in the form :

- XGBoost : (learning rate, n_estimators, max_depth, n_features, n_samples).
- SVM : (C, epsilon, gamma)

XGBoost stands out from the other two models. This can be explained by its high level of flexibility. The parameters can be fine-tuned, allowing the model to adapt to the problem in a very specific way, while remaining robust to extreme values, and thus gaining in accuracy. With Random Forest and SVM, a ceiling is reached more rapidly, and small modifications to the parameters have no significant impact on accuracy. We can conclude from these experiments that there is no unique model that would allow us to predict the electricity consumption of different households in an optimal way, but rather different models, each specifically adapted to a particular data profile.

Chapter 7

Anomaly detection

To obtain this solution, the models have been adapted specifically to each of the eight profiles. In this way, we were able to better capture the consumption habits of each household. However, habits evolve over time, and these solutions could become obsolete. It may be worth implementing an algorithm that can detect when predictions deviate too far from actual values, and to modify the models accordingly. To study the relevance of such an algorithm, we can simulate a change of habit by extracting a segment of the dataset from one profile, and then integrating it in another profile.

The idea is to define an error margin beyond which the model is re-trained by looking for new parameters that are more relevant to the portion of data that triggered the anomaly. However, it is possible for the prediction to exceed this margin of error due to exceptional behaviour. The point here is to detect when consumption patterns change. We will therefore define an anomaly counter. It is incremented when the prediction for a week produces an error that exceeds the threshold, and it is reinitialised otherwise. If this counter exceeds 3, the search for new parameters is triggered. During this search, the portion of the dataset used for training is reduced to 6 weeks so that the model adapts more specifically to the period that triggered the anomaly.

To test this algorithm, we have created two new profiles by concatenating profile 01 with 08 and 07 with 04. These are good candidates, because the error on these profiles is not too high, and the parameters of their respective models are different, which means that the profiles are sufficiently dissimilar to simulate a change in habit. The error threshold is set at 40%. A lower value would risk making the algorithm too sensitive, which could trigger false anomalies due to simple variations in behaviour. Conversely, a higher value could render the algorithm ineffective.

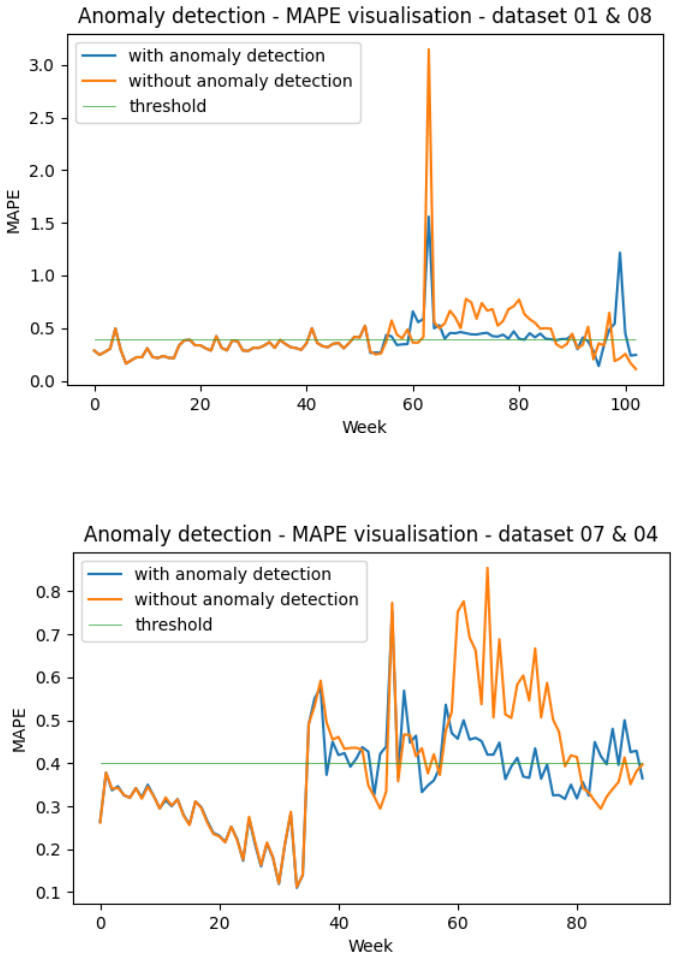


Figure 7.1: Evolution of the error with the anomaly detection algorithm.

Over approximately 100 weeks predicted on the first profile (01 & 08), the mean absolute percentage error is on average 42% without the anomaly detection algorithm, compared to 38% with it. However, it does not always do better, with a large error for week 100. For the second profile (07 & 04), the MAPE is 39.5% without and 36% with the algorithm.

Chapter 8

Results and discussion

The aim of this work was to study the feasibility of a tool for predicting the electricity consumption of households, and more specifically, of Belgian households that are part of an energy community. One constraint of this tool is that it must be capable of predicting far enough into the future, as WeSmart does not currently have real-time consumption data for its users.

The proposed solution consists of eight regression models, each adapted to a specific profile. These are trained on data from the previous 16 weeks to predict the following week. On average, the mean absolute percentage error is 35.33%. However, the error can vary greatly from one profile to another. The XGBoost model delivers better results than other models in most cases. This can be explained by its high robustness against outliers and its great flexibility. By implementing these prediction models, we assumed that we could explain a household's electricity consumption using its consumption history and weather data. The study of these variables has shown that the features most relevant to our problem are :

Feature name	Occurrence rate
prev_4d	98.6%
prev_4w	91%
day of week	50%
time of the day	47%
apparent temperature	48.6%
weekend	47%
dewpoint	38.3%
shortwave radiation	37%
wind speed	37%

Table 8.1: Features occurrence rate, all feature selection methods combined

If such a tool were to be used over the long term, the anomaly detection algorithm would ensure that predictions are maintained at a suitable level of accuracy. If a major change were to occur in a household's consumption habits, such as the purchase of an electric car, this algorithm would automatically adapt the model's parameters. Without this, the predictions could deviate significantly, rendering the tool obsolete.

To assess whether the results obtained during the experiments are good or not, we can compare them with the state of the art. There are not many articles dealing with the prediction of electricity consumption by individual households. The article most related to the subject of this thesis concerns the presentation of the CNN-LSTM model. The authors report an average MAPE of 40.38%, compared with 44.06% for a simple LSTM model. These results were obtained from a sample of 69 households, all part of an Australian government Smart City project. It would be imprudent to claim that the solution proposed in this work surpasses the one in this article, as the experiments were carried out on a limited number of samples. However, we can agree that the results obtained are satisfying.

We can also compare the performance of our solution against typical consumption profiles. Synthetic Load Profiles (SLP) are used by electricity suppliers to assess the quarter-hour-by-quarter-hour distribution of consumption over a year for consumers who do not have a smart meter. They take the form of tables, with a coefficient for each quarter-hour of the year. By multiplying these coefficients by the total electricity consumption, we obtain the consumption distribution.

Dataset	Proposed solution	SLP 2019	SLP 2020
01	29.1%	189%	188%
02	33.6%	158%	158%
03	45.1%	227%	227%
04	39.9%	169%	168%
05	22%	143%	140%
06	47.5%	208%	199%
07	27.3%	130%	131%
08	38.2%	162%	161%
Average	35.33%	173.25%	171.5

Table 8.2: Comparison of MAPE between the proposed solution against SLP.

Despite good results compared with the state of the art and SLPs, the prediction error on certain consumption profiles remains very high. This shows that the accuracy of estimations is highly dependent on individual behaviour. Other models could be implemented on this problem, and relaxing the time constraint would enable the problem to be studied from a shorter-term perspective. However, there is no guarantee of obtaining more accurate results for all profiles. For some households, the evolution of electricity consumption over time is far too variable for a machine learning model to provide a reasonable prediction. These two graphs show the evolution of the error as a function of a measure of variability 8.1. In the first, we use the coefficient of variation. It is calculated as the ratio between the standard deviation and the mean, expressed as a percentage. For the second, we calculate the average of the standard deviation as a function of the time of day. Both graphs show a relatively marked positive trend. It would have been interesting to have additional information on the households analysed, such as the number of adults/children and the heating system, to try and establish a link between the model's performance and certain household characteristics.

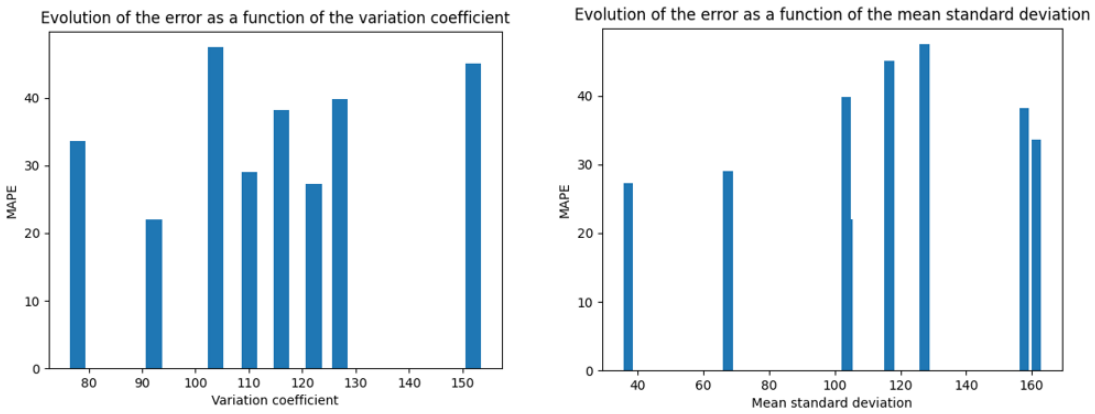


Figure 8.1: Evolution of the MAPE depending on variability.

Conclusion

Predicting household electricity consumption is a complex task. Although we can develop habits over time, many factors can randomly impact our consumption pattern. In this work, several machine learning regression models were tested in an attempt to solve this problem. Analysis of the data has shown that the most interesting features for household consumption forecasting are those designed to capture consumption habits over time. Then comes the time features, and certain meteorological data such as temperature, shortwave radiation and wind speed. Thanks to its robustness against outliers and its great flexibility, the XGBoost regression model performed best on seven of the eight profiles available. On average, the proposed solution produces a mean absolute percentage error of 35.33%. For some profiles, the error resulting from the model's predictions remains significant. There appears to be a slight positive trend between the variability of the electricity consumption and the error generated. In a real-life situation, this solution would not allow us to make sufficiently accurate predictions about consumption profiles that are too variable. This could lead the recommendation service to give irrelevant suggestions for certain households. In this case, the service could provide advice to these households in order to reduce the variability of their electricity demand, and thus improve the accuracy of the prediction algorithm over time. As the models are parameterised specifically for each profile, it is possible that performance may deteriorate over the long term as a result of a change in consumer habits. An anomaly detection algorithm would then make it possible to maintain a suitable level of accuracy by readjusting the model parameters when the predictions deviate too much from the actual values.

Bibliography

[1] B. W. Arnaud de Giovanni, "Renewable energy country attractiveness index 60 edition," 2022. [Online]. Available: https://assets.ey.com/content/dam/ey-sites/ey-com/en_gl/topics/power-and-utilities/ey-recai-60-v2.pdf.

[2] D. S. K. François Bouffard, "Centralised and distributed electricity systems," *Energy Policy*, vol. 36, no. 12, pp. 4504–4508, 2008, Foresight Sustainable Energy Management and the Built Environment Project, ISSN: 0301-4215. DOI: 10.1016/j.enpol.2008.09.060.

[3] M. T. G. Fazıl Gökgöz, "Energy security and renewable energy efficiency in eu," *Renewable and Sustainable Energy Reviews*, vol. 96, pp. 226–239, 2018, ISSN: 1364-0321. DOI: 10.1016/j.rser.2018.07.046.

[4] V. Azarova, J. Cohen, C. Friedl, and J. Reichl, "Designing local renewable energy communities to increase social acceptance: Evidence from a choice experiment in austria, germany, italy, and switzerland," *Energy Policy*, vol. 132, pp. 1176–1183, 2019, ISSN: 0301-4215. DOI: 10.1016/j.enpol.2019.06.067.

[5] J. de la Hoz, A. Alonso, S. Coronas, H. Martín, and J. Matas, "Impact of different regulatory structures on the management of energy communities," *Energies*, vol. 13, no. 11, p. 2892, Jun. 2020, ISSN: 1996-1073. DOI: 10.3390/en13112892. [Online]. Available: 10.3390/en13112892.

[6] N. E. Enock Mulenga Math H.J. Bollen, "Limits set by component loadability on solar power integration in distribution networks," *Electric Power Systems Research*, vol. 209, p. 107 951, 2022, ISSN: 0378-7796. DOI: 10.1016/j.epsr.2022.107951.

[7] M. M. Altmann, M. A. B. amd Mr. J.-Ch. Lanoix, M. D. Ellison, *et al.*, "Directorate general for internal policies, policy department a: Economic and scientific policy. industry, research and energy : Decentralized energy systems," 2010. [Online]. Available: https://www.europarl.europa.eu/document/activities/cont/201106/20110629ATT22897/20110629ATT22897EN.pdf.

[8] N. B. Mohammed Yekini Suberu Mohd Wazir Mustafa, "Energy storage systems for renewable energy power sector integration and mitigation of intermittency," *Renewable and Sustainable Energy Reviews*, vol. 35, pp. 499–514, 2014, ISSN: 1364-0321. DOI: 10.1016/j.rser.2014.04.009.

[9] E. Commission, "Smart grids and meters," 2019. [Online]. Available: https://energy.ec.europa.eu/topics/markets-and-consumers/smart-grids-and-meters_en.

[10] D. Hadjout, J. Torres, A. Troncoso, A. Sebaa, and F. Martínez-Álvarez, "Electricity consumption forecasting based on ensemble deep learning with application to the algerian market," *Energy*, vol. 243, p. 123 060, 2022, ISSN: 0360-5442. DOI: 10.1016/j.energy.2021.123060.

[11] M. Hossain, N. Madlool, N. Rahim, J. Selvaraj, A. Pandey, and A. F. Khan, "Role of smart grid in renewable energy: An overview," *Renewable and Sustainable Energy Reviews*, vol. 60, pp. 1168–1184, 2016, ISSN: 1364-0321. DOI: 10.1016/j.rser.2015.09.098.

[12] T. I. Frédéric Tounquet Clément Alaton, "Benchmarking smart metering deployment in the eu-28," pp. 16–26, 2020. DOI: 10.2833/492070.

[13] S. W. L. Gruber U. Bachhiesl, "The current state of research on energy communities," *e & i Elektrotechnik und Informationstechnik*, vol. 138, pp. 515–524, 2021, ISSN: 1613-7620. DOI: 10.1007/s00502-021-00943-9.

[14] N. M. P. Sachinkumar Suthar S. Hari Charan Cherukuri, "Peer-to-peer energy trading in smart grid: Frameworks, implementation methodologies, and demonstration projects," *Electric Power Systems Research*, vol. 214, p. 108 907, 2023, ISSN: 0378-7796. DOI: 10.1016/j.epsr.2022.108907.

[15] "Directive of the european parliament and of the council of 11 december 2018 on the promotion of the use of energy from renewable sources," 2018. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=celex:32018L2001.

[16] "Commission recommendation on speeding up permit-granting procedures for renewable energy projects and facilitating power purchase agreement," 2022. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=PI_COM%5C%3AC%5C%282022%5C%293219&qid=1653033569832#document1.

[17] "Directive of the european parliament and of the council of 5 june 2019 on common rules for the internal market for electricity and amending directive," 2019. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv:OJ.L_.2019.158.01.0125.01.ENG&toc=OJ:L:2019:158:TOC.

[18] "Décret modifiant diverses dispositions en matière d'énergie dans le cadre de la transposition partielle des directives 2019/944/ue du 5 juin 2019 concernant des règles communes pour le marché intérieur de l'électricité et 2018/2001/ue du 11 décembre 2018 relative à la promotion de l'utilisation de l'énergie produite à partir de sources renouvelables et en vue d'adapter les principes relatifs à la méthodologie tarifaire," 2022. [Online]. Available: https://www.ejustice.just.fgov.be/cgi/article_body.pl?language=fr&caller=summary&pub_date=22-10-05&numac=2022033591.

[19] "Crise énergétique: Pourquoi lie-t-on encore le prix de l'électricité au prix du gaz?," 2022. [Online]. Available: https://www.levif.be/economie/energie/crise-energetique-pourquoi-lie-t-on-encore-le-prix-de-lelectricite-au-prix-du-gaz/.

[20] K. Hoare, "Solar power self-consumption - the facts!," 2014. [Online]. Available: https://www.mysolarquotes.co.nz/blog/how-solar-power-works/solar-power-self-consumption---the-facts-/.

[21] S. N. Bureau, "Here is how you can calculate the annual solar energy output of a photovoltaic system," 2016. [Online]. Available: https://www.saurenergy.com/solar-energy-blog/here-is-how-you-can-calculate-the-annual-solar-energy-output-of-a-photovoltaic-system#:~:text=Globally%5C%20a%5C%20formula%5C%20E%5C%20%5C%3D%5C%20A,output%5C%20of%5C%20a%5C%20photovoltaic%5C%20system..

[22] J. Yan, Y. Liu, S. Han, C. Gu, and F. Li, "A robust probabilistic wind power forecasting method considering wind scenarios," in *3rd Renewable Power Generation Conference (RPG 2014)*, 2014, pp. 1–6. DOI: 10.1049/cp.2014.0828.

[23] A. Laouafi, M. Mordjaoui, and D. Dib, "One-hour ahead electric load and wind-solar power generation forecasting using artificial neural network," in *IREC2015 The Sixth International Renewable Energy Congress*, 2015, pp. 1–6. DOI: 10.1109/IREC.2015.7110894.

[24] Z. Wen, Y. Li, Y. Tan, Y. Cao, and S. Tian, "A combined forecasting method for renewable generations and loads in power systems," in *2015 IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC)*, 2015, pp. 1–5. DOI: 10.1109/APPEEC.2015.7380868.

[25] C. Cui, M. He, F. Di, Y. Lu, Y. Dai, and F. Lv, "Research on power load forecasting method based on lstm model," in *2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC)*, 2020, pp. 1657–1660. DOI: 10.1109/ITOEC49072.2020.9141684.

[26] I. S. Jahan, M. Prilepok, S. Misak, and V. Snasel, "Intelligent system for power load forecasting in off-grid platform," in *2018 19th International Scientific Conference on Electric Power Engineering (EPE)*, 2018, pp. 1–5. DOI: 10.1109/EPE.2018.8396034.

[27] M. Lekshmi and K. N. A. Subramanya, "Short-term load forecasting of 400kv grid substation using r-tool and study of influence of ambient temperature on the forecasted load," in *2019 Second International Conference on Advanced Computational and Communication Paradigms (ICACCP)*, 2019, pp. 1–5. DOI: 10.1109/ICACCP.2019.8883005.

[28] A. K. Imtiaz, N. B. Mariun, M. M. R. Amran, M. Saleem, N. I. A. Wahab, and Mohibullah, "Evaluation and forecasting of long term electricity consumption demand for malaysia by statistical analysis," in *2006 IEEE International Power and Energy Conference*, 2006, pp. 257–261. DOI: 10.1109/PECON.2006.346658.

[29] B. Peng, L. Liu, and Y. Wang, "Monthly electricity consumption forecast of the park based on hybrid forecasting method," in *2021 China International Conference on Electricity Distribution (CICED)*, 2021, pp. 789–793. DOI: 10.1109/CICED50259.2021.9556724.

[30] K. Yotov, S. Hadzhikoleva, E. Hadzhikolev, and D. Orozova, "Forecasting electricity consumption in a national power system," in *2022 22nd International Symposium on Electrical Apparatus and Technologies (SIELA)*, 2022, pp. 1–4. DOI: 10.1109/SIELA54794.2022.9845743.

[31] "Nearest neighbors," [Online]. Available: https://scikit-learn.org/stable/modules/neighbors.html#regression.

[32] "Understanding support vector machine regression," [Online]. Available: https://ch.mathworks.com/help/stats/understanding-support-vector-machine-regression.html.

[33] T. Sharp, "An introduction to support vector regression (svr)," 2020. [Online]. Available: https://towardsdatascience.com/an-introduction-to-support-vector-regression-svr-a3ebc1672c2.

[34] "Rbf svm parameters," [Online]. Available: https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html.

[35] "Boosting in machine learning," [Online]. Available: https://www.geeksforgeeks.org/boosting-in-machine-learning-boosting-and-adaboost/.

[36] "Decision trees," [Online]. Available: https://scikit-learn.org/stable/modules/tree.html#tree.

[37] "Ensemble methods," [Online]. Available: https://scikit-learn.org/stable/modules/ensemble.html#forest.

[38] "Arima simplified," [Online]. Available: https://towardsdatascience.com/arima-simplified-b63315f27cbc.

[39] W. contributors, "Artificial neuron," 2023. [Online]. Available: https://en.wikipedia.org/wiki/Artificial_neuron.

[40] C. Bento, "Multilayer perceptron explained with a real-life example and python code: Sentiment analysis," 2021. [Online]. Available: https://towardsdatascience.com/multilayer-perceptron-explained-with-a-real-life-example-and-python-code-sentiment-analysis-cb408ee93141.

[41] A. Saha, "The need for activation function along with hidden layers in a neural network," 2020. [Online]. Available: https://medium.com/analytics-vidhya/the-need-for-activation-function-along-with-hidden-layers-in-a-neural-network-8b37b4bffa42.

[42] J. Brownlee, "A gentle introduction to the rectified linear unit (relu)," 2019. [Online]. Available: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/.

[43] A. Bhattacharya, "Deep hybrid learning — a fusion of conventional ml with state of the art dl," 2020. [Online]. Available: https://towardsdatascience.com/deep-hybrid-learning-a-fusion-of-conventional-ml-with-state-of-the-art-dl-cb43887fe14.

[44] M. Alhussein, K. Aurangzeb, and S. I. Haider, "Hybrid cnn-lstm model for short-term individual household load forecasting," *IEEE Access*, vol. 8, pp. 180 544–180 557, 2020. DOI: 10.1109/ACCESS.2020.3028281.

[45] J. Brownlee, "How do convolutional layers work in deep learning neural networks?," 2019. [Online]. Available: https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/.

[46] W. contributors, "Convolutional neural network," 2023. [Online]. Available: https://en.wikipedia.org/wiki/Convolutional_neural_network.

[47] A. Biswal, "Recurrent neural network(rnn) tutorial: Types, examples, lstm and more," 2023. [Online]. Available: https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn.

[48] W. contributors, "Long short-term memory," 2023. [Online]. Available: https://en.wikipedia.org/wiki/Long_short-term_memory.

[49] W. Kong, Z. Y. Dong, Y. Jia, D. J. Hill, Y. Xu, and Y. Zhang, "Short-term residential load forecasting based on lstm recurrent neural network," *IEEE Transactions on Smart Grid*, vol. 10, no. 1, pp. 841–851, 2019. DOI: 10.1109/TSG.2017.2753802.

[50] L. Misselyn, "Master thesis repository," 2023. [Online]. Available: https://github.com/lmisselyn/Master-thesis.

[51] W. contributors, "Curse of dimensionality," 2023. [Online]. Available: https://en.wikipedia.org/wiki/Curse_of_dimensionality.

[52] W. contributors, "Pearson correlation coefficient," 2023. [Online]. Available: https://en.wikipedia.org/wiki/Pearson_correlation_coefficient.

[53] W. contributors, "Spearman's rank correlation coefficient," 2023. [Online]. Available: https://en.wikipedia.org/wiki/Spearman%27s_rank_correlation_coefficient.

[54] S. Shrivastaval, "Cross validation in time series," 2020. [Online]. Available: https://medium.com/@soumyachess1496/cross-validation-in-time-series-566ae4981ce4.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**