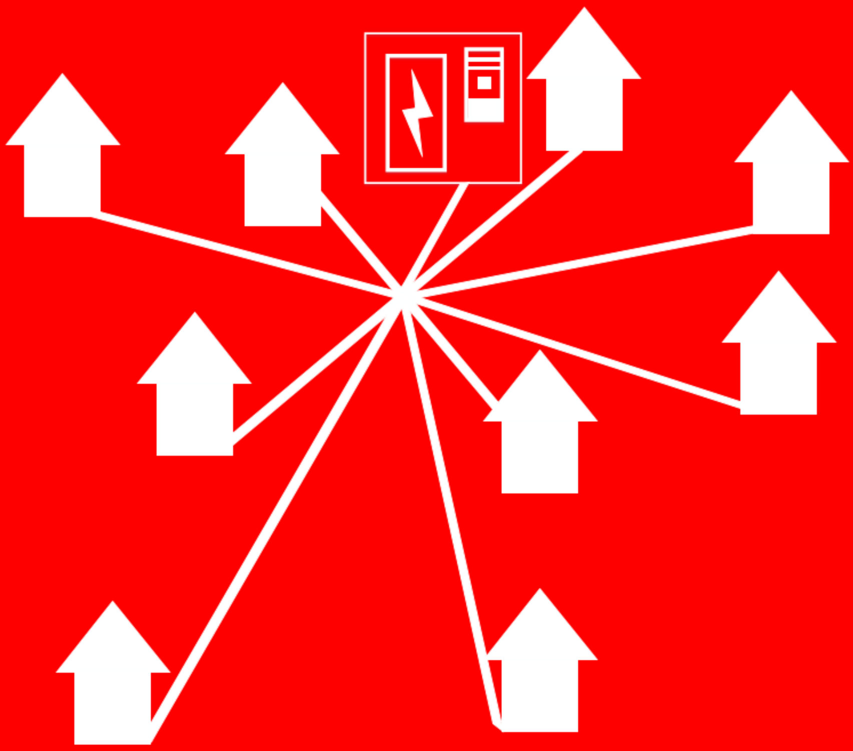
Machine learning approach to predict the energy sharing potential in a neighbourhood.

Determining the sharing potential by analysing smart meter data.



Construction Management and Engineering

Student: V.B.C. (Vince) Bergkamp

ID: 1020277

August 2019

This page is intentionally left blank.

Colophon

TITLE:	Machine learning approach to determine the energy sharing potential in a neighbourhood.
SUB TITLE:	Determining the sharing potential by analysing smart meter data.
DOCUMENT:	Master Thesis
DATE	August 6th 2019
VERSION	1

AUTHOR	V.B.C. (Vince) Bergkamp
ID NUMBER	1020277
ADRESS	Stationsstraat 31
	5701 MK Helmond
PHONE NUMBER	+31 (0) 6 39 239 599
E-MAIL	vince.bergkamp@hotmail.com
LINKED-IN	www.linkedin.com/in/vincebergkamp/

INSTITUTE	Eindhoven University of Technology
FACULTY	Department of the Built Environment
MASTER PROGRAM	Construction Management and Engineering
ADRESS	Den Dolech 3
	5612 AZ Eindhoven

CHAIRMAN	*Prof.dr.ir B. (Bauke) de Vries (Full Professor)*
1st GRADUATION SUPERVISOR	*Dr.ir. D. (Dujuan) Yang (Assistant Professor)*
2nd GRADUATION SUPERVISOR	*Ir. S. (Shalika) Walker (Ph.D. Candidate)*

This page is intentionally left blank.

Abstract

The application of energy storage is crucial for the efficient use of renewable energy. In order to adopt and operate energy storage systems in the energy grid, accurate short-term forecasting of the energy load is essential. In this study, two years of (fifteen-minute interval) electricity consumption data of seventy dwellings inside a residential district is analysed. To determine the best suitable electricity demand prediction model for the given data, a literature study together with an empirical test of multiple machine learning models is conducted. Prior to the prediction algorithm, a clustering model is ran to group households with similar demand profiles in a two week period preceding the prediction. A random forest regressor is used to predict hourly electricity demand for each cluster, one day ahead. Over ten validation days, the proposed model achieves, on average, a R^2 score of 0.77 and a cumulative variation of root mean squared error (cv-RMSE) of 0.41. The k-means clustering, together with a random forest, is able to predict the trends of hourly electricity loads one day ahead, and determine the indirect sharing potential. The most important prediction feature is the electricity consumption of the previous day. Other important features are humidity, outside temperature and electricity consumption of two and three days previous. Based on the validations, it is estimated that the neighbourhood can be completely self-sufficient for seven out of twelve months with a storage system of 660kWh, which is 9.43kWh per dwelling. When applying long-term energy storage, the self-sufficiency could be increased even more. This study covers the prediction of short-term energy demand of a small residential district with on-site renewables, and uses clustering in the pre-processing of the data, which is a new approach in this field of research. Currently this study used only individual days for validation. Validating over longer periods of time, at least multiple days, could increase the understanding of the actual sharing potential between dwellings. Also, by experimenting with additional user specific prediction features such as; occupancy, appliances ownerships, and solar radiation, improved model fitting might be achieved.

Key words: *Energy demand prediction, Machine learning, Random Forest, k-means clustering, Electrical Storage System, Energy Sharing.*

This page is intentionally left blank.

Summary

Ever since the world-wide agreement to pursue efforts to limit the temperature increase to 1.5˚C above pre-industrial levels, governments have started stimulating energy efficiency and sustainability measures by means of grants and regulations. One of the most important focus fields of these regulations are buildings, since they account for 40% of the global energy demand and 30% of global CO2 emissions. This has led to a growth in use of private renewable energy generation, primarily with photovoltaic (PV) panels, and has changed the energy system towards a more decentralized system. Due to the variability of renewable energy sources, energy storage systems must be implemented to make a complete energy transaction possible.

Accurate short-term electricity load forecasting is required in order to operate storage systems in the energy grid efficiently. Research in the field of energy demand prediction does not focus enough attention on local short term energy prediction with on-site renewables, and no straightforward energy storage operation systems are at hand. This study has proposed an approach to short-term forecasting of the electricity demand of a residential district, and efficiently operate an electrical storage system using on-site renewable energy.

Two years of, fifteen-minute interval, electricity usage data from 70 renovated dwellings was used to test the suggested approach. The dwellings are located in the centre of the Netherlands, have no natural gas connection, and are provided with a PV-system, heat pump, and high performing insulation. The building characteristics of the dwellings suggest the buildings can become mostly self-sufficient when a wisely chosen and smartly operated storage system is used.

The choice for the most suitable machine learning model was established by conducting a literature review, and empirically testing several machine learning methods. This led to the conclusion that random forest regressors are suitable for hourly one day ahead electricity demand prediction for the residential district.. The prediction model uses 38 features, historical electricity load, open source meteorological data and time variables such as day type, month, and season. Hyperparameters of the prediction model are optimized by means of a grid search algorithm which selects the best combination of hyperparameter settings. Preceding on running the prediction, electricity demand profiles of dwellings are clustered by means of a k-means clustering algorithm, which minimizes the within cluster variation of the electricity demand

To effectively use a storage system in the residential district, an operation system was proposed. For the operation system to be applicable some functional requirements are drawn up; the houses must be individually bidirectionally connected to the grid and the storage system, the operation system must have access to the real time smart metering data of the dwellings as well as to hourly weather forecast one day ahead. Furthermore, the operation system must be able to control all the switches in the electrical circuit of the neighbourhood. The operation system consists of seven conditional rules to optimize the local usage of renewable energy and thereby self-sufficiency.

The proposed model achieved, on average, over ten validation days, an R^2 score of 0.77 and a cv-RMSE of 0.41. The k-means clustering, together with a random forest regressor algorithm is able to predict the trends of hourly one day ahead electricity loads and determine the indirect sharing potential. The most influential prediction feature is the electricity consumption of the previous day. Other important features are humidity, outside temperature and electricity consumption of two and three days prior. The prediction model predicts better in spring and summer than autumn and winter. Based on the validations, the neighbourhood can entirely be self-sufficient for seven out of twelve months with a storage system of 660kWh, which is 9.43kWh per dwelling. When applying long-term energy storage, thermal or electrochemical, the self-sufficiency could be increased even more.

The model is only validated over single days, which is a shortcoming of this research. Validating over more extended periods, at least multiple days, should give a better understanding of the actual sharing potential between dwellings. Furthermore, by experimenting with additional prediction features such as; occupancy, appliances ownerships (including electric vehicle), and solar radiation, improved model fitting might be achieved. Incorporating dynamic pricing and long term storage in the operation system could also enhance the current approach since it can help to determine the financial feasibility.

Samenvatting

Sinds het wereldwijde akkoord op het limiteren van de opwarming van de aarde op +1,5 ˚C ten opzichte van het pre-industrieel tijdperk, zijn overheden gestart met het stimuleren van duurzame maatregelen door middel van subsidies en wetgeving. Een van de belangrijkste focus gebieden voor deze maatregelen is de gebouwde omgeving. Gebouwen in het algemeen zijn verantwoordelijk voor 40% van de wereld wijde energy behoefte en 30% van de CO^2 uitstoot. Dit heeft ertoe geleid dat de toepassing van particulieren hernieuwbare energieopwekking enorm is toegenomen de afgelopen jaren, vooral door middel van zonnepanelen. Dit heeft het energiesysteem veranderd van een centraal naar een gedecentraliseerd systeem. Doordat de hernieuwbare energie variabel aanwezig is, is het implementeren van energieopslag benodigd om een volledige energietransitie mogelijk te maken.

Om een energieopslagsysteem slim aan te sturen in het elektriciteitsnet, is nauwkeurige voorspelling van korte termijn elektriciteit behoefte benodigd. Onderzoeken op het gebied van het voorspellen van de behoefte van energie, hebben nauwelijks gefocust op lokale korte termijn energie behoefte waarbij ook hernieuwbare energie bronnen op locatie aanwezig zijn. Ook is er geen zijn er geen standaard aansturingssystemen voor opslagsystemen om overtollige hernieuwbare energie te delen. Het doel van dit onderzoek is het bieden van een methode om voor een woonwijk de lokale korte termijn elektriciteitsbehoefte te voorspellen. Verder wordt er een opzet voor een aansturingssysteem voor elektrisch energy opslag systeem voorgesteld om efficiënt gebruik te maken van lokale hernieuwbare energie om het zelfvoorzienend vermogen te vergroten.

Voor 70 gerenoveerde woningen is voor twee jaar aan elektriciteitsverbruik gegevens van slimme meter data beschikbaar gesteld, met metingsintervallen van vijftien minuten. De woningen, gelegen in het midden van Nederland, hebben geen gas aansluiting, en zijn voorzien van zonnepanelen, warmtepomp en hoogwaardige isolatie. De gebouw karakteristieken van de woningen suggereren de mogelijkheid om, met behulp van een elektrisch opslag systeem, grotendeels zelfvoorzienend te worden.

De keuze om een zo geschikt mogelijk machine learning model te selecteren, is gebaseerd op een literatuur studie en empirische testen van verschillende machine learning modellen. Hieruit is geconcludeerd dat een random forest regressor geschikt is voor het voorspellen van korte termijn elektriciteitsverbruik van een woonwijk. Voorafgaand aan de predictie, zijn de elektriciteitsverbruik profielen van de woningen geclusterd door middel van k-means cluster algoritme, welke de varianties binnen de clusters minimaliseert. Het voorspellend model gebruikt 38 variabelen, waaronder, historisch elektriciteitsverbruik, vrij beschikbare meteorologische gegevens en tijd gerelateerde gegevens, zoals indicators voor dag van de week, maand en seizoen. Hyperparameters van het voorspellend model zijn geoptimaliseerd door middel van een algoritme wat de beste combinatie van parameter instellingen zoekt door een groot aantal parameter instellingen te testen.

Om effectief gebruik te maken van een elektrisch opslagsysteem in een woonwijk, is er een aansturingssysteem voorgesteld. Om dit voorstel bruikbaar te maken zijn er een aantal vereiste aan de omgeving waar deze in komt te staan; de woningen dienen direct bi directioneel verbonden te zijn met het opslagsysteem en het elektriciteitsnet, het aansturingssysteem dient toegang te hebben tot de slimme meter standen en tot de uurlijkse weersvoorspelling voor de komende 24 uur. Tenslotte dient het systeem de schakelingen in het elektrische circuit te kunnen bedienen. Het besturingssysteem is gebaseerd op zeven conditionele regels om de gegenereerde hernieuwbare energy lokaal te gebruiken en daarmee het zelfvoorzienend vermogen te bevorderen.

Het gebruikte model behaald, gemiddeld over tien validatie dagen, een R^2 score 0,77 en een cv-RMSE van 0.41. De k-means clustering samen met de random forest regressor zijn in staat om de uurlijkse trend van de komende 24 uur te voorspellen en het indirecte uitwisselingspotentieel te bepalen. De belangrijkste variabele voor het voorspellend model is het energieverbruik van de vorige dag op hetzelfde tijdstip. Andere belangrijke variabele zijn de luchtvochtigheid, temperatuur en elektriciteitsverbruik van twee en drie dagen voorafgaand aan de voorspelling. Op basis van de validatie is ingeschat dat, de woonwijk zeven van de twaalf maanden per jaar volledig zelfvoorzienend kan zijn bij gebruik van een elektrisch opslagsysteem van 660kWh, 9,42kWh per huishouden. Wanneer er ook gebruik wordt gemaakt van lange termijn opslag, kan het zelfvoorzienend vermogen zelfs nog vergroot worden.

Het model is slechts gevalideerd op enkele dagen, wat een tekortkoming is van dit onderzoek. Wanneer er over langere periode gevalideerd wordt, minimaal meerdere dagen aaneengesloten, geeft dit een beter beeld van het daadwerkelijke uitwisselingspotentieel tussen de woningen. Ook kan de nauwkeurigheid van het model waarschijnlijk worden verbeterd door te experimenteren met extra variabelen zoals, aanwezigheid van bewoners, apparaten eigendom en zon intensiteit. Verder zou de huidige aanpak van het aansturingssysteem geoptimaliseerd kunnen worden naar de belangen van de eigenaar. Bijvoorbeeld door het in ogenschouw nemen van dynamische elektriciteitsprijzen en lange termijn opslag.

Table of content

This page is intentionally left blank.

List of abbreviations

ANN:	Artificial Neural Network
ANOVA:	Analysis of Variance
ARIMA:	Autoregressive Integrated Moving Average
ARMAX:	Autoregressive Integrated Moving Average with external variable
BPNN:	Back Propagation Neural Network
CO2:	Carbon Dioxide
CSV:	Comma-Separated Values
DNN:	Deep Neural Network
DSM:	Demand Side Management
DT:	Decision Tree
ELN:	Elastic Net
EUI:	Energy Usage Index
ESS:	Electrical Storage System
EV:	Electric Vehicle
GBM:	Gradient Boosting Machine
GRNN:	General Regression Neural Network
GW:	Giga Watt
HP:	Heat Pump
HV:	High Voltage
kWh:	Kilo Watt Hour
LSTM:	Long Short Term Memory
LV:	Low Voltage
MAE:	Mean Absolute Error
MAPE:	Mean Absolute Percentage Error
MaxAPE:	Maximum Absolute Percentage Error
ML:	Machine Learning
MLP:	Multilayer Perceptron
MLR:	Multi Linear Regression
MPE:	Mean Percentage Error
MSE:	Mean Square Error
MV:	Medium Voltage
NMSE:	Normalized Mean Square Error
OLS:	Ordinary Least Squares
PI:	Performance Index
PV:	Photovoltaic
RASE	Square Root of Average Square Error
RBFNN:	Radial Basis Function Neural Network
RF:	Random Forest
RMSE:	Root Mean Square Error

RRMSE:	Relative Root Mean Square Error
RSS	Residual Sum of Squares
RT:	Regression Tree
R^2:	Coefficient of Determination
SDE:	Standard Deviation of Error
S-MSE:	mean squared error of scaled value
SNA:	Social Network Analysis
SNN:	Shallow Neural Network
SSE:	Sum of Squared Errors
SVM:	Support Vector Machine
SVR:	Support Vector Regression
XGB:	Extreme Gradient Boosting Machine
%error:	Percentage error

List of figures

List of tables

This page is intentionally left blank.

List of equations

This page is intentionally left blank.

1. Introduction

The energy transition currently is and will be for the next years, an important research topic. The application of sustainable developments and improvements in energy self-sufficiency will continue to play an important role, not only in academics but also in business and politics. This research will contribute to the field of energy demand prediction of residential buildings by suggesting a new approach which combines a clustering technique and an ensembled regression tree machine learning (ML) model. Also this research will discuss applying energy sharing to increase energy self-sufficiency of a neighbourhood.

1.1 Research context

Similar to most of the countries around the world, the Netherlands has agreed upon pursuing efforts to limit the temperature increase to 1.5 ˚C above pre-industrial levels (United-Nations, 2015). To achieve the goals from this agreement, energy, in general, must be used more efficiently, and more energy must be generated from sustainable or renewable sources. Buildings account for 40% of the global energy demand and 30% of global CO2 emissions, and is therefore, one of the most focused fields for stimulating energy efficiency and sustainability (Costa, Keane, Torrens, & Corry, 2013) (Wang, Wang, Zeng, Srinivasan, & Ahrentzen, 2018). This has led to regulations and subsidies which stimulate investments in energy efficiency and sustainability measures, such as applying better insulation and PV-systems, for private and commercial buildings (Wiebes, Stimulering Duurzame Energieproductie, 2019). These regulations and subsidies did have results, as the number of wind turbines and photovoltaic panel (PV) systems have grown steadily the previous years. High penetration of energy conversion technologies based, on fluctuating renewable energy sources, shifts the power system from centralized to decentralized energy systems. Energy is no longer generated centrally only using large power generation stations, and top-down distributed from the high-voltage (HV) to the low-voltage (LV) grid where individuals can tap their desired demand (Junker, et al., 2018). In the decentralized systems, energy is generated, used, and distributed from numerous places across the grid. Currently, only a small portion of the total consumed energy is generated from sustainable or renewable sources. However, this is expected increase in the near future, rapidly. The report of CBS (2018) mentioned that the total share of renewable energy in the Netherlands was just 6.6% in 2017, but, the target for 2023 is to achieve 16% (CBS, 2018). This target demands a substantial growth of renewable energy generation, among which will be residential PV-installations. The trend report for PV applications of the International Energy Agency (IEA) state that the Netherlands has a total PV capacity of 2,983MW of which 853 MW is installed in the year 2017. Currently, three-quarter of the PV capacity is installed at private dwellings. To reach the renewable energy goals of 2023 there is room for 1GW of PV capacity installations per year (Masson & Kaizuka, 2018). This is also acknowledged by the Smart Grids European Technology platform, as they state that a significant fraction of the generation capacity in 2035 will be stochastic and/or intermittent (SmartGrids, 2012).

1.2 Problem definition

Most of the renewable energy sources, especially wind and solar energy, do not have a high energy density and are irregularly available. The utilisation of energy by the industry, dwellings, and the work place, have different demand patterns compared to the renewable energy generation patterns (Li & Chan, 2017). This is also true for PV-systems on residential buildings where the solar intensity is at its peak during the day where electricity demand, in general, is low. To use produced renewable energy effectively and to follow the demand, energy storage and smart energy dispatch technologies are necessary (Li & Chan, 2017). Although this is a clear statement, little applications of energy storage systems are currently used for residential buildings. Usage of electrical storage systems (ESS) are not yet established in the current energy system. More research should be conducted to expose the usability and feasibility of ESS. To properly use a storage system, the short-term detailed local energy demand of residential districts is necessary. There are many studies which suggest approaches to forecasting energy demand, but little of these are specific for short-term local residential buildings (Seyedzadeh, Rahimian, Glesk, & Roper, 2018) and almost none consider renewable energy generation on-site. As the adoption of renewable energy keeps increasing, also on the local residential district level, the need for energy storage and efficient energy distribution is increasing.

The two main problems addressed in this research are: (i) the lack of research towards local short-term residential energy demand forecast with renewables on-site, which is needed for having a proper electricity distribution system, and (ii) the absence of a general concept to improve the effective usage of PV- generated electricity and the energy self-sufficiency in a residential district by utilizing an ESS.

1.3 Research objective

This research aims to suggest an approach to accurately predict local residential electricity demand, with short time intervals, based on actual smart meter data. Currently, most researches, regarding energy demand prediction, have focused on enormous energy-consuming entities, such as entire cities and campuses. Also, the prediction horizon of these studies are often very long, such as aggregated usage of daily, monthly or even yearly (Seyedzadeh, Rahimian, Glesk, & Roper, 2018). Having access to detailed energy demand profiles have not been widely available before the upcoming of smart meters. The smart meter data has enabled a closer look into the energy demand patterns of individuals, and might reveal possibilities in improving energy self-sufficiency. Smart meter data is often considered as privacy-sensitive. Therefore, not many studies have been able to use such detailed energy consumption data of a large sample of households. The suggested approach should contain a method or combination of methods, which is innovative and helps to improve currently used models. Further, the potential of increasing self-sufficiency by using an ESS and by sharing electricity between dwellings should be determined. The goal is to accurately formulate a methodology to achieve growing self-sufficiency for a group of houses.

The research should show one of the many possible applications that analysing smart meter data makes possible. The research objectives will add knowledge to the research field of energy demand prediction since a rarely available dataset of smart meter data will be analysed in a way current researches have not done yet. The study should bring energy demand prediction a step closer to detailed individual household prediction, which can help efficiently micro-manage the smart grid and expose energy saving possibilities.

1.4 Research questions

To achieve the desired research objective, the following main research question should be answered:

How can self-sufficiency of a neighbourhood be increased by using electricity exchange and an electrical storage system, based on analysed smart meter data?

In order to answer this main research question, 3 sub-questions are drawn up:

SQ1. How can the houses with similar electricity demand profiles be clustered?

SQ2. What machine learning algorithm performs best in predicting the hourly electricity demand one day ahead, for clusters of dwellings?

SQ3. What is the expected direct and indirect electricity sharing potential between the clusters?

1.5 Research design

This research is divided into three stages, see Figure 1: (i) explorative research, (ii) validation and (iii) reporting. The following three sections will describe the content of each research stage.

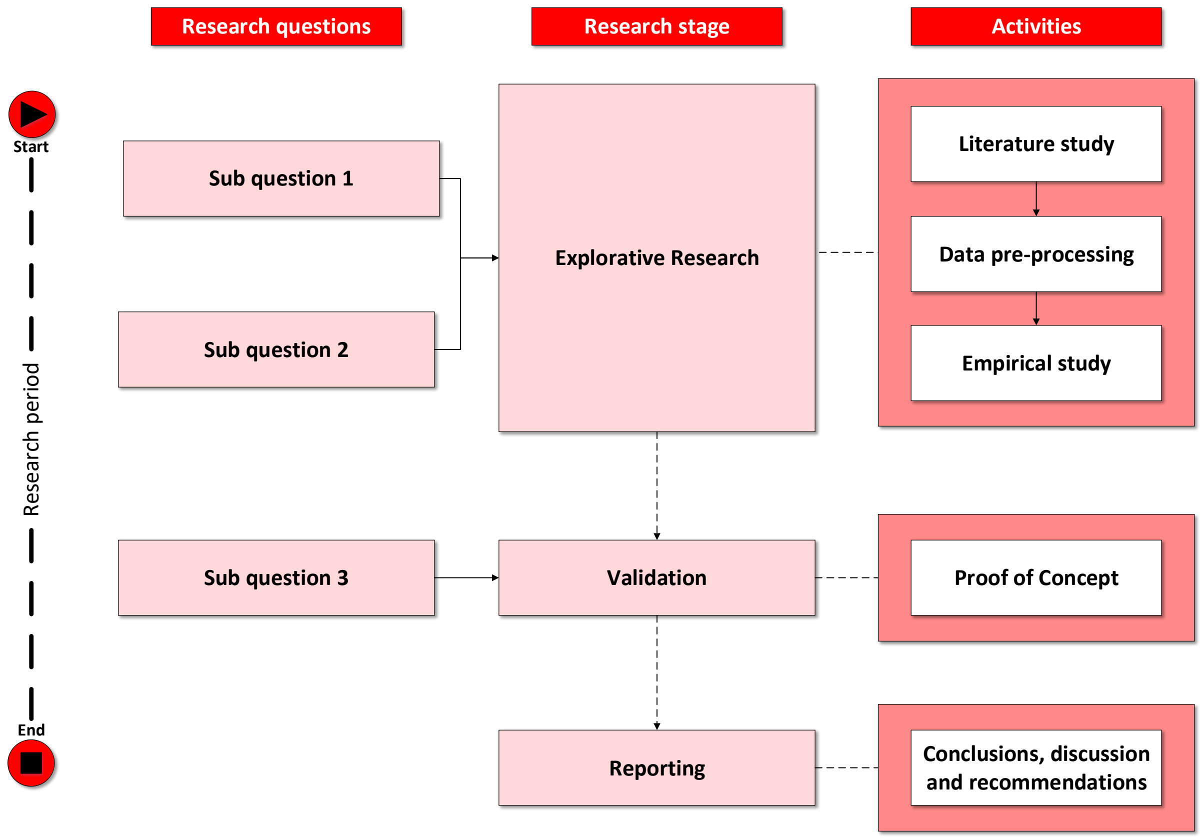


Figure 1 Research design

1.5.1 Explorative research

The explorative research stage consists of three parts; (i) literature study, (ii) data preparation and (iii) empirical study. In the literature study related scientific papers will be studied to see what methods and approaches are currently used in the field of research. The goal of this literature study is to explore the current state of the art techniques for energy demand forecasting. The data preparation part is a necessary step to make the received data useful for analysis and modelling. In this step the raw datasets will be transformed and processed so that they can be used for machine learning purposes. Activities such as removing outliers and replacing missing values will be performed. During the empirical study part, various machine learning algorithms will be tested to find empirical evidence for which algorithm performs best on the received data. After the explorative research section a well-grounded choice can be made to select a suiting model for predicting residential electricity demand.

1.5.2 Validation

In the validation section, the third research question will be answered. The direct and indirect sharing potential between houses will be determined by means of predicting the electricity demand for clusters. The suggested approach aims to predict the electricity demand for one day ahead and identify the direct and indirect sharing potential between the clusters. The validation part will compare the outcome of the prediction model with the actual values and discuss the results.

1.5.3 Reporting

During the reporting stage of the research the report will be finalized. This will contain answering individual research questions, drawing up conclusions, discussing the overall study, and giving recommendations for future related research.

1.6 Reading guide

In chapter two, a literature review of some close related scientific articles is given. Primarily energy demand prediction related articles will be covered throughout the literature review. Chapter three contains a thorough description of the used methodology. It includes explanation of the clustering technique and the prediction model as well as the method to determine the sharing potential. In chapter four, a description of the data and pre-processing methods are described. Chapter five presents the results of one validation day, and the overall results of the used approach. The conclusions, a critical discussion of the used methodology, and recommendations for future research are written down in chapter six.

2. Literature review

This literature review summarizes scientific publications related to energy demand prediction to see what methods could be implemented, what kind of data and performance indices are used, and what topics need further research. By reviewing these studies, it becomes clear what is already done in this field of research, and where additional work is desired. This literature review points out the necessity and relevance of this research. The covered studies are the inspiration for the used approach in this research.

2.1 Energy system transformation

The decentralized energy system demands smart solutions to be implemented, which enables achieving more energy efficiency and energy flexibility. Energy flexibility is the ability to adapt the energy profiles without threatening technical and comfort constraints (Reynders, et al., 2018). Therefore, to achieve a more efficient and flexible energy system, the widely known trais energetica method was adapted to five steps, including user demand and behaviour and the effect of energy exchange and storage systems (HaskoningDHV, 2018), see Figure 2. The newly added step 1, which is "Design according to user demand and behaviour", helps in exploring the required energy flexibility without compromising the user comfort. On the other hand, step 4, "energy exchange and storage systems", helps in overcoming the grid-related bottlenecks. Even though the method is fully established, further research and exploration are still needed to identify the potential of this methodology in different case studies. Therefore, this research will contribute to the knowledge and applications of step-four by identifying the potential for energy sharing and storing.

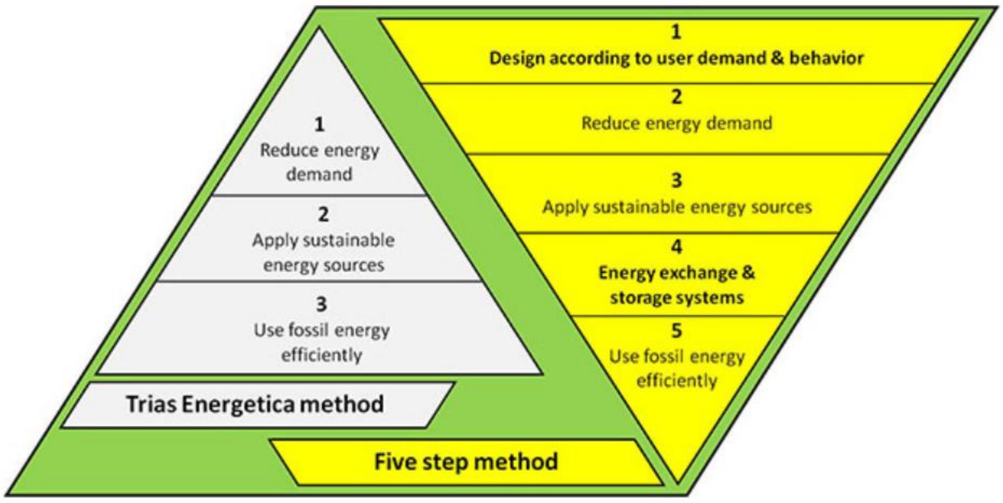

1
Reduce energy demand
2
Apply sustainable energy sources
3
Use fossil energy efficiently
Trias Energetica method
1
Design according to user demand & behavior
2
Reduce energy demand
3
Apply sustainable energy sources
4
Energy exchange & storage systems
5
Use fossil energy efficiently
Five step method

Figure 2 Five step method (HaskoningDHV, 2018)

As step four includes storage systems, a short declaration on the necessity of storage systems in the electricity grid will be given. There is a broad acknowledgment that, due to the variability of renewable energy sources, energy storage systems must be implemented to make a complete energy transaction possible. Electrical storage systems can contribute to a variety of manners for efficient energy usage. The electrical storage system enables maximization of PV systems and shared usage of electricity generated by PV systems. When the electricity is not used by a household with a PV system the electricity can be stored in the ESS, during daytime when demand is low, and consumed, by the household itself or another household in the neighbourhood, when demand is high. The decentralisation of the energy network and the usage of ESS enables consumers to shape their power demand actively. This possibly will increase the energy demand flexibility of a smart neighbourhood.

Other examples of benefits, dependent on the capacity of the storage system and the time the electricity is stored, described by Barton and Infield (2004) are:

- Spinning reserve – clouds on PV panels and wind power smoothing
- Standing reserve – peak shaving smoothing of loads
- Smoothing of weather effects – PV, wind, small hydro
- Voltage and frequency control

The research of Barton and Infield (2004) show that the electrical storage systems can increase the absorption of renewable energy by 10 -25%, dependent on the storage time, without grid reinforcement. Some scenarios used in the research are proven to be economically worthwhile (Barton & Infield, 2004). One possible storage system is the lithium-ion battery. Some studies have determined the financial feasibility of such electrical storage systems with different subsidy scenarios (Kantor, Rowlands, Parker, & Lazowski, 2015) (Cucchiella, D'Adamo, & Gastaldi, 2017). The approach of these studies suggests that with the current practices, ESSs are, currently not feasible without governmental support in these specific cases. However, as mentioned earlier, the electricity grid, as well as the regulations regarding feed-in electricity, are changing rapidly, which makes storage becoming more a feasible additive to the network.

In order to operate an ESS efficiently, accurate short-term electricity load forecast is desired (Lahouar & Slama, 2015). Energy load forecasting or energy demand prediction is the speculation and prediction of what the power demand for a certain period in the future will be, using historical data, and its influencing factors and by applying mathematical algorithms (Huo, Shi, & Chang, 2016). Energy demand prediction has always been acknowledged to be a difficult task. The dependency on weather conditions, building characteristics, operation of sub-level components (e.g. HVAC-system and appliances), occupancy and user behaviour, makes it a complex problem (Zhao & Magoulès, 2012). However, the high penetration levels of intermittent resources in the grid, such as wind and solar energy, increases the degree of uncertainty even more, due to their non-regular behaviour. This is an incentive to extend the knowledge in this field of research (Lahouar & Slama, 2015) (Huo, Shi, & Chang, 2016). The technological advancement of communication infrastructures, mathematical modelling techniques, and numerical simulation environments are promising developments for the integration of short-term electricity load forecasting into the existing grids. This is also true for storage capacity within the energy network, which can be achieved by intelligent demand side management (DSM) (Jurado, Peralta, Nebot, Mugica, & Cortez, 2013). Research has proven that empirical ML modelling can provide noticeable prediction results and outperform engineering-based building energy modelling when the algorithm and settings are appropriately chosen (Wang, Wang, Zeng, Srinivasan, & Ahrentzen, 2018). ML is generally referred to as a computer algorithm that learns from existing data (Seyedzadeh, Rahimian, Glesk, & Roper, 2018). The focus of this study is therefore towards the application of ML models to forecast short-term electricity demand.

2.2 Energy demand prediction

2.2.1 Essence of energy demand forecasting

There are many arguments on why improving the accuracy of energy demand modelling is beneficial. Better prediction of energy demand can lead to a reduction of monitoring expenses, initial cost of hardware components, and long-term maintenance costs in the future grids (Kuo & Huang, 2018). Accurate forecasting of energy consumption can help to determine the required storage size, to delay and postpone energy consumption. It can be used at early design stages of renewable energy systems to see its impact. It can also help in the DSM, to forecast the likely future development of electricity demand (Rodrigues, Cardeira, & Calado, 2014). Wang et al. (2018) state that building energy demand prediction is becoming more significant for improving efficiency due to its essential role for implementing energy efficiency measures such as; demand response control, system fault detection and diagnosis, building energy benchmarking and building system measurement and verification (Wang, Wang, Zeng, Srinivasan, & Ahrentzen, 2018). Mocanu et al. (2016) state that the future grid needs a system that can monitor, predict, schedule, learn, and make decisions regarding local energy consumption. Ryu et al. (2016) state that short term energy load forecasting is becoming increasingly important. A wide variety of applications is mentioned for accurate prediction models such as demand response, targeted dynamic pricing, load monitoring, energy storage operation and peak load reduction (Ryu, Noh, & Kim, 2016) (Lahouar & Slama, 2015). Other authors explain that accurate forecasting is a vital issue to support individual and organisational decision making (Jurado, Peralta, Nebot, Mugica, & Cortez, 2013). Predicting energy demand should not only be done on an aggregated level but also individual building level so that the distribution of locally generated energy can be distributed based on the local demand. Also, the decomposition of energy demand, locally in short-term time intervals, helps to analyse energy demand patterns and give insight in potential energy conservation targets (Mocanu, Nguyen, Gibescu, & Kling, 2016). These are reasons why much effort is put into investigating different models and approaches to improve energy demand prediction.

2.2.2 Energy demand forecasting over the years.

Energy demand forecasting has been studied intensively since the 1950s (Jurado, Peralta, Nebot, Mugica, & Cortez, 2013). Early researches regarding demand forecasting, often used methods such as autoregressive integrated moving average (ARIMA), exponential smoothing, non-parametric regression and Kalman filter (Taylor, Menezes, & McSharry, 2016) (Ryu, Noh, & Kim, 2016). In the 1980s the Kalman filter was especially attractive an model to forecast demand, due to its computational efficiency. By the time computing power increased, researchers were able to explore the complex load data better. This led to rule-based and fuzzy logic expert systems to model the complexity in the load data, using domain knowledge. Despite the fact that these approaches were promising, they relied on rules that were extracted from experts' and operators' individual experiences, which are subject to inconsistencies and therefore not reliable (Taylor, Menezes, & McSharry, 2016). Another promising approach introduced to help the increase accuracy and reliability of demand forecasting is neural networks. Artificial neural networks (ANN) made experimenting with models that can identify complex nonlinear relationships in data and predict future behaviour. These models are theoretically weak and need rich data since they learn through examples which consist of input features and output. This learning through examples enables the potential to discover a far greater range of relationships compared to models with pre-defined format (Taylor, Menezes, & McSharry, 2016). Artificial neural networks are therefore broadly used and have received lots, if not most, of attention and interest in scientific publications regarding energy demand prediction since it was introduced in the building services sector in the 1990s (Rodrigues, Cardeira, & Calado, 2014). More recent studies have used other ML techniques such as Support Vector Machines (SVM) and ensembled trees and showed that these models also achieve highly accurate performance results (Zhao & Magoulès, 2012).

2.2.3 Commonly used approaches and methods

As the essence of accurate energy demand forecasting has become clear, researches have started exploring and optimizing all sorts of different approaches. Many studies have compared or suggested different models to predict buildings' energy demand. For some of these studies, the most important fragments, such as the used data types, number, characteristics of prediction variables, performance indicators, the results, and conclusions, are summarized.

An important factor for the approach of energy demand forecasting is the forecasting horizon and corresponding time intervals. In general, a shorter prediction horizon (days, hours, sub-hourly) is harder to predict than mediocre (weeks or months) and long-time (many months or a year) horizons, since short intervals are more susceptible for statistical variations. Besides the time intervals, the prediction entity is an important factor, for the same reason, as small energy-consuming entities are more sensitive for little variations. Please take this in mind when reading this review.

Kontokosta and Tull (2018) used a yearly prediction horizon, which is considered as long. The authors used three ML algorithms, which are ordinary least squares (OLS) regression, RF, and SVM to forecast annual building, district and city-scale energy demand of New York City, USA. The authors used a combination of actual building energy consumption and detailed property and building-level attributes, such as floor area, use type, and building year. This information is collected from three sources: (i) Local Law 84 energy disclosure provides building energy use information, building occupancy and physical characteristics of 20,000 buildings, (ii) city-scale PLUTO provides parcel-level physical characteristics, zoning and use type data of all 1.1 million buildings in New York, (iii) zip code energy data set provides aggregated annual energy consumption data of all 176 zip codes in New York. All types of buildings are included in the data set, residential, commercial, industrial, etc. For the RF and SVM, the hyperparameter settings were optimized by creating a grid of different parameters and stepwise testing all the different combinations. The best performing combination was eventually chosen to use for the actual prediction. This optimization step is conducted for both electricity and natural gas consumption. Fivefold cross-validation is used to overcome the possibility of overfitting. The model is validated at the building and zip code level using actual consumption data of 2014. The authors chose to use the mean absolute error (MAE) and the mean log accuracy ratio (mean-LAR) to compare the model accuracies. The feature importance is dependent on the scale-level, building or zip code level, and on the energy source; electricity or natural gas. In general, building use, size and layout serve as the most important predictors. The authors conclude that, for the OLS model, using more variables decreases the model accuracy. For the RF and SVM, the first six features provide the vast majority of the prediction ability. Overall, the authors conclude that there is little difference in the prediction ability and accuracy of the three used machine learning models. When looking on the city and zip code scale, the simple OLS model performs best, on the building level prediction the SVM has the lowest MAE. Further, it is concluded that, given the used models, predicting natural gas consumption is harder than electricity, for both building level and zip code level scale. The authors state that the results have proven that there is a need for higher data transparency and data access from utility companies, as it is a valuable resource for helping cities to plan and evaluate sustainability and carbon reduction strategies (Kontokosta & Tull, 2017).

Robinson et al. (2017) also compared multiple machine learning models to predict annual electricity demand. The prediction entities were commercial buildings. In total 14 different machine learning models are tested and compared based on the R^2 and the Mean Absolute Error (MAE). They used data from the CBECS, published every five year by the U.S. Energy Information Administration (EIA). This dataset contains 6720 rows (buildings) of data which are representative for 5.6 million commercial buildings in North America. The data is gathered by means of a questionnaire to building owners. The models are based on five building and weather features; (i) number of floors, (ii) square feet, (iii) heating degree days, (iv) cooling degree days, and (v) principal building activity. This small amount of commonly accessible features makes applying this approach convenient, elsewhere. They conclude that gradient boosting regression models perform the best with an R^2 score of 0.82. They note that they used the default settings of the models and did not optimize the hyperparameter settings (Robinson, et al., 2017).

Li et al. (2010) predicted the annual energy consumption of residential buildings with a SVM and multiple ANNs; general regression neural network (GRNN), back-propagation neural network (BPNN), and radial basis function neural network (RBFNN). The models use 16 technical building features such as; building size, window wall ratio, heat transfer coefficient, as well as annual electricity consumption of the buildings, as prediction features. Stepwise searching is used to select suitable parameter settings for each model. For the neural network models, the number of neurons in the hidden layers drastically impacts the results and is, therefore, an important parameter to optimize. The parameters for the SVM are optimized by using genetic algorithms. The models are compared based on the relative errors, MSE, and root mean square error (RMSE). The study uses 50 houses for training and nine houses for testing. They conclude that the SVM and the GRNN perform significantly better than the BPNN and RBFNN given the nine testing samples (Li, Ren, & Meng, 2010).

A Net-zero energy residential test facility is used to test the application of a regression (OLS) model to predict its energy performance by Kneiffel and Webb (2016). The test facility contains a PV system for electricity production and has data acquisition and control system to collect data and monitor the performance. A heat pump and a thermal solar system are used for space heating and domestic hot water; no natural gas connection is available. For lightning and insulation of walls and windows, high performing materials are used. The installed instruments measure weather data at the test facility such as temperature, humidity and solar insulation, electricity consumption (of the building as a whole, as well as system-specific values), electricity production. These variables are collected every 3 or 60 seconds, dependent on the specific measurement. These observations collected one year and aggregated to daily average values for the analysis. The prediction is separately executed for the consumption and the production of electricity. The first two weeks of every month were selected to be the training data, which contains 140 daily consumption and production values. The RMSE and R^2 values are used as performance indicators for the prediction model. Also an analysis of variance (ANOVA) is conducted to determine the significance of the models and their coefficients. The proposed OLS models are compared with broadly used, physics-based, energy performance indication models. The authors found that the production model has an extremely strong linear correlation, while the consumption model shows a more unexplained variation. The suggested OLS model performs better than the other three tested, engineering-based models. The higher accuracy is, according to the authors, mainly achieved through the actual in-situ data input of the test facility, where the engineering models work with the specifications of all materials and appliances (Kneiffel & Webb, 2016). The study does not use actual occupancy behaviour, which is a major shortcoming although it is generally recognized that this plays an important role in the complexity of consumption patterns (Arregi & Garay, 2017), by means of opening windows, usage of appliances and changing thermostat settings.

A study by Xu et al. (2019) proposed an integrated social network analysis (SNA) and ANN to predict multi-building energy use. Specifically they try to predict the energy use index (EUI), in kWh per square meter. The SNA is used to determine reference buildings, buildings with similar energy use patterns, and identify correlations between the total energy usage of a building and that of a reference building and non-reference building. The data used in this approach consists of three years of monthly energy use of 17 buildings on the Southeast University campus in Nanjing, China. The different building types on the campus are; office, educational, laboratory and residential. Missing values and erroneous data is replaced by interpolating according to Lagrange polynomials. Features used in the prediction model consist of historical energy use, change in energy use and building characteristics such as materials, physical dimensions and building year. The model was evaluated and compared with a regular ANN based three indices; MAE, MAPE, RMSE. The study concludes that the SNA-ANN model predicts the monthly energy use for the building groups, offices, laboratory and education around 90% accurate and residential dwellings 83.32% accurate. Dependent on the network strength between the reference buildings' EUI and the total buildings' EUI and the standard deviation this is outperforming the regular ANN (Xu, Wang, Hong, & Chen, 2019).

These long ranging prediction horizons are often very accurate and can be used for more general capacity planning. There are also studies which focus on mediocre energy demand prediction. Dong et al. (2015) have examined the feasibility and applicability of SVM in forecasting building loads monthly. They have used weather data such as monthly mean outdoor temperature, relative humidity, and global solar radiation. The approach is applied to four, randomly picked, buildings from a business district in Singapore. The used data was collected from an extended survey started in 1996, where building owners have collected monthly electricity consumption from the main meters. Data from October 1996 through October 1998 and the year 2000 were used as training data, and 2001 was used as a test set.

Different parameter settings are tested to find the best suitable model for the problem. To select the best model for each building, four performance indicators are used, the MSE, mean squared error of scaled value (S-MSE), cv-RMSE, and the percentage error (%error). The best performing models have all very accurate results with cv-RMSE of less than 3% and %errors of under 4%, which is excellent (Reddy, et al., 1997). Dong et al. (2015) compare their results with other researches regarding demand forecasting of commercial buildings and conclude that their results are superior with the lowest errors and highest prediction accuracy. The authors compared their research with studies who used other, daily or hourly instead of monthly, time intervals, which makes the comparison somewhat unfair. Nevertheless, the suggested SVM does prove to have a very high performance in the researched case. Advantages of SVM are the little parameters that have to be optimized compared to genetic programming or ANNs. This study has only optimized two parameters for the prediction model. A disadvantage of SVM is the large computation time when applied to large-size problems. This disadvantage was not relevant in this study since little data, monthly consumption of four buildings was used. The authors conclude that, due to the good results, future focus should be on short-term load data exploration (Dong, Cao, & Lee, 2005).

Another mediocre prediction time horizon study is conducted by Tso and Yau (2005), who compared multiple machine learning models for prediction total weekly electricity consumption of individual dwellings in Hong Kong. Commonly used approaches are selected to be compared; Stepwise regression, Decision Tree (DT) and ANN (stepwise regression, and intercept regression were also considered). The comparison is made, based on two seasonal cases, winter and summer. For each case data for prediction variables were collected, by an extensive survey of over 1000 households. Detailed information about the ownership and usage patterns of appliances were collected and monitored. The monitoring of the appliance usage of the households has pointed out that the air-conditioning consumes on average 59% of the total electricity in a typical household in Hong Kong. Also, the housing type, household characteristics data is used as prediction variables. The square root of average square error (RASE) is used as a performance indicator to compare the three models. The authors conclude that the DT slightly outperforms the other more complicated methods based on the RASE score. The ANN performs worst, although the differences, based on the RASE score are very small. For all the models, three variables were proven to have a significant impact on the prediction; House (flat) size, Number of household members and ownership of air-conditioning, are the most important in the summer period. The winter period indicates, housing type, number of household members and ownership of electrical water heater, as the most important prediction variables, which makes sense. The authors remark that the inclusion of meteorological features should improve the model fitting results (Tso & Yau, 2005).

The short-term prediction horizon, with aggregated data of a maximum of one day, is covered in scientific publications. Biwas et al. (2016) conducted a case study to assess the capabilities and possible implementations of two ANN based prediction models, a 'Levenberg-Marquardt' and an 'Output Weight Optimization' variant. The MATLAB neural network toolbox was used to run the models. Two specially constructed research and demonstration dwellings on the UT Tyler campus were used as test facilities. These houses are designed to serve as realistic test facilities for developing and demonstrating new technologies related to energy efficiency. Due to the many energy efficiency measures applicate at these dwellings, the energy usage is approximately half of that of a similar regular dwelling. The test buildings only have an electricity connection and heating is provided by means of a HP. Every five minutes the energy consumption and weather conditions of three months (72 days), was collected at the test dwellings. The houses are however not occupied, which is a major shortcoming of the test facilities, since user behaviour is known to have a significant impact on the energy consumption profiles. Both ANN models have identical setup and use similar prediction features, number of days, temperature and solar radiation. 70% of the data is used for training and the remaining 30% for testing. Each model is used to predict the total daily energy consumption of the dwelling and the hourly electricity usage of the HP. The R^2 was used as a performance indicator to compare two models. The Levenberg-Marquardt model performs slightly better than the Output Weight Optimisation model. The authors mention that the difference is not significant due to the small number of data points used. Further, it is concluded that the prediction for the electricity consumption of the HP is more accurate than the total energy consumption (Biwas, Robinson, & Fumo, 2016). Although detailed, five-minute interval energy consumption data is available, the authors chose to test the models on aggregated daily consumption data.

Most of the studies which suggest or test the models for short-term load forecasting focus on large energy consuming entities, such as Darabellay and Slama (2000), who tested different models to predict the electricity demand of the entire Czech Republic. These authors investigated the linear and nonlinear correlation of electric load time series profiles. This is tested by predicting the short term (four different time horizons of 1h, 12h, 24h and 36h) electric load by means of a nonlinear model, ANN, and a linear model, ARMAX. Two years of hourly intervals of electric load of the Czech Republic is collected. From this data, some periodic components have been visually identified; there is an annual, weekly and daily cycle visible in the load profiles. The, the nature of the correlation between time and electricity load is investigated. The autocorrelations of the electric load over time is determined, by observing the differences in electric load value over a time interval. The authors found that the nonlinear correlations were weak, however, they were not sure whether to completely neglect these correlations in a predictive model. The data is split into two sets, working days and holidays. For the prediction, only the working days are used. One year of data is used for training, and one year for testing. For the ANN model, the Matlab neural network toolbox was used, and the linear ARMAX model is generated by means of genetic programming. For the daily forecasting, average daily outdoor temperature, which has a linear correlation with electricity consumption in the Czech Republic (Darbellay & Slama, 2000), is used as the prediction variable. The normalized mean square error (NMSE), MAPE and the maximum absolute percentage error (maxAPE) were used to determine the model accuracy. It is concluded that the prediction abilities of the linear model are slightly superior to that of the nonlinear model. This is in line with the earlier conclusions regarding the linearity of the autocorrelations. The main point of the research is the advice to check whether the problem is indeed nonlinear before embarking some complex nonlinear model. The authors have put more effort in optimizing the ARMAX model compared to the ANN, and mention that experimenting further with the ANN model would have resulted in better results, yet it would have cost them more time than building the ARMAX model (Darbellay & Slama, 2000).

From the available l iterature on prediction models with hourly intervals, note that most of the studies focus here on large energy consuming entities. These studies towards short-term prediction models are chosen because they are most relevant for this study.

Lahouar and Slama (2015) used a random forest for day-ahead electric load forecast. A large dataset, of electricity and gas is used with half-hourly intervals from 1 January 2009 to 31 August 2014 from the Tunisian power system. As a comparison, hourly data from Pennsylvania-New Jersey-Maryland Interconnection, USA, is also used. Data preparation consisted of aggregating the data to hourly values and, removing missing values, by replacing them with previous ones. No normalisation is applied, the data consists of real ranges. Special days, such as public holidays, are intentionally kept within the dataset so that the model could learn their behaviour. The training set for the prediction model consists of all the available data up until the prediction day. Model inputs used for the RF concern of autoregressive features, two previous days at the same hour as well as morning and evening peaks of the previous day. Further, external factors, month number, day type, maximum and minimum temperature of the predicted day are used as features. Using weather data of the predicted day are not possible when predicting a day ahead. Therefore, weather forecast data is used. Bad weather forecast entails misbehaviour of the prediction. The RF is compared with extensively used ML models, ANN and SVM. Also, a persistence model, where the prediction is exactly the same load demand as the previous day, is added as a reference. For the ANN and SVM no extensive hyperparameter tuning is conducted, only slight manual adjustments for optimisation. For each season in 2013, a week of hourly electric load is predicted by the models, for the USA data as well as the Tunisia data. A separate prediction is run for the special days, with feature selection by an expert on the cultural background of these days. The combination between the RF and expert input is able to capture complex load behaviour and can solve special cases that are specific to cultural or religious events, by means of appropriate inputs. The RF performs overall better than the ANN. Dependent on the season and weekday, the RF performs equal, better or worse than the SVM, on both the Tunisian and the USA set (Lahouar & Slama, 2015).

Another study compared a new deep neural network algorithm called DeepEnergy with a SVM, a Random Forest (RF), a Multilayer Perceptron (MLP) and a Long Short Term Memory neural network (LSTM). A dataset from the U.S. district public consumption and a dataset of electric load provided by the Electric Reliability Council of Texas was used. This dataset represents roughly 90 percent of the electric load of Texas. In this study, the models trained on 7 days of hourly energy loads (168 hours) to predict for 3 days, 72 hours, for the whole state of Texas. The input for these models is purely the past energy loads. This study compares the models with Mean Absolute Percentage Error (MAPE) and Cumulative Variation of Root Mean Square Error (CV-RMSE). The authors mention that the RF does outperform the decision tree and the SVM, which proves that the model ensemble solution is effective in the energy demand forecasting. On both performance indices, the DeepEnergy model outperforms the other models, closely followed by the RF and the LSTM (Kuo & Huang, 2018). It is notable that the amount of data, used in this study is fewer than comparable studies. The large amount of electricity consumption appliances creates relatively smooth graphs, which eases the prediction.

Taylor et al. (2016) have compared six algorithms for forecasting electricity demand; autoregressive moving average (ARMA), linear regression (OLS) with principal component analysis (PCA), exponential smoothing, ANN and two simple benchmark methods. In the study, two datasets are used, one from Rio, Brasil, which contains hourly electricity consumption of the whole city, and one from England and Wales which contains the total electricity used every 30 minutes. They tested all six models on both datasets and compared the results based on the MAPE. They conclude that the exponential smoothing model performs the best overall, followed by the PCA linear regressing model. This points out that more simple methods, which requires little domain knowledge, can outperform sophisticated alternatives, in this case ANN and ARMA (Taylor, Menezes, & McSharry, 2016). A notable remark Taylor et al. (2016) make, is that they used a ANN approach very similar to the one Darbellay and Salma (2000) used, but got different results (Darbellay & Slama, 2000) (Taylor, Menezes, & McSharry, 2016). Taylor's achieved accuracy is significantly lower, which again indicates that the data, on which the training and prediction is executed, plays an important role in the suitability and quality of the model.

The study of Huo et al. (2016) compared SVM and RF for short-term electric load forecasting. For prediction variables, the month number, weekday indicator, holiday indicator, minimum temperature, maximum temperature and previous days' load were used. The MAPE was used as the performance indicator to compare the models. Hourly electricity demand data from different sources is used, from multiple cities worldwide. This enables investigating the impact of the nature of the data on the different models. The programs were run in Matlab bridged with R environment, by Matlab-R link with SVM and R packages installed. For both the SVM and the RF, the parameters are optimized by step wise changes. The authors conclude that both models are excellent for short-term load forecasting. Overall, no significant differences could be observed between the two prediction models. The performance of these models is dependent on the parameter settings, the data and even seasons. Parameter settings are more important for the model fitting of SVM than of the RF (Huo, Shi, & Chang, 2016).

The study of Juardo et al. (2013) tested different ML techniques to for short-term electric load forecasting; RF, ANN, SVM, and Fuzzy Inductive Reasoning. Data from three locations are used, the entire campus of University of Catalonia, and two office buildings in Barcalona of 200m^2 and 50m^2. For the campus data, a complete year of hourly electricity consumption data is collected. For the office buildings, approximately six months of hourly data is collected. The input for the models only consists of historical energy load behaviour, both electricity and gas. Validation of the models for the campus data, is based on four days of data in different seasons. For the office buildings, three days are used for validating the model. The training set consists of all the available data up until the prediction day. The number of data points do differ per validation day. The normalized root mean square error (NMSE) is used to compare the different models with each other. The authors conclude that the Fuzzy Inductive Reasoning model has the best performance based on the NMSE, closely followed by the RF.

The SVM and ANN show less accurate results, especially compared to the computational efforts of the models, indicating that these models are not the best solution for the used data. The RF and Fuzzy Inductive Reasoning model can handle sudden changes. Further, it is concluded that the Fuzzy Inductive Reasoning model, RF and ANN are computationally efficient enough to be used as real-time prediction models. The authors recommend that the Fuzzy Inductive Reasoning model and the RF should be studied and used more in-depth for short-term electricity load forecasting (Jurado, Peralta, Nebot, Mugica, & Cortez, 2013).

Another study which has focussed on predicting hourly energy demand is that of Wang et al. (2018). The authors of this study have used a random forest to predict the hourly electricity demand for two institutional buildings on the campus of the University of Florida, with surfaces of 47.270 and 72.520 square feet. In total, eleven prediction variables were used in the prediction model. Meteorological prediction features such as; temperature, humidity, wind speed, rainfall and solar intensity, and time related data such as indicators for days of the week were used. Also they used occupancy data of the buildings. The occupancy was estimated based on the operation and class schedule of the buildings. A data transformation process was performed to determine the hourly occupancy of each building. This is something which not many other researches have implemented in energy prediction modelling, probably because such specific information is hard to get. The output, or dependent variable, is the hourly building level electricity usage. The research has used one whole year of hourly data, consisting of 8760 time stamps. After removing observations with missing values for both buildings, an acceptable 99% and 95% of the observations were preserved for the prediction model. Two approaches were used to test the applicability of RF for energy demand prediction: (i) complete year as a data set, (ii) months as separate sets. In the latest mentioned approach they tested three months, February, July and October. The monthly set contain significantly less data points. However, the total trend in the dataset will be less variable when only a month is used, due to the seasons they are in. They used 80% of the data for training and 20% for testing, by randomly splitting the data into two sets, for both the yearly and the monthly sets. They compared the RF with a single Regression Tree (RT) and a Support Vector Regression model (SVR) to see the performance differences. They compared the models based on the R^2, RMSE, Performance Index (PI) and MAPE. For both buildings, the RF outperforms the other two models, in training and testing, on all the performance indicators. Regarding the variable importance, the authors conclude that for the yearly models, similar variable importance results are observed. The monthly models did differ more, which indicates that certain variables play a more important role depending on the season. The occupancy feature, which is not much used in other researches, did have big impact on the prediction results. Based on this result, it is recommended in this study that future researches should try to implement more user behaviour in the field of energy demand prediction (Wang, Wang, Zeng, Srinivasan, & Ahrentzen, 2018).

Fan et al. (2017) used a deep learning approach to predict 24 hour ahead cooling load of a large institutional building in Hong Kong. The authors describe deep learning as a collection of machine learning algorithms which are powerful in revealing nonlinear and complex patterns in big data. A full year of 30-minute interval data is collected for, temperature, humidity, supply and return chilled water temperature, and the flow rate of the chilled water temperature. The cooling load, which is the dependent variable in this study, is calculated based on the three latest mentioned variables. Also, time data, e.g. month, day type, and hour of the day, is used as prediction variable. Lastly, the previous 24 hour of cooling load is used as a prediction variable, adding 48 more features, due to 30 minute interval. Seven different models were used in the same way to see the performance differences namely, Multi Linear Regression (MLR), RF, SVM, Elastic Net (ELN), Gradient Boositing Machine (GBM), Extreme Gradient Boosting Machinen (XGB), and ANN. The MAE, RMSE and cv-RMSE, are used as performance indices to compare the model performances. The data set is split into a training, testing and validation set, with proportions of 70%, 15% and 15% respectively. Stepwise optimisation for the hyperparameter settings is used for optimisation. The MLR and ELN, which assume linearity, have the poorest performance. The XGB has overall the best performance followed by the other non-linear models. The authors mention that there are large differences between the performance with basic, default, settings and optimized settings. The authors conclude that using deep learning approach can provide accurate and reliable 24 hour ahead building cooling load prediction (Fan, Xiao, & Zhao, 2017).

Li et al. (2009) have compared SVM and different ANNs to predict the hourly cooling load of an office building in Guangzhou, China. In this study the Matlab 7.0 Neural Network toolbox is used to train and develop models. Features such as meteorological data, relative humidity, temperature and solar radiation as well as historical data of cooling load are used. In total six months of hourly data is collected for these features. Five months are used for training, and one for validation. They compared the models based on the Mean Relative Error (MRE) and the Root Mean Square Error (RMSE). The SVM performs slightly better than the three ANN models, but all models have sufficient accuracy for engineering purposes (Li, Meng, Cai, Yoshino, & Mochida, 2009).

Ryu et al. (2016) used two different Deep Neural Network (DNN) approaches that are suggested to identify the applicability, (i) the restricted Boltzmann machine, and (ii) rectified linear unit DNN. Hourly electricity consumption data provided by Korea Electric Power Corporation. For eight different industrial categories, (Retail; R&D; Healthcare; Networking business; Vehicle and Trailer manufacturing; Electronic component and Computer manufacturing; and other manufacturing) five consumers are randomly selected. Also, weather data, such as cloud coverage, solar radiation and temperature, together with indicators for seasons, month, and date are used as prediction variables. All data, including the hourly consumption data, is normalized (when numerical values are considered), cleaned and restructured before used in the prediction models. The focus for the prediction is only on business days. For each consumer, 750 days of hourly data is gathered for three years without holidays. The MAPE and the RRMSE are used as performance indicators. To overcome overfitting, k-fold cross-validation is used to determining the stopping criteria for the training models. The two suggested DNNs are compared with Shallow Neural Network (SNN), ARIMA, and DSHW to verify the DNNs performance of forecasting energy demand for individual users. This comparison is executed in three cases; single load types various load types, and aggregated load types. This rather extensive experiment shows that the DNN based models can be trained well, with up to three years of customer load data for predicting 24-hour load profiles day-ahead, without overfitting. Both suggested DNN based models, significantly outperform the other tested models, based on the MAPE and RRMSE (Ryu, Noh, & Kim, 2016).

Publications with short term energy demand prediction models for individual households are not present in abundance. One of the fewer studies which try to forecast hourly electricity demand for individual households is that of Rodrigues et al. (2014). They have used an ANN to predict households' electricity usage up to three days ahead. The authors did deliberately chose not to use weather data, such as temperature, as a prediction variable, to prove the possibility of obtaining good results without using extended amount of data. Still, the model uses 16 inputs such as; available electric appliances, apartment area and number of occupants. These input variables are very user specific and rarely used in other researches. They used 6 weeks of hourly electricity consumption data of 93 households in Lisbon, Portugal. Two-thirds of the data is used for training and testing, and one third is used for validation. The R^2, MAPE and the standard deviation of error (SDE) are used to indicate the models' performance. The Levenburg-Marquardt algorithm was used to simplify and reduce the computational effort to run the ANN, by means of pruning. This algorithm determines which units are not necessary for the solution and removes them making the model more efficient. The conclusion of this study is that their ANN can accurately forecast daily, average and maximum, and hourly energy consumption (Rodrigues, Cardeira, & Calado, 2014).

Houimli et al. (2019) used a ANN to forecast the half hourly electric load demand of Tunisia. Nine years, 2000 to 2008, of half hourly electric load demand is used in the prediction model. The first eight years are used for the training and validation of the model, and the last year is used for testing. As prediction variables, the past days' electricity profile, 48 inputs, together with meteorological data, minimum and maximum temperatures, and calendar variables, such as type of day, week, month, year, are used. All the used data is normalized, which is essential for a good ANN performance (Houimli, Zmami, & Ben-Salha, 2019). A pattern search optimisation algorithm is used to determine the best number of hidden layers and the number of neurons in each layer. The proposed ANN, with Levenberg-Marquardt algorithm, is compared with two other ANN model, the resilient backpropagation and conjugate gradient. For the evaluation of the model several performance indicator are uses; the MAE, MPE, MSE, MAPE and RMSE. Five average day profiles, Monday, Thursday, Friday, Saturday, Sunday, from 2008 are used in the final testing of the model. The proposed Levenberg-Marquardt ANN outperforms the other two tested models in all cases (Houimli, Zmami, & Ben-Salha, 2019). The usage of testing on average day profiles of an entire year makes the prediction less impressive. These profiles are completely smoothed out and no sudden changes, spikes or whatsoever appear in these profiles.

Table 1 summarizes the global approaches from some of the, energy demand prediction related scientific articles, covered in this literature review.

Table 1 Literature review summary

Authors	**Models used**	**Performance indicators**	**Time interval of prediction**	**Prediction entity**	**Data used**
(Kontokosta & Tull, 2017)	OLS, SVM, RF	MAE, Mean-LAR	Annual	Building, zip code and city level	1 year
(Robinson, et al., 2017)	Multiple Tree, OLS and SVM based models	MAE, R2	Annual	Commercial buildings (6000+)	1 year
(Li, Ren, & Meng, 2010)	ANN, SVM	MSE, RMSE	Annual	Residential buildings (59)	1 year
(Kneiffel & Webb, 2016)	OLS	R2, RMSE	Annual	Individual residential building (1)	140 days
(Xu, Wang, Hong, & Chen, 2019)	SNA-ANN	MAE, MAPE, RMSE	EPI[1]	Buildings (17)	3 years
(Dong, Cao, & Lee, 2005)	SVM	MSE, S-MSE, cv-RMSE, %error	Monthly	Commercial buildings (4)	4 years
(Tso & Yau, 2005)	OLS, RT, ANN	RASE	Weekly	Individual residential buildings (1000)	6 months
(Biwas, Robinson, & Fumo, 2016)	ANN	R2	Daily	Individual residential buildings (2)	72 days
(Darbellay & Slama, 2000)	ANN, ARMA	MAPE, maxAPE, NMSE	Daily, Hourly	Country (Czech Republic)	2 years
(Lahouar & Slama, 2015)	ANN, PER, RF, SVM	MAPE	Hourly	Country (Tunisia)	5,5 years
(Kuo & Huang, 2018)	DeepEnergy[2], SVM, RF, MLP, LSTM	cv-RMSE, MAPE	Hourly	State (Texas)	10 days
(Taylor, Menezes, & McSharry, 2016)	ARMA, OLS, ANN	MAPE	Hourly	City level	30 weeks
(Huo, Shi, & Chang, 2016)	RF, SVM	MAPE	Hourly	City level	Variable per model
(Jurado, Peralta, Nebot, Mugica, & Cortez, 2013)	ANN, Fuzzy logic, RF, SVM	NMSE	Hourly	Campus (1), commercial buildings (2)	1 year
(Wang, Wang, Zeng, Srinivasan, & Ahrentzen, 2018)	RT, RF, SVM	R2, RMSE, PI, MAPE	Hourly	Large institutional buildings (2)	1 year

(Fan, Xiao, & Zhao, 2017)	ANN, GBM, XGB, SVM, RF, ELN, MLR	cv-RMSE, MSE, RMSE	Hourly[3]	Institutional building (1)	1 year
(Li, Meng, Cai, Yoshino, & Mochida, 2009)	ANN, SVM	MRE, RMSE	Hourly[3]	Commercial building (1)	6 months
(Ryu, Noh, & Kim, 2016)	ANN(multiple), RIMA, and DSHW	MAPE, RRMSE	Hourly	Industrial buildings (5)	750 days
(Rodrigues, Cardeira, & Calado, 2014)	ANN	R2, RMSE, MAPE, SDE	Daily, Hourly	Individual residential buildings (93)	6 weeks
(Houimli, Zmami, & Ben-Salha, 2019)	ANN	MAE, MPE, MSE, MAPE, RMSE.	Half-hourly	Country (Tunisia)	9 years

Some studies focus on electricity, natural gas, or both.
[1]Energy performance index, thus no quantitative energy consumption.
[2]ANN based model.
[3]Building cooling load.

Some studies have reviewed publications on the usage of different models to predict energy demand profiles. Zhao and Magoulès (2012) conclude that each model has its advantages in certain cases and applications. In general, engineering-based models are difficult to create and show a large variety in prediction accuracy. Basic statistical models are relatively easy to develop but can be inaccurate and are less flexible. Artificial intelligent models, such as ANN and SVM, are good at solving nonlinear problems, which makes them very suitable for energy demand prediction, as long as the hyperparameters are tuned appropriately. In many cases SVM have even more superior results than the ANN approaches. Drawbacks of these two models are that they require sufficient historical data and are complex, which makes interpretation of results difficult. The authors state that the establishment of databases with precise and sufficient historical data of different entities is necessary to further research develop of reliable and effective prediction models. Also, the optimisation of parameter settings when using ML models is an important point (Zhao & Magoulès, 2012).

Another, more extensive, literature review study is conducted by Seyedzadeh et al. (2018). Not only did they review the ML models utilised for energy demand prediction, but also different pre-processing techniques to enhance prediction accuracy are discussed. The authors conclude that ML has shown great potential for energy modelling and assessment for different types of buildings. It has been shown that SVM outperforms ANN in load forecasting and has the potential to build models from limited amount of data. The authors do question this statement since they mention that the earlier used ANN models were from a simple structure and might not have the optimal hyperparameter settings. The Gaussian Process models (GP) are the only ones that have been used with uncertainty assessment. This is however not the only possible model to apply uncertainty and sensitivity analysis to. The authors recommend devoting additional research to these approaches. It is also recommended that more thorough research is desired with a focus on the tuning of the models. So that selecting ML models for energy demand forecasting becomes more convenient. The authors mention, as also brought forward in this literature study, that some studies did fairly not compare different ML models. A common mistake is putting much effort in optimizing one model and comparing it with default versions of other models which gives an unreliable result (Seyedzadeh, Rahimian, Glesk, & Roper, 2018).

2.2.4 Model selection

Many studies have suggested and compared different machine learning models for predicting energy demand of buildings and point towards the most promising algorithm in their case. Overall it is challenging to determine the best machine learning model, since the covered literature concludes that many models can provide decent accuracy when used appropriately, with sufficient data and optimized parameters. Multiple authors agree that there is no method that is clearly better than others (Taylor, Menezes, & McSharry, 2016). It is stated that the choice of the model is determined by the nature of the data (Darbellay & Slama, 2000). Therefore, it is essential to analyse the available data and application, to determine which model suits best in the given situation (Seyedzadeh, Rahimian, Glesk, & Roper, 2018). Empirically test different prediction models first, without optimizing hyperparameters, to see which model naturally performs better on a given dataset, is useful to select a proper method.

2.3 Research gap

The growing concerns about energy consumption in residential buildings have driven an interest in low- and net-zero energy buildings and legislation to increase building energy efficiency (Kneiffel & Webb, 2016). Although the residential building sector accounts for a large portion of the growing energy demand in the world today, the majority of the research is focused on commercial, industrial and transportation (Swan & Ugursal, 2009). Residential energy consumption is thus, underdeveloped for optimal and robust solutions (Biwas, Robinson, & Fumo, 2016). One of the reasons that residential energy consumption is less studied, is the lack of financial incentive compared to industrial, commercial and transportation sectors (Swan & Ugursal, 2009). The privacy sensitivity of collecting households' energy consumption data does also provide an obstacle in such studies (Biwas, Robinson, & Fumo, 2016). Authors of different studies have stressed the essence of additional research on smart grid solutions. The integrated grid solutions are important since they enable other sustainable energy solutions, such as EV, variable renewable sources and demand response (Kuo & Huang, 2018). More research is desired for enhancing forecasting capabilities to identify effective and appropriate use of renewable energy and energy storage (Rodrigues, Cardeira, & Calado, 2014). Although the availability of smart metering data has led to the expectation that electricity demand prediction will move toward the individual household prediction (Fumo & Biswas, 2015), most studies focus on large energy consuming entities or predict for a aggregated daily, monthly or annual values. Consumption of many different consumers accumulated, or large entities in general, often show a more smooth and constant demand pattern, which makes predicting overall more accurate.

Since many studies have already tested a variety of machine learning models, a combination of methods should be used to add significant value in the field of research. By combining different methods, an improvement can be made relative to only running an optimized algorithm, which most studies have done.

Lastly, most of the studies have not incorporated renewable energy production at the demand side, when predicting the energy demand. As this is becoming more and more the norm, additional research, regarding predicting consumers' energy demand with renewable energy generation systems on-site, is desired.

3. Methodology

The methodology of this research contains a combination of methods, to predict short term electricity demand of residential buildings. To improve the prediction accuracy, a clustering algorithm will be run preceding on the prediction model. Clustering households based on their electricity consumption profiles could be beneficial for the prediction accuracy. An operating system will be suggested to identify opportunities for improving self-sufficiency, by sharing electricity between clusters. The underlying theoretical principles of the used models will be clarified in the upcoming three sections of this report.

3.1 Clustering model

Many studies, as described in the previous chapter, have used different approaches to improve energy demand prediction accuracies. In this study a new approach is suggested for this same objective. Before the prediction model will take place, the houses in the neighbourhood will be clustered based on their electricity demand profiles. The goal of this clustering is to group houses with similar electricity demand patterns, so that the prediction for these clusters is more accurate and robust.

K-means clustering is a simple and convenient approach to divide a dataset into K distinct, and non-overlapping groups. This approach scales well to large number of samples, and has been used across a large range of application areas in different fields. The main principle for the K-means clustering approach is to minimize the 'within cluster variation'. Taking into account two important properties; (i) each observations, in this case demand profile, belongs to one of the K clusters. (ii) no observations belong to more than one cluster, thus, non-overlapping clusters. The problem that has to be solved to create good clusters according to the K-means method is (James, Witten, & Hastie, 2017):

$$\underset{C_1 \dots C_K}{Minimize} \left\{ \sum_{K=1}^{K} W(C_K) \right\} \tag{1}$$

Where K is the number of cluster, $C_1 \dots C_K$ represent all clusters and $W(C_K)$ is a measure of the amount by which the observations within the clusters differ from each other, named; the within cluster variation. The within cluster variation, summed over all the clusters, has to be as small as possible, to create good clusters according to the k-means approach.

The within cluster variation can be expressed in a number of ways, however, by far the most used method involves the squared Euclidean distance. Here, within the cluster variation is expressed as (James, Witten, & Hastie, 2017);

$$W(C_K) = \frac{1}{|C_K|} \sum_{i,i' \in C_K} \sum_{j=1}^{p} \left(x_{ij} - x_{i'j}\right)^2 \tag{2}$$

In Equation 2, corresponding to this research work, $|C_K|$ gives the number of houses in the Kth cluster. x_{ij} represents the value of feature j of observation i. $x_{i'j}$ is the average observation value of the cluster, or centroid.

Further, p stands for the total number of features used, in this case, only one feature is considered; total grid electricity demand. To put this equation to words; the within cluster variation for the Kth cluster is the sum of all the pairwise squared Euclidean distances between the observations of the Kth cluster, divided by the total number of observations in the Kth cluster (James, Witten, & Hastie, 2017).

The two previously described equations combined, gives the optimization problem that defines K-means clustering;

$$\underset{C_1 \ldots C_K}{Minimize} \left\{ \sum_{K=1}^{K} \frac{1}{|C_K|} \sum_{i,i' \in C_K} \sum_{j=1}^{p} \left(x_{ij} - x_{i'j}\right)^2 \right\} \tag{3}$$

In this study only one feature is used in the clustering, the optimization problem can therefore be simplified as:

$$\underset{C_1 \ldots C_K}{Minimize} \left\{ \sum_{K=1}^{K} \frac{1}{|C_K|} \sum_{i,i' \in C_K} (x_i - x_{i'})^2 \right\} \tag{4}$$

The algorithm that solves consists of two steps.

(I) Randomly assign a number, from 1 to K, to each of the houses. These serve as the initial cluster assignments for the houses.

(II) Iterate the following, until the cluster assignment stops changing:

a. For each of the K cluster, compute the cluster centroid, the average values for each observation of the cluster. The Kth cluster centroid is the mean of the p feature for the houses in the Kth cluster.

b. Assign each house to the cluster whose centroid is the closest, by using the Euclidean distance.

This algorithm is guaranteed to decrease the value of the objective function at each step. However, the (local) minimum that is reached, is dependent on the initial assignments of houses to clusters. Therefore, it is important to run the algorithm multiple times with different random initial configurations. So that the best solution, with the smallest outcome of Equation 3, can be selected (James, Witten, & Hastie, 2017).

This clustering technique will be executed in Python, by means of the Scikit-learn module 'cluster.kmeans'. This function has some parameters to be set before running it.

n_cluster:	Represents the number of desired clusters.
n_init:	The number of times the k-means algorithm runs, with different initial assignment of houses to clusters.
max_iter:	Maximum number of iterations of the k-means algorithm for a single run. In the case the clusters keep changing after a lot of iterations, this is the stopping rule.

When observing the electricity demand profiles of the clusters for a whole year, trends are very similar. Individual houses can have different energy consumption behaviour in winter compared to summer. This causes the clustering algorithm to distribute the clusters unevenly. In fact, when clustering the houses based on the full 2 years of data, the 70 houses are distributed over four clusters as: 67-1-1-1, which adds no value to the approach. When making the period smaller, the model can, after a number of iterations, determine clusters with substantial number of houses in it. To select a suitable period for the clustering model, multiple time periods are tested, seasons (three months), months and two week. The season and month periods show similar, although less extreme, behaviour as the two years of data. Two weeks seems to be a suitable period for the k-means clustering approach to create decent clusters. Each prediction day, a clustering algorithm must be executed.

To determine a suitable number of clusters (n_clusters), multiple tests have been conducted; six, five and four clusters, as shown in Appendix II – Cluster sizes.. When the parameter n_clusters is set to five or six, it is observed only three clusters have substantial size, the remaining clusters contain only a few houses. The small clusters in these cases do not add value to the overall prediction accuracy, so they will be added to the second smallest cluster which makes the effective number of clusters smaller. Based on this test the best number of clusters is selected to be four, and this will be used to do the clustering.

The number of times the clustering algorithm randomly puts centroids in the data (n_init) is on default set to 100. This means the model is run 100 times and selects the cluster distribution, from these 100 runs, where the total within cluster variation is the lowest. The maximum number of iterations (max_iter), which is an early stopping rule for when the cluster distribution keeps on changing, is set to 1000. The default setting of max_iter is 300, this is increased to 1000, to be sure the best cluster distribution is achieved and the algorithm is not stopped before the lowest within cluster variation is achieved.

Table 2 shows the parameter settings for the k-means clustering model that will be used throughout the research.

Table 2 Parameter settings clustering model

Parameter	Value
n_clusters	4
n_init	100
max_iter	1000

3.2 Prediction Model

3.2.1 Model selection

The literature has pointed out that multiple models can accurately predict energy demand when used under the right circumstances. To make a well grounded decision, multiple algorithms are tested with their default settings to see which model naturally performs best on the given data of individual houses without clustering. This is done by using the built in Matlab application Regression Learner. This application offers the possibility to test multiple algorithms on a dataset relatively quickly. In total 15 algorithms are tested, namely, multi linear regression models, ensembled trees, support vector machines and single prediction trees. Table 3 shows the results for the house ID-7056 as an example. In total 5 randomly picked houses are tested to determine the best performing algorithm empirically. In the used models, 70% of the two years' data is used for training and 30% for testing. For each tested algorithm, the R^2, RMSE, MSE and MAE are generated. The model with the best average values of the performance indices, of these 5 tests, is selected.

Table 3 Algorithm comparison, example house ID-7056

Model	***R-squared***	***RMSE***	***MSE***	***MAE***
Linear regression[1]	0.66	0.4160	0.1730	0.2594
Robust linear regression[1]	0.65	0.4248	0.1805	0.2432
Interaction linear regression[1]	0.67	0.4137	0.1711	0.2510
Ensemble Boosted trees	0.67	0.4080	0.1664	0.2363
Ensemble Bagged trees	0.67	0.4082	0.1666	0.2421
SVM Linear [1]	0.65	0.4226	0.1786	0.2428
SVM Quadratic	0.67	0.4138	0.1712	0.2293
SVM Cubic	0.65	0.4213	0.1775	0.2343
SVM Fine Gaussian	0.44	0.5367	0.2881	0.2110
SVM Medium Gaussian	0.66	0.4173	0.1742	0.2316
SVM Coarse Gaussian	0.66	0.4170	0.1739	0.2339
Tree Coarse	0.65	0.4228	0.1788	0.2554
Tree Medium	0.61	0.4466	0.1995	0.2736
Tree Fine	0.51	0.4979	0.2479	0.3032

House 7056
[1]Theoretically not suitable due to non-linear relationships between autoregressive features and the dependent variable.

This empirical test points out that the ensemble trees have the best performance among the tested algorithms; highest coefficient of determination and lowest errors terms. The literature study has clarified that the ensembled trees are performing well when a training set contains many correlated variables. The training set of this research, does indeed contain mutually dependent predictors such as autoregressive and meteorological features, see section 4.2.2. The literature is therefore in line with the empirical test results. A more in-depth review of ensemble tree based models, and their origin, will be given in the next section of this report.

3.2.2 Regression tree

Since the ensembled trees are derivatives of the regression tree a sort explanation of the regression tree will be given first. A regression tree consists of branches nodes and leaves (or terminal nodes) created by a series of splitting rules starting at the top of the tree (James, Witten, & Hastie, 2017) .

The basic principle of regression trees will be explained by means of Figure 3, which shows an example of a single regression tree with a depth of three.

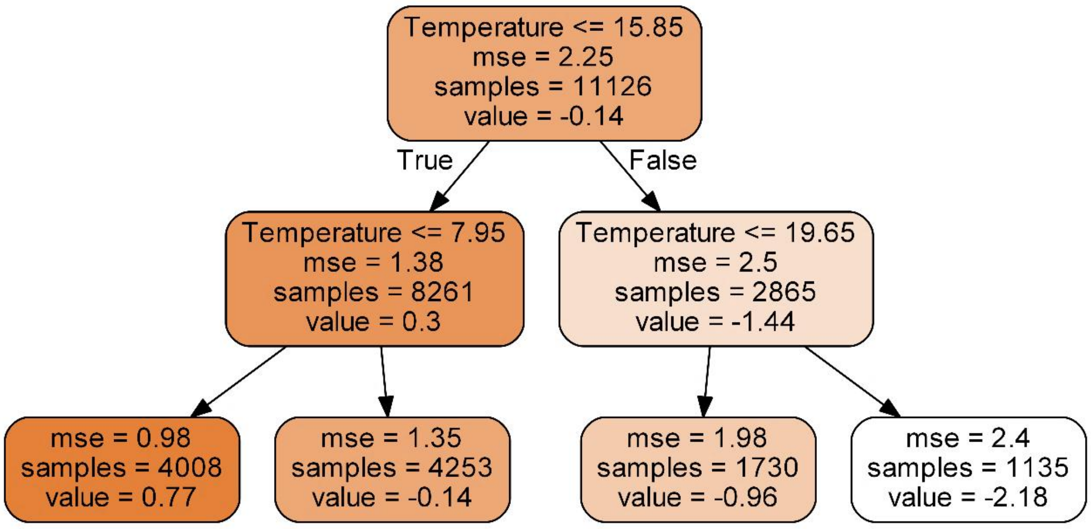


Figure 3 Example regression tree

In this example, a households' electricity demand for a certain hour is predicted based on the temperature on that particular hour. The top node splits the data for which the temperature is equal or smaller than 15.85 degrees Celsius to the left branch, and data for which the temperature is larger than 15.85 degrees Celsius to the right branch. These first regions are split ones more, so that four regions are created, see Figure 3, with each a predicted amount of electricity used:

$R_1 = \{X|Temperature \leq 7.95\}, \quad \hat{y}_{R_1} = 0.77kWh$

$R_2 = \{X|Temperature \leq 15.85, Temperature > 7.95\}, \quad \hat{y}_{R_2} = -0.14kWh$

$R_3 = \{X|Temperature > 15.85, Temperature < 19.65\}, \quad \hat{y}_{R_3} = -0.96kWh$

$R_4 = \{X|Temperature > 19.65\}, \quad \hat{y}_{R_4} = -2.18kWh$

The tree keeps building until a stopping point is reached determined by a stopping rule. In this case the stopping rule is a *max dept of 2*, the data is split twice. The prediction value for all the observations within a region is equal to the mean of the response variable of the training data in that region. Defining the size of the regions R_1 … R_J is based on minimizing the residual sum of squares (RSS), at the point of splitting of the total tree. The RSS is defined as (James, Witten, & Hastie, 2017):

$$RSS_j = e_1^2 + e_2^2 + \cdots + e_n^2 \tag{5}$$

Here e_i is the error of a prediction, this can be described as the difference between the measured value and the predicted value, defined as:

$$e_i = y_i - \hat{y}_i \tag{6}$$

Here, y_i is the observed value of the i-th sample and $\hat{y}_i$ is the predicted value of the i-th sample.

Total minimization formula can be described as (James, Witten, & Hastie, 2017):

$$Minimize \sum_{j=1}^{J} \sum_{i \in R_j} \left(y_i - \hat{y}_{R_j}\right)^2 \tag{7}$$

Where, $\hat{y}_{R_j}$ is the mean value of the dependent variable in the *j*-th region.

The algorithm which solves Equation 7, consists of two steps which create the regression tree (James, Witten, & Hastie, 2017):

(I) The predictor space is divided into J distinct and non-overlapping regions.
(II) For every value that falls in a particular region, the same prediction is made. This prediction is the mean of the dependent variable for the training observations in that region.

3.2.3 Random Forest

In the previously described example, only one independent variable is used and only four regions are constructed. Regression trees where vast amount of data is used to train, will have hundreds of regions and use multiple independent variables. When problems become more complex, with more independent variables and non-linear relationships, a single tree will not predict accurate (James, Witten, & Hastie, 2017).

Random forest is an ensemble learning method, consisting of a collection of regression trees. It is a homogeneous ensemble method since the model uses the same algorithm to create its base models, in this case regression trees. Random forests consist of multiple trees, where the prediction value is an averages of all the constructed trees. However now that there are multiple predictors, a rule must be defined to select the predictors for the splitting regions. In each tree, every time a split in a tree is created, a random sample of *m* predictors is chosen as split candidates from the full set of *p* predictors. So only a part of the predictors is considered each node. The goal of this approach is the decorrelation of the individual trees. Training sets often contain one, or a few, strong predictors, along with a number of moderately predictors. When multiple trees are constructed based on the strength of the predictors, the trees will look very much alike. The predictions of these trees will therefore be highly correlated. Averaging many highly correlated quantities will not reduce the variance as much as averaging many uncorrelated quantities. So creating multiple, comparable trees, will not reduce the variance over a single tree. Therefore, random trees try to create uncorrelated predictions by randomly choosing predictors at each node of each tree. By only using a subset of the predictors, the strong predictor will in some trees not even be considered and the other predictors have more influence. The average of the predicted values will be less variable and hence more reliable. Using this random forest with a small value of *m* will typically be helpful if a training set has many correlated predictors (James, Witten, & Hastie, 2017). The process of a random forest is visualized in Figure 4.

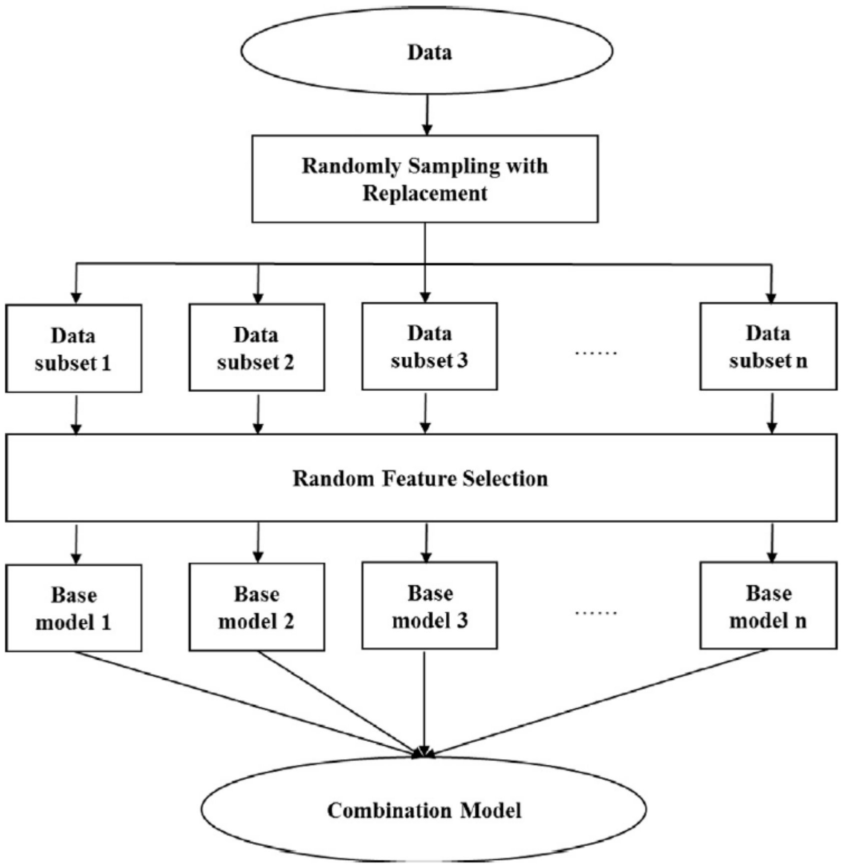


Figure 4 Random forest process (Wang, Wang, Zeng, Srinivasan, & Ahrentzen, 2018)

To measure the impact of each variable on the overall prediction performance, data permutation is used. By calculating in- or decrease of prediction accuracy resulting from randomly permuting the values of a variable, the importance of a variable can be determined. The larger the difference in prediction accuracy, the more the important of the variable, the smaller the difference the lesser the importance of a variable is (Wang, Wang, Zeng, Srinivasan, & Ahrentzen, 2018).

3.2.3.1 Parameters

The prediction model is executed with the python module Scikit-learn. Scikit-learn is an integrated Python module with a wide range of state-of-the-art machine learning algorithms for medium-scaled supervised and unsupervised problems. This package is, like Pandas and Numpy, also open source and encouraged to use both for scientific and commercial purposes (Pedregosa, et al., 2011). This program is chosen since it allows more flexibility in the fine tuning of parameter setting. With regression learner app from Matlab, used for the model selection in 3.2.1, only a few parameters, minimum leaf size and number of learners, can be changed. Where the Scikit-learn random forest regressor has much more possibilities in terms of parameter settings, see Table 4 (Scikit-learn, 2019).

Table 4 Prediction model parameters

Parameter	Description	Default setting
n_estimators	The number of trees in the random forest.	100
criterion	The function to measure the quality of a split.	'mse'
max_depth	The maximum depth of the tree.	'None'
min_samples_split	The minimum number of samples required to split an internal node.	2
min_samples_leaf	The minimum number of samples required to be at a leaf node.	1
min_weight_fraction_leaf	The minimum weighted fraction of the sum total of weights (of all the input samples) required to be at a leaf node.	0
max_features	The number of features to consider when looking for the best split.	'auto'[1]
max_leaf_nodes	Grow trees with max_leaf_nodes in best-first fashion. If None then unlimited number of leaf nodes.	'None'
min_impurity_decrease	A node will be split if this split induces a decrease of the impurity greater than or equal to this value.	0
bootstrap	Threshold for early stopping in tree growth. A node will split if its impurity is above the threshold, otherwise it is a leaf.	1e-7

[1]If auto, than max_features=n_features
(Scikit-learn, 2019)

Besides the model specific hyperparameters given in Table 4. There are also some parameters regarding the hardware usage such as, n_jobs, which indicates the number of jobs to run parallel, which is limited by the number of processors used (Scikit-learn, 2019). For these parameters the default settings are used, they will not be covered or explained further in this research.

3.2.3.2 Hyperparameter tuning

To optimize the hyperparameter settings, a model selection tool from Scikit-learn, called 'RandomizedSearchCV', is used. This tool enables running many different settings at ones and gives the parameter settings that have the best results. The randomized search tool creates a table with inputs for each predictor and runs all possible combinations of parameters on a threefold cross validation.

Table 5 Hyperparameter tuning input

Parameter	**Inputs**	**options**
n_estimators	Start=200, stop=400[1]	5
max_features	'auto', 'None'	2
max_depth	100, 120, 140, 'None'	4
min_samples_split	2, 3, 5, 10	4
min_samples_leaf	2, 3, 4	3
bootstrap	'True', 'False'	2
Total combinations		960
cross validation	3	3
Total runs		**2880**

[1]Start and stop indicate boundaries of the range where the options are distributed over

For all the parameters, not included in the optimisation model, the default settings are used. The exact coding and output of this optimization step can be found in Appendix V – Hyperparameter tuning. The combination with the best area under curve (AUC) score is considered as the best estimator. The confidence interval for AUC indicates the uncertainty for the prediction (DeLong, DeLong, & Clarke-Pearson, 1988). This optimization step has result in the following parameter settings, see
Table 6. These settings will be used for all the validations. The optimization of parameters is a one-time effort in this process.

Table 6 Result parameter tuning

Parameter	**Inputs**
n_estimators	350
max_features	'None'
max_depth	120
min_samples_split	3
min_samples_leaf	4
bootstrap	'True'

3.2.4 performance of the prediction model

There are different manners to evaluate the quality of fit of a prediction model to a set of observed data. One of them is the coefficient of determination (R^2), see Equation 8 (Fumo & Biswas, 2015):

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y}_i)^2} \tag{8}$$

Here, $\bar{y}$ is the mean of the dependent variable defined as:

$$\bar{y} = \frac{1}{n}\sum_{i=0}^{n-1} y_i \tag{9}$$

Where y_i is the observed value of the response variable of the *i*-th observation. $\hat{y_i}$ is the predicted value of the i-th sample. The terms $\sum(y_i - \hat{y_i})^2$ and $\sum(y_i - \bar{y_i})^2$, in Equation 8, are respectively named; *sum of squared errors* and *total sum of squared errors.* The value of the coefficient of determination varies between 0 and 1 (0 and 100 percent). This percentage indicates how much variability in the dependent variable, is accounted for by the independent variables. Many software packages have a built in R^2 calculation method. It is not necessarily true that a model with a high R^2 value fits the data well. Specifically for multi linear regression the coefficient of determination is adjusted and expressed as follow, see Equation 9:

$$R^2_{adj} = 1 - (1 - R^2)\frac{n-1}{n-p-1} \tag{10}$$

Here, n is the number of observations and p is the number of features, predictive variables.

Another performance indicator is the mean absolute error *(MAE). It* can be calculated to evaluate the quality of the model, the MAE is expressed in Equation 11.

$$MAE = \frac{\sum_{i=1}^{n} y_i - \hat{y_i}}{n} \tag{11}$$

Here, y_i is the observed or measured data and $\hat{y_i}$ is the predicted data generated by the prediction model. n represents the total number of observations. The MAE has a value between 0 and 1, where 0 is a perfect fit.

The mean squared error (MSE) calculates a risk metric corresponding to the expected value of the squared error, see Equation 12:

$$MSE = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \hat{y_i})^2 \tag{12}$$

Here, $\hat{y_i}$ is, again, the predicted value of the i-th sample and y_i is the corresponding true value, then the MSE is estimated over n number of observations.

Another parameter that tells something about the quality of fit of the model is the root mean square error *(RMSE)*, which is a measure of the scatter in the data around the model. The equation of RMSE for multi linear regression is written in Equation 13:

$$RMSE = \sqrt{\frac{\Sigma\left(y_i - \hat{y}_i\right)^2}{n}} = \sqrt{MSE} \tag{13}$$

Since the models will predict clusters of dwellings with a variable number of houses, the error terms will have different ranges, due to the different cluster sizes.

To compare the error terms of the different models the cumulative variation of root mean squared error (cv-RMSE) is used, this is the normalized RMSE. Normalisation is done by dividing the error term by the mean value of the particular observations of the cluster.

$$cv - RMSE = \frac{RMSE}{\bar{y}} \tag{14}$$

3.3 Operating system

3.3.1 Framework for operating system

For the proposed operating system, a framework is of some necessary conditions and requirements is drawn up. Without these conditions, the operating system cannot be applicable. Firstly, a computer system must be available which has access to all real time smart meter data of the neighbourhood and meteorological forecast for one day ahead. The computer must be able to store and pre-process the data and execute the clustering algorithm to create clusters, as well as execute the prediction algorithm to run the prediction. Further, this computer must, be able to operate all the switches in the electrical circuit of the neighbourhood. This computer system is the 'core' of the operation. Another requirement is that all houses are individually connected on the LV-grid, to the ESS and to the other houses. So that they can be connected and disconnected to the LV-grid circuit, ESS-circuit or another cluster, in the appropriate conditions.

3.3.2 Proposed circuit

A schematic overview of the electric circuit of the neighbourhood is given in Figure 5.

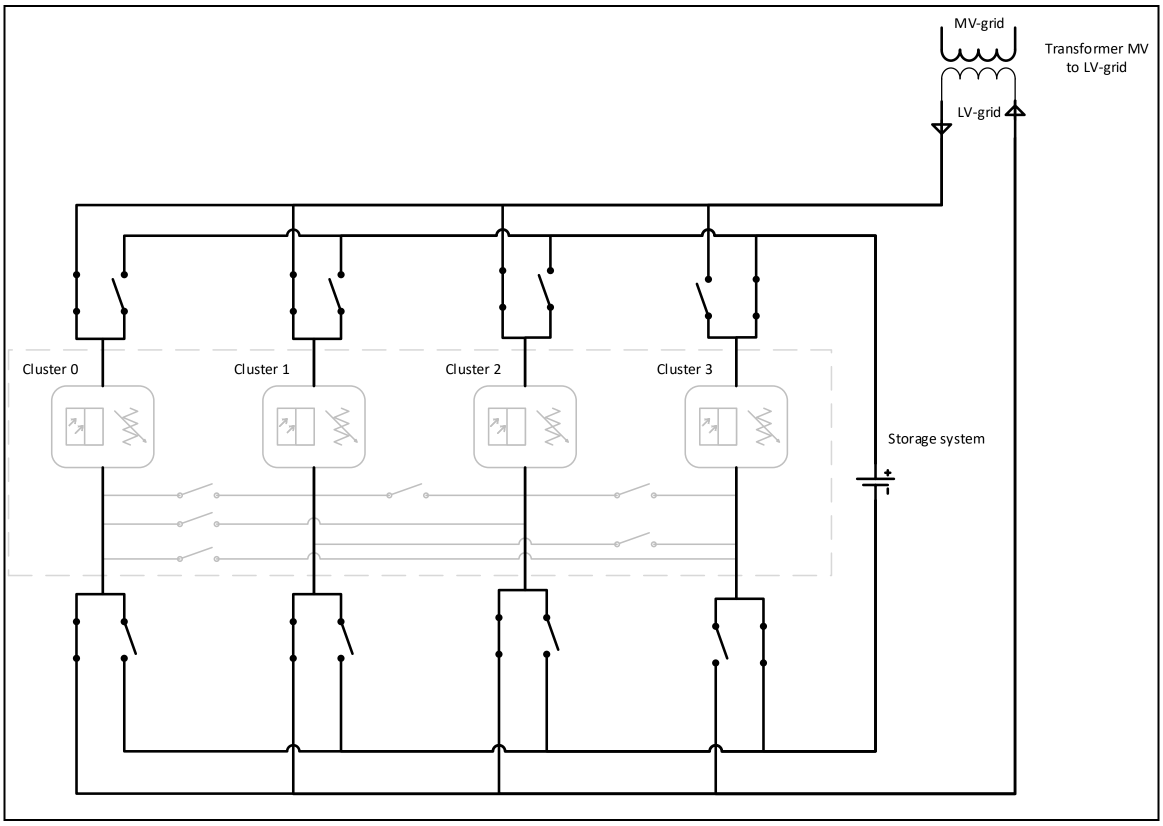


Figure 5 Schematic view of electrical circuit neighbourhood

In Figure 5 the houses are not pictured individually, but as clusters. In the circuit, each cluster is schematized as a variable resistor, representing the energy demanding state, and a solar station, which can provide electricity to the circuit when production exceeds demand. The clusters are connected to two switches, one for connecting to the LV-grid, and one for connecting to the ESS. The clusters can either be connected to the LV-grid, connected to the storage system, or completely disconnected. For example, in periods where enough solar energy is produced to provide the demanded electricity, or even more, to the cluster, it can be switched off from the LV-grid, and only connect to the storage system, so that it gets charged.

In cases where direct sharing is possible, the switches are able to create connections between them. Figure 6 highlights the subsystem of interconnected clusters to clarify the principle of the direct sharing circuits. These are two-way connections, electricity is able to go towards, and away from each cluster. This bidirectional connections breaks down the border between electricity generation and consumption, which is an important characteristic of the smart grid (Ryu, Noh, & Kim, 2016).

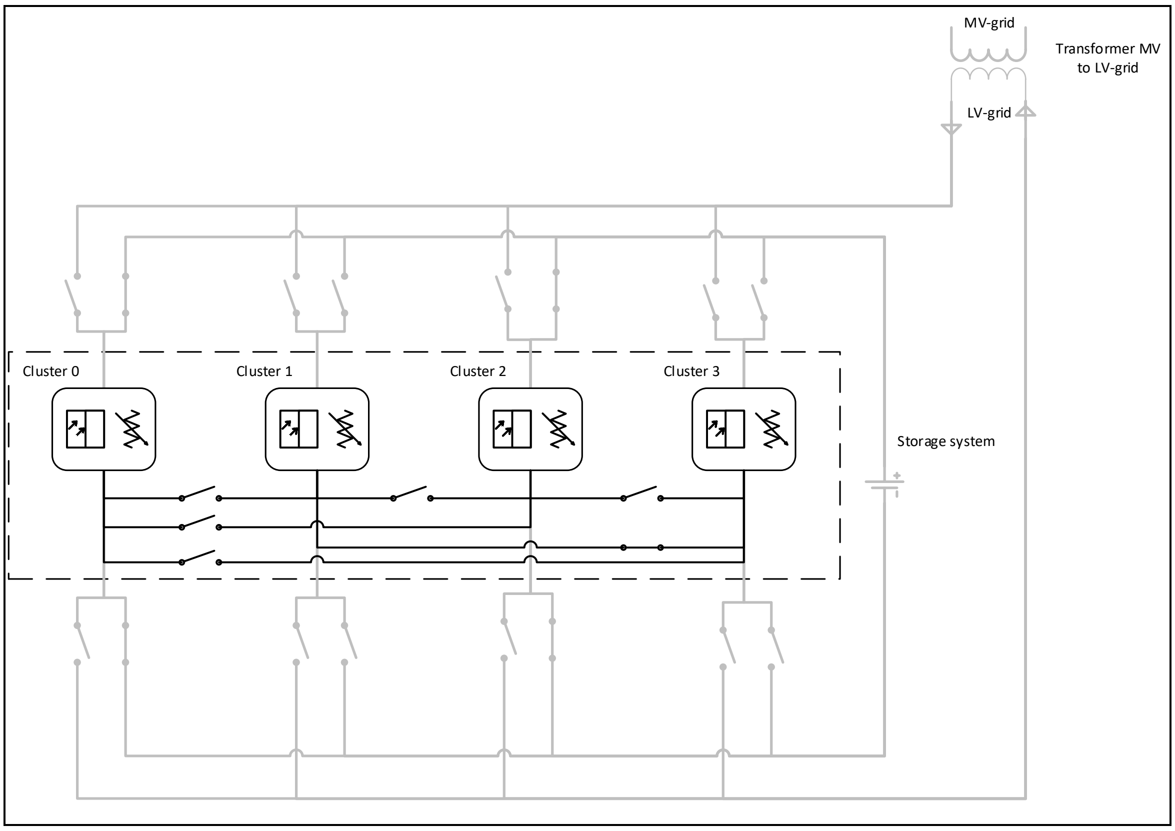


Figure 6 Electrical circuit for direct sharing

3.3.3 Operating rules

Together with the set framework and the proposed circuit, the following rules should be executed. Cluster 0 is used as example. Similar conditions and rules are applicable for the other clusters as well, see Table 7.

Table 7 Operating rules

	Conditions	*Operation rules*
1.	**If** Cluster0 has **positive demand**, **and** Cluster1 **and** Cluster2 **and** Cluster3 have **positive demand**, **and** ESS has **no** capacity[1]; **execute rule 1**, **else**; **check condition 2**.	Use gird power.
2.	**If** Cluster0 has **positive demand**, **and** Cluster1 **and** Cluster2 **and** Cluster3 have **positive demand**, **and** ESS has capacity; **execute rule 2**, **else**; **check condition 3**.	Use ESS power.
3.	**If** Cluster0 has **positive demand**, **and** Cluster1 **and/or** Cluster2 **and/or** Cluster3 have **negative demand**; **execute rule 3**, **else**; **check condition 4.**	Use surplus electricity from other cluster.
4.	**If** Cluster0 has **negative demand**, **and** Cluster 1 **and/or** Cluster2 **and/or** Cluster3 has/have **positive demand**; **execute rule 4**, **else**; **check condition 5**.	Share surplus of electricity to demanding cluster(s).
5.	**If negative demand** is **larger than** accumulated **positive demand** in other clusters; **execute rule 5**, **else**; **check condition 6**.	Charge ESS with excess electricity.
6.	**If** Cluster0 has **negative demand**, **and** Cluster1 **and** Cluster2 **and** Cluster3 have **negative demand**, **and** ECC **does have** charging capacity left; **execute rule 6**, **else**; **check condition 7**.	Charge ESS with excess electricity.
7.	**If** Cluster0 has **negative demand**, **and** Cluster1 **and** Cluster2 **and** Cluster3 have **negative demand**, **and** ECC **does not have** charging capacity left; **execute rule7**, **else**; **check condition 1**.	Send excess back to grid/ sell in demand response market/long -term storage.

*[1]*Substantially charged so, that it can provide the cluster with electricity.

These rules cover the main circumstances which can take place in the neighbourhood. There are many combinations possible for the exact status of each cluster. For example the surplus electricity, mentioned in operation rule 3, can come from any cluster which has a surplus in a specific point in time.

The objective is to create an operation schedule based on the predicted values of electricity demand in the different clusters. The operation schedule consists of a table with hourly operation commands, based on the condition rules in Table 7, for the operating system. Please note that the applicability of such schedule is very dependent on the prediction accuracy. In cases where the prediction models conditions deviate from the actual state of the neighbourhood, the operating rules might not be suitable.

Rule number-7 in Table 7, suggests to use excess electricity for long-term storage, when there is negative demand in the neighbourhood and the ESS is fully charged. Excess energy generated in summer and spring can be stored to consume during the winter and autumn when demand is higher. Li and Chan (2017) have summed up widely recognized energy storage technologies, besides ESS (Li & Chan, 2017):

- Thermal energy storage;
- Electrical and mechanical energy storage using flywheels;
- Pumped hydroelectric energy storage relying on reservoirs;
- Compressed air energy storage;
- Electrical energy storage using a combination of electrolysers and hydrogen fuel cells.

Exact application and technology of the long-term energy storage is not further considered in this study.

4. Data

The upcoming of smart meter data enables a closer look into the energy demand patterns of individuals, and possibly reveals possibilities in improving energy self-sufficiency. Smart meter data is often considered as privacy sensitive, therefore, not many studies have been able to use such detailed energy consumption data of a large sample of households. Since this research is completely built around a provided dataset, it is appropriate to assign a chapter to this data. In this chapter, the origin, source and content of the data will be described. Also an in-depth overview of the data pre-processing and feature selection will be given.

4.1 Data description and analysis

The data used in this research is provided by Royal BAM group. The renovated houses are located in the municipality of Soest, a village west of Amersfoort in the province of Utrecht. The data is considered as privacy sensitive, therefore no exact address information of the dwellings nor information about the housing corporation is given. No sociodemographic information or time schedules of residents is provided for this same reason.

The provided dataset consists of comma-separated values (CSV) files, with data of households' energy usage for every 15 minutes of the past two years (2016, 2017). The data is collected from renovated houses of a social housing corporation. The houses have living surfaces of 85 to 120 square meters, this is not known per dwelling. All houses have high quality insulation and new windows applied during the renovation. Further, the dwellings are not connected with natural gas and contain PV panels and HPs. More exact specifications, given by the provider of the dataset, are shown in Table 8.

Table 8 Technical specifications dwellings

Specification	**Value**
Living surface:	85-110m2
RC-score roof:	6
RC-score walls:	3.5
U-value windows:	1.1
Hot water buffer:	150L
Heat Pump power:	1.2kW (SPF 3.9)
PV installation capacity:	5-8kW peak

The dataset contains 13 variables, divided over; electricity consumption variables, electricity production variables, heat pump variables and boiler variables, each measured every 15 minutes, see Table 9.

Table 9 Variables description of provided dataset

Variable	Description
Time stamp	15-minute intervals from 2016-01-1T00:00:00+01:00Amsterdam to 2018-12-31T22:00:00+01:00Amsterdam
Consumed high	Consumed electricity on high tariff hours in kWh
Consumed low	Consumed electricity on low tariff hours in kWh
Solar inverter produced	Produced solar electricity
Produced solar high	Produced and fed back electricity on high tariff hours in kWh
Produced solar low	Produced and fed back electricity on low tariff hours in kWh
Heat pump consumed	Consumed electricity by the heat pump in kWh
Heat pump set point	Temperature set point in C˚ of the heat pump
Heat pump room temp	Actual room temperature in C˚
Heat pump space heating delivered	Amount of heating delivered by heat pump in GJ
Boiler hot tap water consumed	Volume of consumed hot tap water in m^3
Boiler set point temperature tap water	Set point temperature of boiler in C˚
Boiler supply temperature	Temperature of hot tap water when consumed in C˚

In many of the CSV-files, the columns regarding boiler set point temperature, actual boiler temperature and space heating delivered, is completely empty. These variables are taken out of the datasets entirely for consistency purposes, since it is not desirable to have differences in the number of variables in the dataset. The variables regarding boiler temperatures will not be found anywhere further in the research.

The households use a dynamic cost pricing for their electricity consumption. In the daytime period, from 06:30 in the morning to 22:30 in the evening, the high tariff is applicable. The low tariff is applicable in the night time, from 22:30 in the evening to 06:30 in the morning. This could be used to optimize the operation system of the ESS. This research will however not further analyse any opportunities regarding dynamic pricing.

4.2 Feature engineering

The data pre-processing is done in Python (PFS, 2019). In the data cleaning process two modules are used, which enable fast and efficient data preparation in the Python programming language; Pandas and NumPy. Pandas is a Python data analysis library (Pandas, 2019). Numpy is a Python module for scientific computing with, among many other things, useful linear algebra and random number capabilities (NUMFOCUS, 2019). Another used panda module is Matplotlib, which is an open source plotting library (Hunter, Dale, Firing, Droettboom, & team, 2019). Matplotlib will be used to visualize pre-processed data and results.

4.2.1 Prediction variable

The goal is to predict the total demand these houses have on the electricity grid, and improve the energy self-sufficiency by sharing between houses, or clusters of houses. The total demand on the grid computed by summing the consumption variables 'Slimme Meter p1 consumed kWh high',' Slimme Meter p1 consumed kWh low and 'Heat pump consumed kWh', and subtracting the total produced electricity from the PV system 'Solar Inverter kWh produced'. This new variable is called 'grid_consumption' and will be the response variable. This variable contains both positive, when the houses are demanding electricity and negative values, when they have a surplus from electricity generated by the PV panels.

In total 38 predictive features are created, divided over; autoregressive variables, categorical time-related variables and meteorological variables.

4.2.2 Auto regressive features

Features based on historical data are used to improve the model accuracy. In this case total grid consumption will be used to create the autoregressive features. In total eight autoregressive features are created, see Table 10.

Table 10 Autoregressive features

1.	tot_gridconsumed ts-1day	5.	tot_gridconsumed ts-5days
2.	tot_gridconsumed ts-2days	6.	tot_gridconsumed ts-6days
3.	tot_gridconsumed ts-3days	7.	tot_gridconsumed ts-7days
4.	tot_gridconsumed ts-4days	8.	-tot_gridconsumed ts-14days

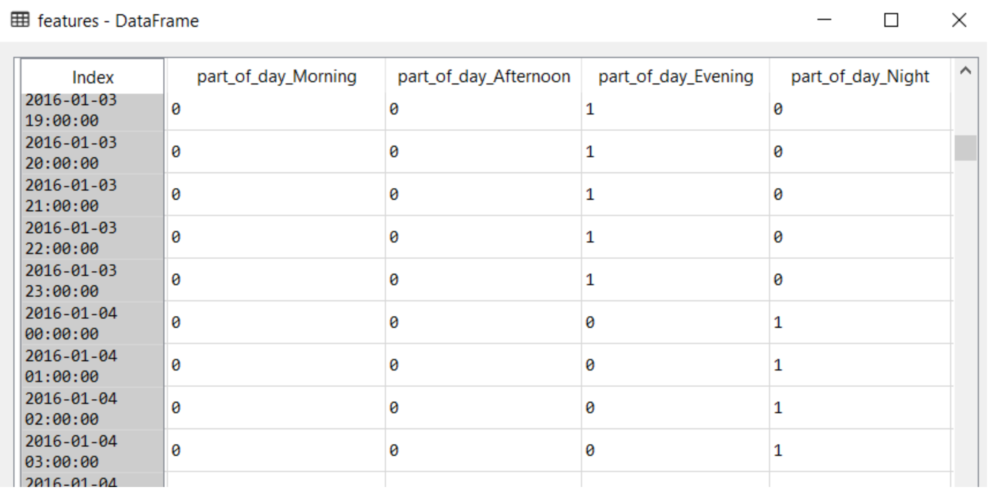
These features can be made based on the 'tot_grid_consumed' variable by means of the Pandas function 'shift()' function. This shifts the index by the desired number of periods, where each observation is a period. The 'tot_grid_consumed' variable is shifted by the number of observations to reach previous day, or days. For example, to reach the previous day the data must be shifted by 96 observations (24 hours has 96 observations of 15 min). This means that these autoregressive variables lack data in the first rows, since there is no previous data for the first day of the dataset. For 'tot_gridconsumed ts-14days' this means that the first 1344 observations are not present.

4.2.3 Categorical features

It is recommended by Mocanu et al. (2016) that adding extra information concerning the time, such as day and month would improve their model (Mocanu, Nguyen, Gibescu, & Kling, 2016). To do this, four features are added to the dataset, 'part of the day', 'weekday', 'month' and 'season'. The Pandas type 'DateTime' recognizes the weekdays (Monday – Sunday) and months (January – December) based on the date and time. Here, two columns are added, one with a label for the weekday and one with a label for month. The part of day variable is created manually by adding new column in the dataset with the label 'Morning' for daytime intervals 6:00 – 12:00, 'Afternoon' for daytime interval 12:00-18:00, 'Evening' for daytime interval 18:000 -24:00, and 'Night' for daytime interval 24:00-6:00. The variable 'season' is created based the meteorological season designation; summer 'June-July-August', Autumn

'September-October-November', winter 'December-January-February ' and Spring 'March-April-May', according to the KNMI (KNMI, 2019).

For all the previously described categorical features, dummy-features are created. This allows the model to recognize these features as a categorical type. To do this the Pandas function which creates dummy variables. This transforms the designated features to dummy features. For example, 'part of the day' is split into four features, morning, afternoon, evening and night, where by means of zeros and ones it indicates if a particular datapoint belongs to which part of the day, see Figure 7.


features - DataFrame

Index	part_of_day_Morning	part_of_day_Afternoon	part_of_day_Evening	part_of_day_Night
2016-01-03 19:00:00	0	0	1	0
2016-01-03 20:00:00	0	0	1	0
2016-01-03 21:00:00	0	0	1	0
2016-01-03 22:00:00	0	0	1	0
2016-01-03 23:00:00	0	0	1	0
2016-01-04 00:00:00	0	0	0	1
2016-01-04 01:00:00	0	0	0	1
2016-01-04 02:00:00	0	0	0	1
2016-01-04 03:00:00	0	0	0	1

Figure 7 Example dummy variables 'part of day'

In total 27 categorical features are created, see Table 11.

Table 11 Categorical features

Part of day and day of week	Months	Seasons
1. part_of_day_Night	12. month_January	24. season_winter
2. part_of_day_Evening	13. month_February	25. season_spring
3. part_of_day_Morning	14. month_March	26. season_summer
4. part_of_day_Afternoon	15. month_April	27. season_autumn
5. weekday_Wednesday	16. month_May	
6. weekday_Tuesday	17. month_June	
7. weekday_Thursday	18. month_July	
8. weekday_Sunday	19. month_August	
9. weekday_Saturday	20. month_September	
10. weekday_Monday	21. month_October	
11. weekday_Friday	22. month_November	
	23. month_December	

Some of these variables, such as 'season_winter' and 'month_January' are correlated, this should be taken into account when picking a prediction model.

4.2.4 Meteorological features

In addition to the features from the provided dataset, freely available meteorological data is gathered from the KNMI database. In many, energy demand prediction models, meteorological variables are used. The models tested by Fumo and Biwas (2015) contained meteorological data such as temperature, humidity and solar radiation (Fumo & Biswas, 2015). Also, Mocanu et al. (2016) recommend using extra weather related information such as outside temperature would improve their model (Mocanu, Nguyen, Gibescu, & Kling, 2016).

Weather station de Bilt, with a distance of approximately ten kilometres is the closest weather station from renovation project in Soest. The KNMI has freely available historical data which can be downloaded online by everyone. For the period from 31-12-2015 to 31-12-2018 the hourly measured data for temperature, relative humidity and dew point temperature is downloaded. Temperature is measured in 0.1 degrees Celsius, 1.50 meter above ground level. The relative humidity is measured in percentage, 1.50 meter above ground level. Dew point temperature is measured in 0.1 degrees Celsius, 1.50 meter above ground level. These three meteorological features are numerical and have the notation in the datasets as visible in Table 12.

Table 12 Meteorological features

1.	Temperature
2.	Humidity
3.	Dew point temp

It should be taken into account, when selecting a prediction model, that the meteorological variables are correlated with each other (Fumo & Biswas, 2015). When applying linear models to this data, the highly correlated variables should be merged. Also, these meteorological variables are actually measured values. When predicting one day ahead, no certain information of these variables is available and only weather forecast data can be used.

4.3 Data cleaning

4.3.1 Unwanted strings removal

The provided dataset contains unwanted strings in numerical columns, such as units, which makes it unable to operate the values. This is because of the fact that Python automatically recognizes the data as strings wen literal characters are involved, rather than the numerical values they represent. The first column of the dataset indicates the timestamp of the measured values, an example of this is ‘2016-01-01T01:30:00+01:00 Amsterdam’. The Pandas module is able to recognize time indication when it is displayed according to a certain standard. As can be seen the provided timestamp contains some unwanted information; ‘T’, ‘+01:00’ and ‘Amsterdam’. These strings are removed and the timestamp column is converted to ‘pandas.DateTime’ so python recognizes it as a date time. This is necessary for feature engineering later where indicators will be made for date and time related features. Consumption and production data in the dataset are consequently displayed with units ‘*kWh*’, temperature related data contains ‘*°C*’, heating delivery data contains ‘*GJ*’ and hot water usage contains '*m^3*'. These units are removed and the values are transformed to type *'float'* to create editable numerical values.

4.3.2 Replacing outliers

The provided data contains some unrealistic outliers, values a hundred times larger than the mean. The data is inspected visually to see where the threshold for a so called outlier should be. According to this visual inspection a value of 2kWh is determined to be the threshold for outliers, every value above will be replaced by a realistic replacement, its preceding value. This is done by means of a Numpy function 'forward fill'. So instead of the outlier, the value preceding on that outlier, will occur twice. To test the whether this threshold of 2kWh is suitable, a random sample of 5 houses are tested. As an example, for house ID-5005, only 20 values are above 2kWh, which is only 0.028% of the observations. Since the amount of values above the threshold is so small, and the observations just below the threshold are high, the threshold value is considered realistic and suitable for this data. After filling the outliers, the pattern of electricity demand over time becomes more visible, see Figure 8.

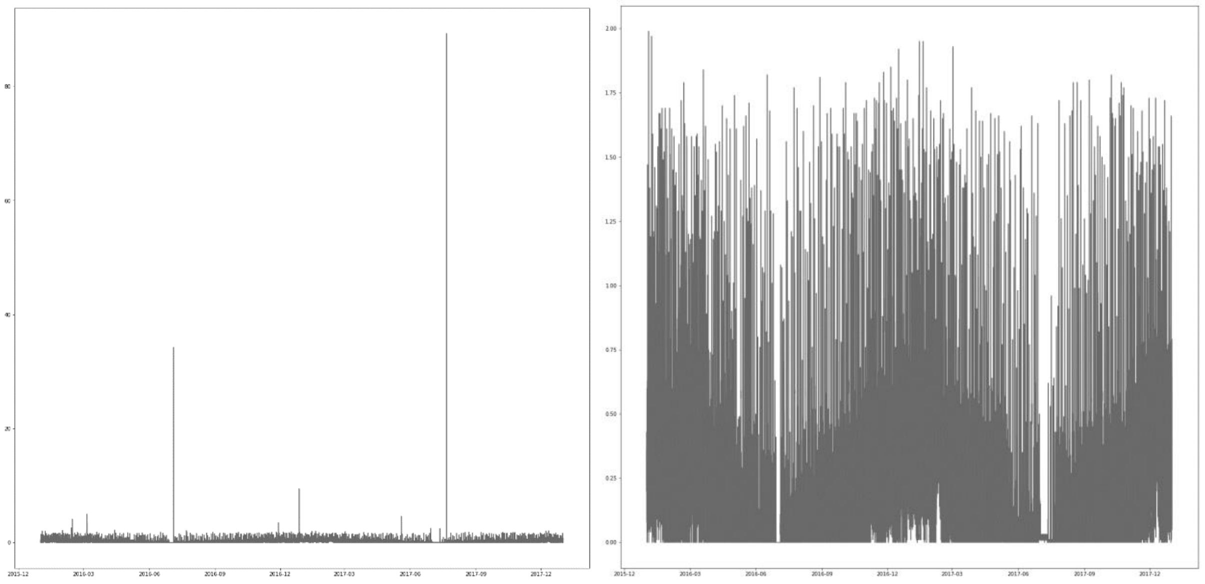


Figure 8 Removing outliers, before and after

4.3.3 Replacing missing values

For the sake of the quality and reliability of the model, 70 houses with the least missing values are selected. These houses have on average *10.8%* missing values in the 'grid consumption' variable, ranging from 5.2% to 29.8%. These missing values are firstly filled by the values of the previous day same time. This makes sense since there is a high correlation (correlation value of 0.84), between 'tot_grid_consumption' and 'tot_gridconsumed ts-1day'. This first step does not completely fill all the missing values. On places where more than a full day of data is missing, missing values remain unfilled. The next step is to fill the missing values with the consumption values of the previous week same time. The correlation between 'tot_grid_consumption' and 'tot_gridconsumed ts-14days' is lower than for the previous day but still notable with 0.59. Still not all missing values are filled, on places where more than a week of data is missing consecutive the missing values remain. However, the number of left over missing values are considered acceptable. After filling the missing values with the described two steps, the average amount decreased from 10.8% to 2.91%, ranging from 1.1% to 14.2%. This is considered acceptable. As a comparison, the study of Robinson et al. (2017) removes samples with more than 25% missing values (Robinson, et al., 2017). The complete overview of missing values can be seen in Appendix I – Missing values.

4.4 Clustering and prediction sets

After going through the data pre-processing, a clustering set is created. This set only contains the grid demand of each house, since the clustering will be solely based on the grid demand profiles of the houses. A visualisation of grid demand for two years of a single house is given in Figure 9. The grid demand is often, especially in the summer months, negative. This indicates that the PV system generates more electricity than used at that specific point in time. The total grid consumptions, which can be negative, is calculated according to Equation 15.

$$\begin{aligned}\text{tot\_grid\_consumpiton} = {} & \text{Slimme Meter p1 consumed kWh high} + \\ & \text{Slimme Meter p1 consumed kWh low} + \text{Heat pump consumed kWh} \\ & - \text{Solar Inverter Produced kWh}\end{aligned} \qquad (15)$$

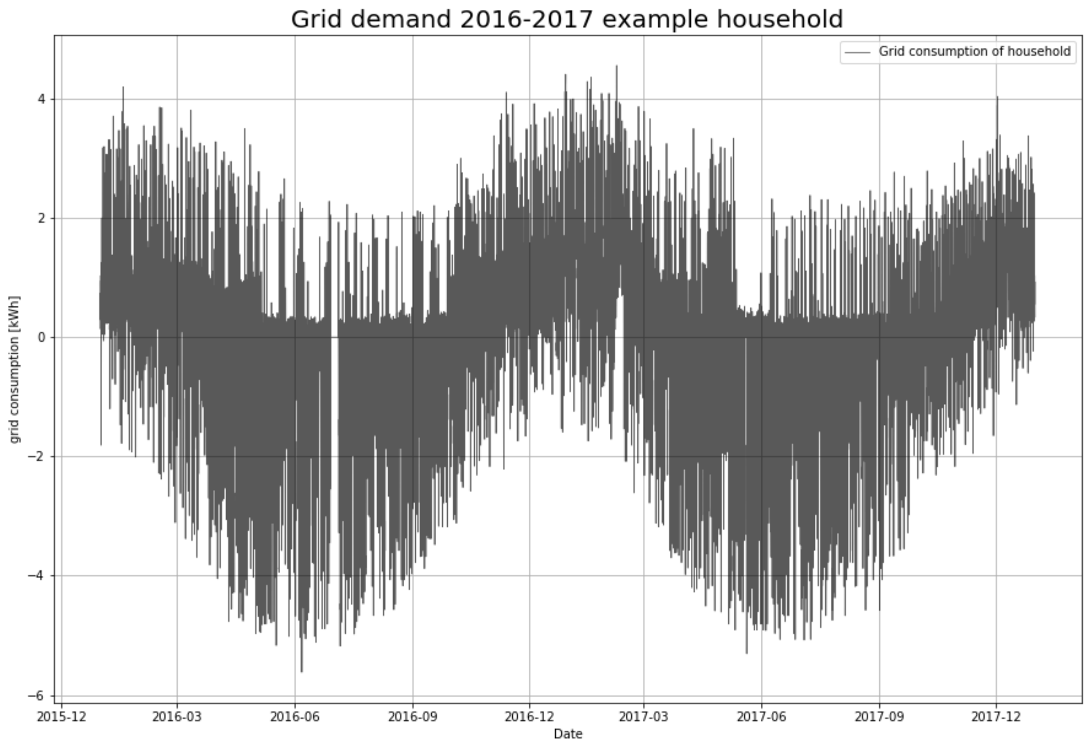
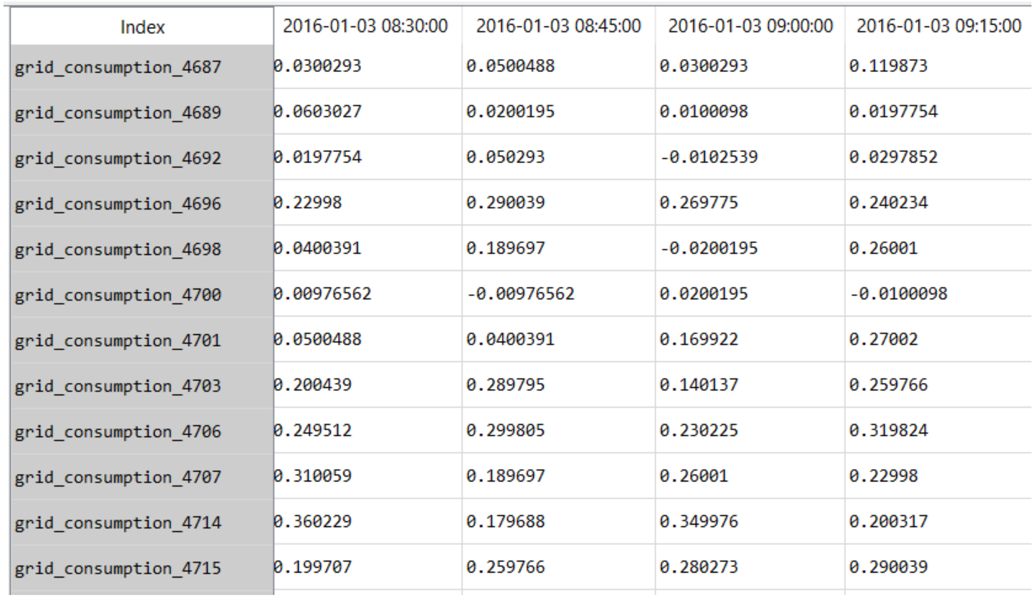


Figure 9 Grid demand 2016-2017 for example household

It can be visually observed from Figure 9 that, currently, large quantities of electricity are fed back to the grid. Managing this excess electricity by using a ESS could therefore increase the self-sufficiency of the dwellings.

For each individual household a prediction set is created, containing all the previously described features. The prediction set is made hourly and has a shape of 38x17544 with 666,627 data points per prediction set. In total one clustering set and 70 prediction sets are generated.

In this clustering set the rows represent an individual house, the columns contain the timestamps, see Figure 10. The clustering set consists of fifteen-minute interval data and has a shape of 70176x70, which is 4,912,320 data points. From this data set is used to extract the two weeks preceding on the prediction day.



Index	2016-01-03 08:30:00	2016-01-03 08:45:00	2016-01-03 09:00:00	2016-01-03 09:15:00
grid_consumption_4687	0.0300293	0.0500488	0.0300293	0.119873
grid_consumption_4689	0.0603027	0.0200195	0.0100098	0.0197754
grid_consumption_4692	0.0197754	0.050293	-0.0102539	0.0297852
grid_consumption_4696	0.22998	0.290039	0.269775	0.240234
grid_consumption_4698	0.0400391	0.189697	-0.0200195	0.26001
grid_consumption_4700	0.00976562	-0.00976562	0.0200195	-0.0100098
grid_consumption_4701	0.0500488	0.0400391	0.169922	0.27002
grid_consumption_4703	0.200439	0.289795	0.140137	0.259766
grid_consumption_4706	0.249512	0.299805	0.230225	0.319824
grid_consumption_4707	0.310059	0.189697	0.26001	0.22998
grid_consumption_4714	0.360229	0.179688	0.349976	0.200317
grid_consumption_4715	0.199707	0.259766	0.280273	0.290039

Figure 10 Clustering set example

This page is intentionally left blank.

5. Results

In the first part of this results chapter shows the overall results of the 10 validation days. The days represent both week- and weekend days and contain all four seasons. The second part of this chapter focuses on one validation day, May 19th 2017. The results and process to get to the results will be described and explained in detail.

.

5.1 Overall results

Figure 11 shows a visual representation of the complete process. This process has to be worked through for every prediction that is executed.

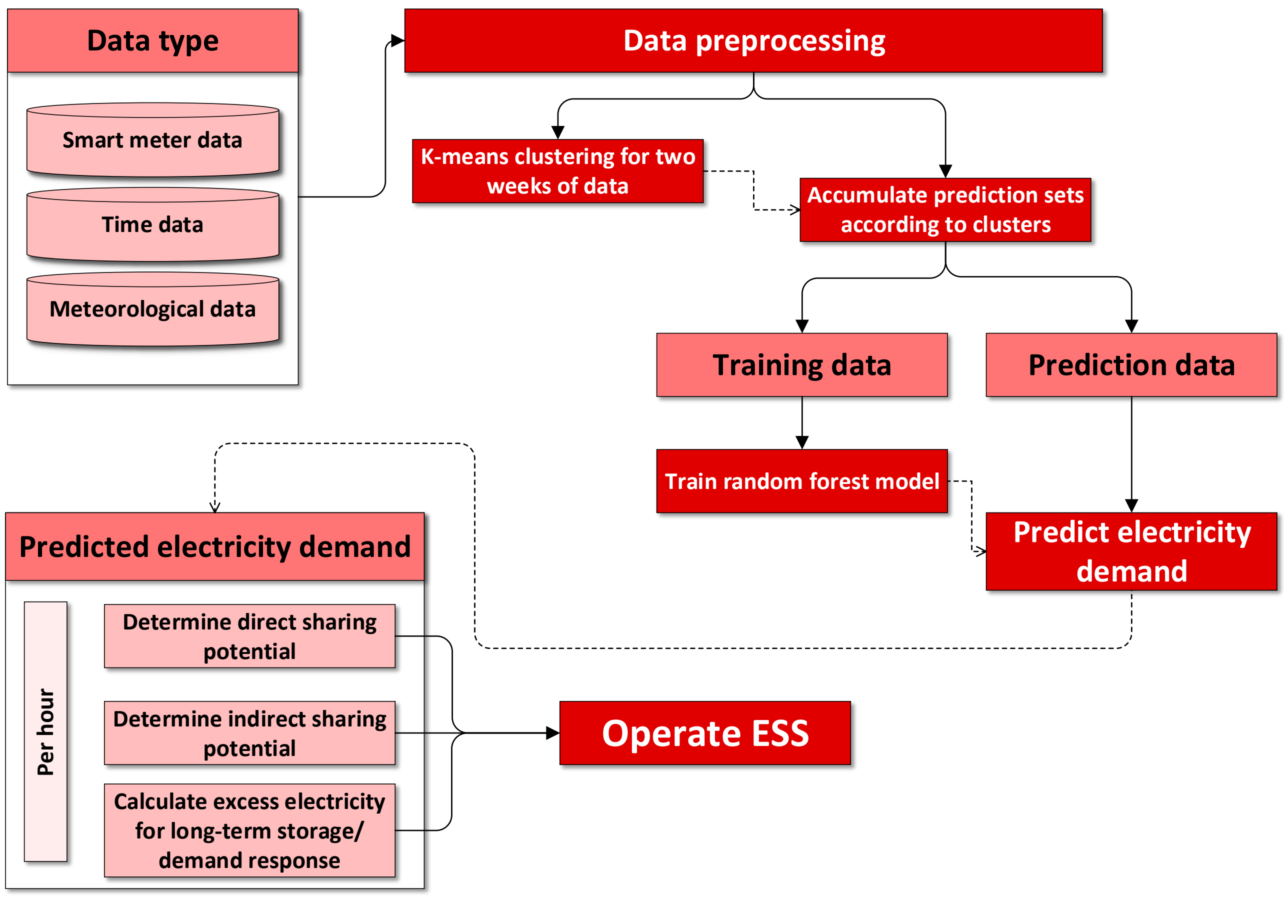

Data type
Smart meter data
Time data
Meteorological data
Data preprocessing
K-means clustering for two weeks of data
Accumulate prediction sets according to clusters
Training data
Prediction data
Train random forest model
Predict electricity demand
Predicted electricity demand
Per hour
Determine direct sharing potential
Determine indirect sharing potential
Calculate excess electricity for long-term storage/ demand response
Operate ESS

Figure 11 Research methodology

In total, 10 validation days have been selected throughout 2017, see Table 13. The validation days cover week and weekend days, are all in different months and cover all seasons.

Table 13 Validation days

VALIDATION	CLUSTER WEEKS	PREDICTION	DAY	# OF CLUSTERS
1	20-01-2017 to 03-02-2017	04-02-2017	Saturday	2
2	09-03-2017 to 22-03-2017	23-03-2017	Thursday	3
3	06-04-2017 to 19-04-2017	20-04-2017	Thursday	4
4	04-05-2017 to 18-05-2017	19-05-2017	Friday	4
5	05-06-2017 to 18-06-2017	19-06-2017	Monday	3
6	09-07-2017 to 22-07-2017	23-07-2017	Sunday	4
7	01-08-2017 to 14-08-2017	15-08-2017	Tuesday	3
8	09-09-2017 to 23-09-2017	24-09-2017	Sunday	4
9	03-10-2017 to 16-10-2017	17-10-2017	Tuesday	3
10	06-12-2017 to 19-12-2017	20-12-2017	Wednesday	3

As can be seen in Table 13, some validation days have less than four clusters. The k-means clustering technique is forced to create 4 cluster, which is in some cases, is not suitable for the actual patterns in the data. This can results in a cluster with only one or a few houses. In such cases, the cluster with the smallest number of houses, is added to the second smallest cluster, see Appendix III – Cluster distribution September 2017 for an example. This approach is used to keep the prediction accuracy of the clusters high.

Table 14 shows, for all the validation days, the model accuracy indices. Some noteworthy observations can be made from these results.

Table 14 Total validation results

	Validation nr.	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
	Date	February 3rd	March 23rd	April 20th	May 19th	June 19th	July 23rd	August 15th	September 24th	October 16th	December 20th
Cluster	Weekday	Saturday	Thursday	Thursday	Friday	Monday	Sunday	Tuesday	Sunday	Tuesday	Wednesday
Cluster 0	R2	0.69	-	0.92	0.73	0.91	0.85	0.80	0.77	-	0.48
	R2-adjusted	0.68	-	0.92	0.73	0.91	0.85	0.80	0.77	-	0.47
	MAE	8.21	-	10.50	16.22	7.14	11.55	9.05	10.33	-	3.47
	MSE	107.93	-	173.19	524.61	127.27	274.46	161.37	218.06	-	18.07
	RMSE	10.39	-	13.16	22.90	11.28	16.57	12.70	14.77	-	4.25
	absolute mean	27.90	-	40.39	41.55	33.53	35.39	23.14	25.29	-	28.89
	cv-RMSE	0.36	-	0.33	0.55	0.34	0.47	0.55	0.58	-	0.15
Cluster 1	R2	-	0.94	0.90	0.77	0.94	0.76	0.85	0.83	0.89	0.26
	R2-adjusted	-	0.94	0.90	0.77	0.94	0.76	0.85	0.83	0.89	0.25
	MAE	-	8.51	7.47	7.71	6.05	5.88	8.56	4.18	3.39	3.32
	MSE	-	120.87	93.60	107.17	66.30	79.66	131.33	40.99	19.46	19.06
	RMSE	-	10.99	9.67	10.35	8.14	8.93	11.46	6.40	4.41	4.37
	absolute mean	-	37.51	26.16	20.85	32.56	18.17	24.74	13.67	11.77	16.46
	cv-RMSE	-	0.29	0.37	0.50	0.25	0.49	0.46	0.47	0.37	0.27
Cluster 2	R2	078	0.89	0.92	0.75	-	0.87	0.86	0.87	0.41	0.48
	R2-adjusted	0.78	0.89	0.92	0.75	-	0.87	0.86	0.87	0.41	0.48
	MAE	8.54	10.86	7.33	9.58	-	5.24	4.05	8.17	7.84	4.33
	MSE	175.73	190.05	101.04	188.75	-	45.68	38.65	116.41	123.71	30.98
	RMSE	13.26	13.79	10.05	13.74	-	6.76	6.22	10.79	11.12	5.57
	absolute mean	36.68	36.65	30.31	26.07	-	17.97	14.58	24.74	16.21	31.67
	cv-RMSE	0.36	0.38	0.33	0.53	-	0.38	0.43	0.44	0.69	0.18
Cluster 3	R2	-	0.90	0.94	0.81	0.62	0.71	-	-	0.79	-
	R2-adjusted	-	0.90	0.94	0.81	0.62	0.71	-	-	0.79	-
	MAE	-	7.60	3.84	7.53	8.56	6.63	-	-	4.29	-
	MSE	-	98.82	28.41	101.94	181.60	103.37	-	-	29.74	-
	RMSE	-	9.94	5.33	10.10	13.48	10.17	-	-	5.45	-
	absolute mean	-	28.12	19.14	21.15	20.30	20.06	-	-	9.43	-
	cv-RMSE	-	0.35	0.28	0.48	0.66	0.51	-	-	0.58	-

All validation days are from 2017

The variance of the prediction accuracies from the different validation day is rather large. Multiple validations have R2 values of over 0.90 which is high, considering that this is a validation. An example of very good results is visible in Figure 12. As can be seen does the prediction line follow the same trend as the actual measured profiles. In the middle of the day, when excess electricity is at its peak, the prediction model shows less extreme shapes, especially in cluster 1 and 2. This causes the errors, performance indices, to be rather high.

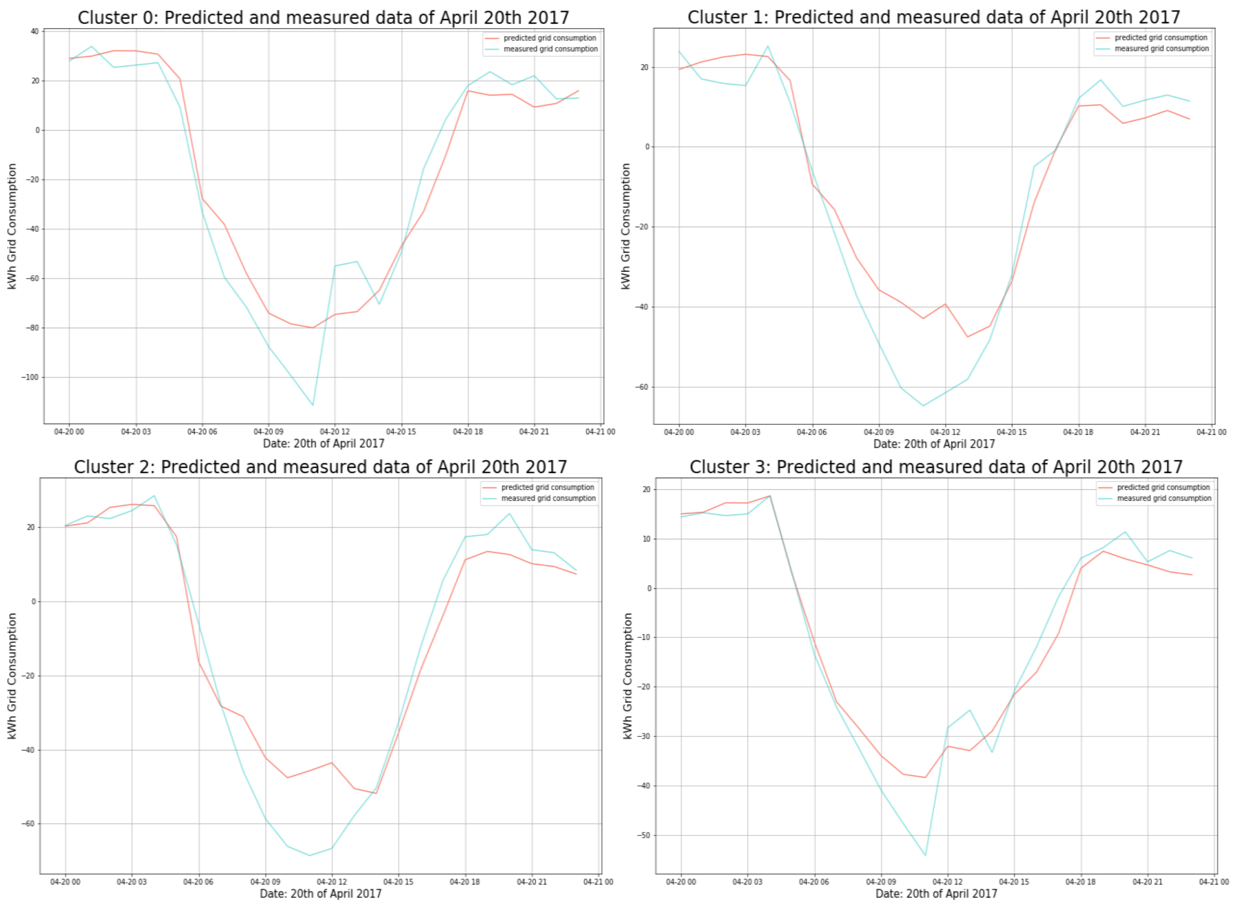


Figure 12 Predicted and measured values validation number-3

On the other hand there are also validation runs where the R2 is below 0.50. The worst performing models are from validation number-10, December 20th, and cluster 2 of validation number-9, October 16th. As can be seen in Table 15, this latest mentioned cluster, consists of 36 houses, which is large, compared to the other clusters. This cluster has also very high error terms, with a cv-RMSE of 0.69.

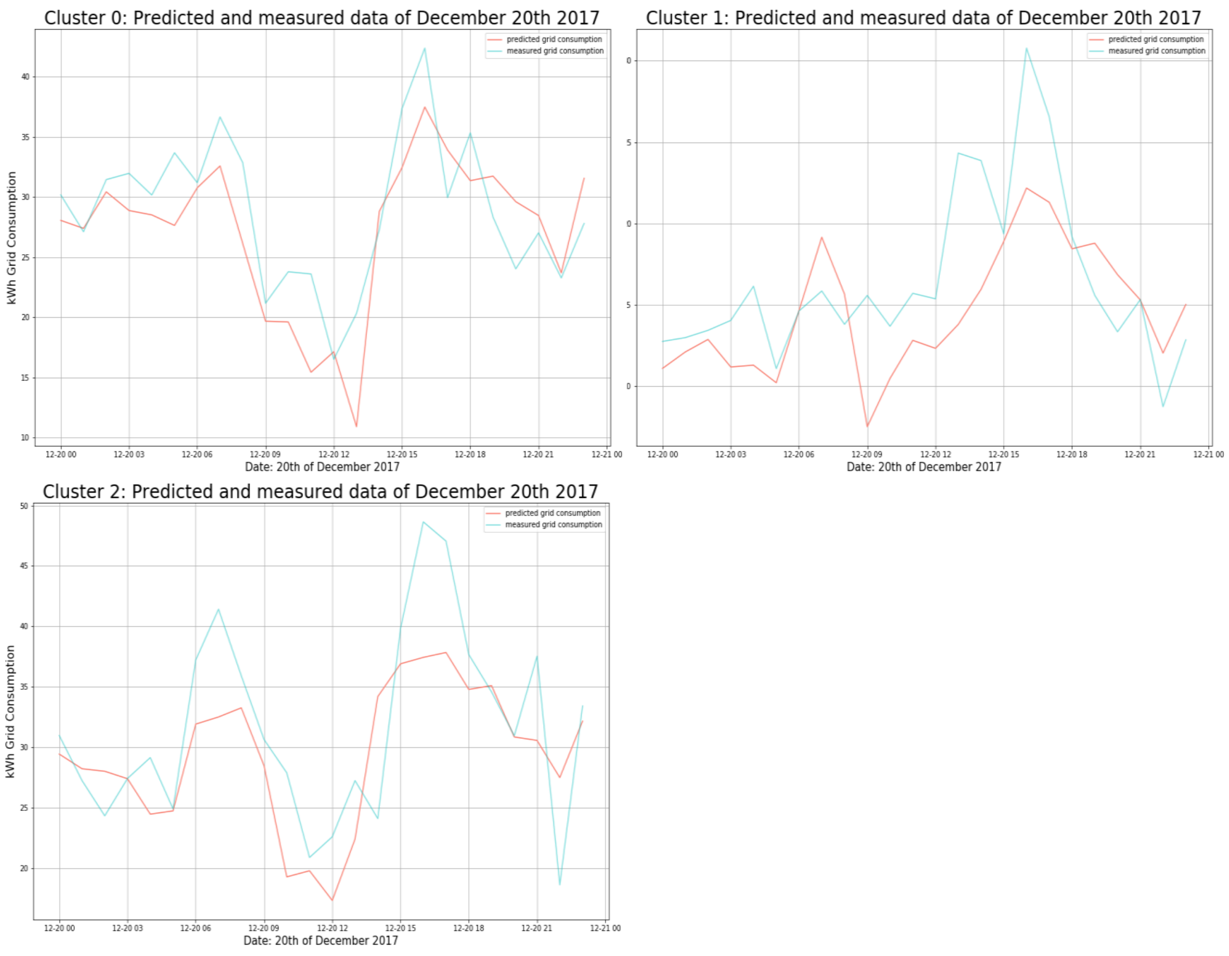


Figure 13 Predicted and measured values validation number-10

A contradicting observation is made in validation number-10, where the error terms are very good, in fact the best of all validations. This, while the corresponding R^2 of the validation is very low. Figure 13 shows the plots of the predicted and measured values of validation number-10 as an example. As can be seen in the plots, the values of the predicted and actual values are almost everywhere different. This explains the bad R^2. Why specifically this day in December is performing different, less accurate, from the other validation days is guessing. As earlier mentioned it probably is correlated with the weather conditions in winter which causes the poor performance, when looking at the R^2.

According to these validations, there is no indication that there is a difference in prediction accuracy between week days and weekend days. There is however a difference in accuracy when looking into seasons. The worst performing validations, validation 1, 9 and 10, all occur in autumn and winter. This indicates that the predicting accuracy is worse during autumn and winter and prediction accuracy improves in spring and summer. This could be correlated with the less consequent energy production by the PV-systems during these months. To statistically determine this hypothesis more validations should be executed, averaged and compared.

Table 15 Number of learners per validation day

Validation nr.	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
Date	February 3rd	March 23rd	April 20th	May 19th	June 19th	July 23rd	August 15th	September 24th	October 16th	December 20th
Weekday	Saturday	Thursday	Thursday	Friday	Monday	Sunday	Tuesday	Sunday	Tuesday	Wednesday
Number of learners	9577	10729	11401	12121	13561	14353	14953	15889	16393	17257
Size Cluster 0	25 *+2*	(1)	25	24	23	25	23	24	(1)	31
Size Cluster 1	(1)	27	16	11	24	16	27	16 *+3*	17	10 *+1*
Size Cluster 2	43	23	18	19	(7)	10	16 *+4*	27	36	28
Size Cluster 3	(1)	19 *+1*	11	16	16 *+7*	19	(4)	(3)	16 *+1*	(1)
Cluster size values between brackets are merged with second smallest cluster										

Further, from these validation days, it can be observed that the number of learners do not impact the model accuracy. There is a significant difference between the number of learners of each validation day, see Table 15. The number of learners is variable since, as mentioned earlier, the training set goes up until the validation data.

Table 16 Total average results

Validation nr.	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	***Total average***
Date	February 3rd	March 23rd	April 20th	May 19th	June 19th	July 23rd	August 15th	September 24th	October 16th	December 20th	
Weekday	Saturday	Thursday	Thursday	Friday	Monday	Sunday	Tuesday	Sunday	Tuesday	Wednesday	
Average R2	0.73	0.91	0.92	0.77	0.82	0.80	0.84	0.82	0.70	0.40	**0.77**
Average cv-RMSE	0.36	0.34	0.33	0.51	0.42	0.46	0.50	0.50	0.55	0.20	**0.41**

Overall the R^2 score averages on 0.77, which proves the model is able to explain the majority of the data decently, but is not in a highly accurate manner. The cv-RMSE tells how well the model fits the data. ASHRAE uses a cv-RMSE of ±0.30 is considered to be a well calibrated model and sufficiently close to physical reality for engineering purposes, when using hourly data (ASHRAE, 2002). As can be seen in Table 16 and Table 14, the range of ±30% is achieved in some of the validation runs. However, on average the cv-RMSE comes to be 41% which is somewhat higher, and indicates that improvements are necessary for practical applications according to the standards of ASHRAE. The error terms MSE, MAE and RMSE are not taken into account in these average numbers since they cannot be used to compare. These terms are correlated with the absolute mean and the number of houses. Normalizing these values by dividing them by the number of houses of the corresponding cluster would still not give a comparable value since the grid consumption is very dependent on the weather conditions. In the winter and autumn the grid consumption is higher due to more heating and less electricity production form the PV-systems.

Table 17 shows the comparison between the predicted values of the excess electricity and how they are distributed according to the operating rules described in 3.3.3. Also it shows the total grid demand for each validation day.

Table 17 Validation results, measured versus predicted values

	Validation nr.	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
	Date	February 3rd	March 23rd	April 20th	May 19th	June 19th	July 23rd	August 15th	September 24th	October 16th	December 20th
	Weekday	Saturday	Thursday	Thursday	Friday	Monday	Sunday	Tuesday	Sunday	Tuesday	Wednesday
Measured	Total excess production	60.97	1581.41	1976.46	2297.73	1857.64	1375.77	905.46	920.20	397.75	0.00
	Direct sharing	0.07	0.00	2.43	5.91	2.56	0.00	0.00	12.29	1.72	0.00
	Indirect sharing day	60.89	443.10	331.72	172.41	141.61	350.36	336.60	342.37	396.03	0.00
	Remaining excess production[1]	0.00	1138.32	1642.30	2119.41	1713.47	1025.41	568.86	565.54	0.00	0.00
	Total grid demand[2]	1453.35	873.32	805.02	270.61	215.78	663.11	545.51	563.34	583.45	1848.61
Predicted	Total excess production	22.77	1109.33	1739.66	1594.07	2021.26	1438.39	805.27	815.35	395.56	0.00
	Direct sharing	5.27	5.09	0.00	0.79	1.00	0.00	9.22	10.81	14.15	0.00
	Indirect sharing day	17.50	408.68	197.01	219.68	150.97	390.83	270.34	273.53	381.40	0.00
	Remaining excess production*	0.00	695.56	1542.66	1373.61	1869.30	1047.56	525.70	531.01	0.00	0.00
	Total grid demand**	1737.72	833.18	719.40	360.59	240.6959	738.28	438.15	441.33	643.79	1706.69

All values are in kWh
[1]*After sharing*
[2]*Without taking into account direct or indirect sharing*

It is assumed, that the ESS is completely discharged at the beginning of the day, at midnight. The 'indirect sharing day' rows, shows the amount of excess produced kWh which is stored in the ESS that can be consumed the same day. In all of the cases, the indirect sharing of electricity takes place late in the afternoon or in the evening, when PV-systems do not produce enough to cover the electricity demand of the clusters.

The total values, total excess production, and total grid demand, of the actual and predicted values are very similar to each other. The values for indirect are also quite similar and can be used to determine the indirect sharing potential of the next day in order of magnitude. The comparison of the measured and predicted direct sharing on the contrary, is less similar and promise currently little practical application.

The total excess production in seven of the ten validations is much higher than the total grid demand. This indicates that the neighbourhood, with sufficient battery capacity is in the majority of the year capable of being self-sufficient energy provision with sufficient battery capacity and even generates a surplus of electricity. To determine the storage size needed, to become completely self-sufficient, the total grid demand between ±19:00 until ±6:00 hour must be accumulated. This is the time period where the PV-systems do little to no production. In order of magnitude, this will be equal to the 'total grid demand' in Table 17. From May until October the neighbourhood can be self-sufficient with a storage system of 660kWh. This is equal to 9.43kWh per dwelling, which is large but not extreme considering battery packs for private individuals offered by companies, are in the range of 15kWh (Panasonic, 2019) (Tesla, 2019).

Even with such a large storage system the neighbourhood produces a vast amount of excess electricity, up to 1500kWh per day in June. The surplus of electricity generated in spring and summer could be used for long-term storage so that is can be used in the winter and autumn where the neighbourhood does need energy to improve the self-sufficiency even more. Some suggestions for long term storage are described in 3.3.3. Currently compensation for feeding excess electricity, on top of the total grid demand, to the grid is roughly 50% of the electricity cost (SwitchExpert, 2019). With current electricity prices of around €0.20/kWh (Pricewise, 2019) this means saving of €0.10 per kWh used from the long term storage. It is expected for the feed-in compensation to decrease rapidly in the near future, which makes storing excess electricity more attractive.

5.2 Proof of concept

In this proof of concept section, a single validation day is described in detail to show the applied methodology and the process of the used approach. Validation number-4 is used since it has results reasonably close to the average.

5.2.1 Data pre-processing

The process starts with the collecting and preparation of smart meter data, time data and meteorological data. Smart meter data and meteorological data both come from different sources, being the smart meters in the houses and meteorological centre. Time data, such as indicators for weekday and season, can be extracted from the smart meter data, this is described in chapter four 4.2.3. Two datasets have to be created, since the clustering model and the prediction model require different data formats, as described in 4.4. The clustering set only consists of the grid consumption per house. The prediction datasets contain all the prediction features.

5.2.2 Clustering Model

After the data collection and pre-processing, clustering is performed. The houses are clustered based on the grid consumption data two weeks preceding the prediction day. In this example the clusters are made based on the grid consumption data of 4th of May to the 18th of May 2017. The prediction will be done for May 19th 2017. The training model will use data from January first 2016, until the 18th of May 2017. This means that the model has not seen any of the data that it will predict, which is necessary for a proper model validation. For these validations, it is assumed that the measured meteorological data is the weather forecast.

The clustering, as described in 3.1, is executed with the Scikit-learn module, resulting in four clusters with a substantial number of houses, see Table 18.

Table 18 Cluster distribution May 2017

Cluster 0	**Cluster 1**	**Cluster 2**	**Cluster 3**
grid_consumption_4776	grid_consumption_4775	grid_consumption_4687	grid_consumption_4696
grid_consumption_4778	grid_consumption_4783	grid_consumption_4689	grid_consumption_4698
grid_consumption_4780	grid_consumption_4791	grid_consumption_4692	grid_consumption_4701
grid_consumption_4781	grid_consumption_4794	grid_consumption_4700	grid_consumption_4707
grid_consumption_4788	grid_consumption_4796	grid_consumption_4703	grid_consumption_4715
grid_consumption_4790	grid_consumption_4804	grid_consumption_4706	grid_consumption_4718
grid_consumption_4793	grid_consumption_4805	grid_consumption_4714	grid_consumption_4735
grid_consumption_4798	grid_consumption_5006	grid_consumption_4716	grid_consumption_4745
grid_consumption_4800	grid_consumption_5013	grid_consumption_4731	grid_consumption_4747
grid_consumption_4807	grid_consumption_5014	grid_consumption_4732	grid_consumption_4752
grid_consumption_4808	grid_consumption_5015	grid_consumption_4733	grid_consumption_4754
grid_consumption_4809	**Total: 11**	grid_consumption_4734	grid_consumption_4757
grid_consumption_4818		grid_consumption_4738	grid_consumption_4758
grid_consumption_4820		grid_consumption_4739	grid_consumption_4759
grid_consumption_4821		grid_consumption_4741	grid_consumption_4760
grid_consumption_4824		grid_consumption_4742	grid_consumption_7056
grid_consumption_4827		grid_consumption_4746	**Total: 16**
grid_consumption_4829		grid_consumption_4756	
grid_consumption_4997		grid_consumption_5010	
grid_consumption_5005		**Total: 19**	
grid_consumption_5008			
grid_consumption_5009			
grid_consumption_5012			
grid_consumption_5016			
Total: 24			

For each cluster, the individual consumption profiles are accumulated. The accumulated sets are used in the prediction model.

5.2.3 Prediction model

As described in 3.2.3, a random forest regressor, an ensembled learning machine from Scikit-learn module, is used for the prediction model. The exact parameter settings of this model, and how these are established, are described in 3.2.3.2, Table 6.

5.2.3.1 Validation results

After preparing the data, clustering the houses and optimizing the parameters, the actual prediction takes place. For each cluster the random forest regressor algorithm is used to predict the validation days. Each validation run uses two datasets, a training set and a test set. The training testing set contains, as mentioned earlier, all the available data up until the validation day. Therefore the number of learners, visible in the subscript of Table 19, differs per validation day. The test set consists of the 24 hours from the validation day. As can be seen in Table 19, the indicators for the model accuracy, described in 3.2.4, are calculated for each cluster.

Table 19 Prediction accuracy per cluster May 19th 2019

Prediction results	*Cluster 0*	*Cluster 1*	*Cluster 2*	*Cluster 3*
Cluster size	24	11	19	16
R2	0.73	0.77	0.75	0.81
R2-adjusted	0.73	0.77	0.75	0.81
MAE	16.22	7.71	9.58	7.53
MSE	524.61	107.17	188.75	101.94
RMSE	22.90	10.35	13.74	10.10
absolute mean[1]	41.55	20.85	26.07	21.15
cv-RMSE	0.55	0.50	0.53	0.48

number of learners = 12121
number of features = 38
[1]Absolute mean from the measured values of the prediction day.

The MAE, MSE and RMSE are all correlated with the absolute mean value, which in general means that higher the absolute mean is, higher the indices are. The absolute mean is again correlated with the number of houses, since more houses generally consume more electricity. The R^2, R^2 – adjusted and the cv-RMSE are normalized and can be used to compare the accuracy of different prediction runs or validation days. It was expected that the cluster with the most houses and the largest absolute mean value would have the best prediction results, since this cluster has most likely the smoothest grid demand profile. However, these results show that the best performing cluster does not have the highest number of houses nor absolute mean. This indicates that clustering the houses based on grid demand profiles does add value for the prediction accuracy.

When looking at the R^2 and the cv-RMSE, of this validation day, there is little variance between the clusters. This is also true for the MAE, MSE and the RMSE, when taking the corresponding absolute mean into account. This is an indicator that the robustness of the model is decent.

Even though the cluster sizes, and thus the values, are very different, the model operates similar based on these indices.

In Figure 14 a plot of the predicted and actual values is shown. This plot shows that the predicted grid consumption line follows the same trend as the measured grid consumption line. However, there are some large discrepancies between the two lines, especially in the middle of the graph, between 9 in the morning and 2 in the afternoon, which results in the error term values given in Table 19.

The plots of the other three clusters can be found in Appendix VI- Prediction and measured values plots. It is clearly visible that the prediction line follows the same trend as the actual demand profile. Although the trends are similar, the values of the predicted and measured profiles differ at almost every point.

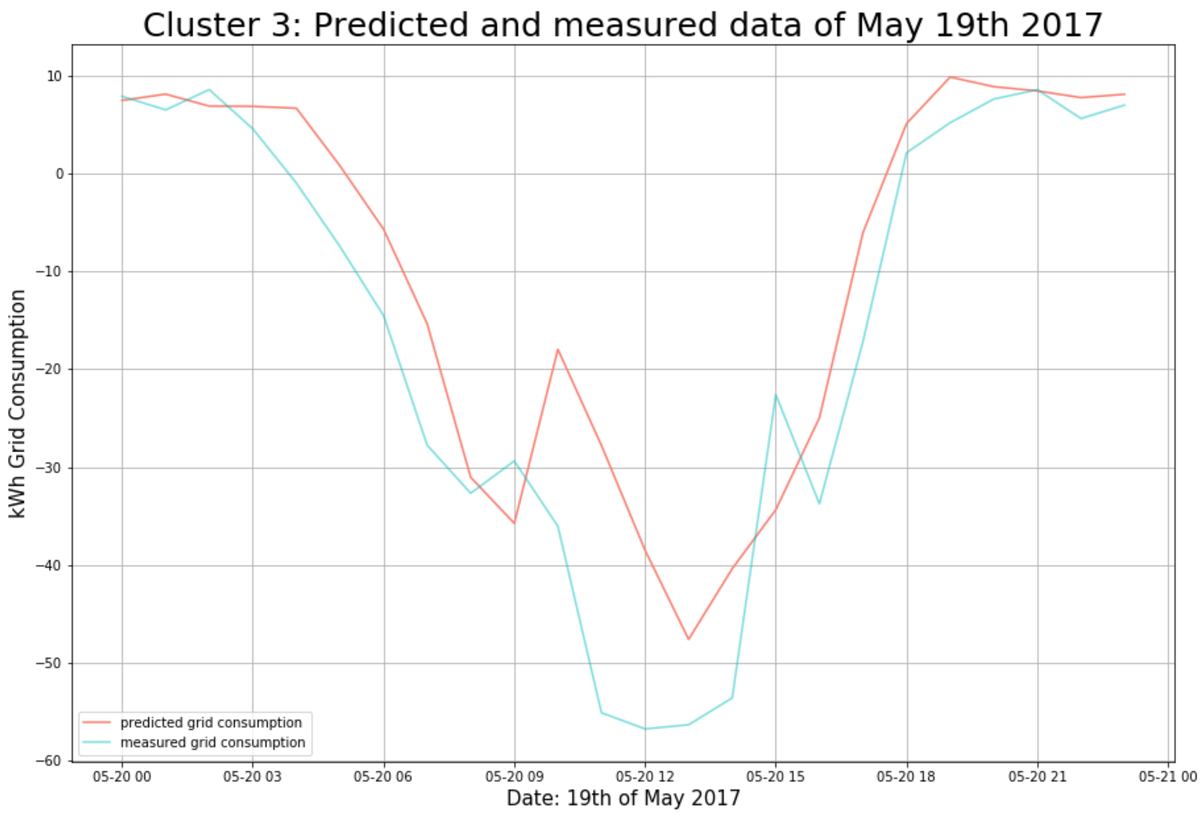


Figure 14 Prediction and measured values plot

This plot, together with the plots in Appendix VI – Prediction and measured values plots, show that the prediction model has a tendency to generate lower grid demand values. This is remarkable, yet the specific cause for this behaviour is uncertain.

The features used in the prediction model are described in 4.2. Table 20 show the five most important prediction features per clusters.

Table 20 Top-5 feature importance validation day 4

Cluster 0		Cluster 1		Cluster 2		Cluster 3	
Feature name	**Importance**	**Feature name**	**Importance**	**Feature name**	**Importance**	**Feature name**	**Importance**
tot_gridconsumed ts-1day	0.5548	tot_gridconsumed ts-1day	0.5698	tot_gridconsumed ts-1day	0.5357	tot_gridconsumed ts-1day	0.4561
Humidity	0.1059	Humidity	0.1069	Humidity	0.1222	tot_gridconsumed ts-3days	0.1701
tot_gridconsumed ts-2days	0.1038	tot_gridconsumed ts-2days	0.0916	Temperature	0.1096	Humidity	0.1308
tot_gridconsumed ts-3days	0.0521	Temperature	0.0396	tot_gridconsumed ts-3days	0.0754	tot_gridconsumed ts-2days	0.0631
Temperature	0.0341	tot_gridconsumed ts-3days	0.0371	tot_gridconsumed ts-2days	0.0423	Temperature	0.0613

All four clusters have the same top-five prediction features, only the order is slightly different. The autoregressive feature, ‘tot_gridconsumed ts-1day’, is the most important feature for all the clusters in this validation example. This shows that the previous day’s consumption is a very strong predictor. It is remarkable that in cluster 3, the feature ‘tot_gridconsumed ts-3days’ has a much stronger influence on the prediction accuracy than ‘tot_gridconsumed ts-2days’, in contrary with the other three clusters. Besides the autoregressive features, the meteorological prediction features are important. Humidity and temperature are strong predictors in all clusters. The complete feature importance list for the prediction of 19th of May 2017, can be found in Appendix IV – Feature importance. The fact that the importance of the prediction features is different per cluster, proves that the clusters have different characteristics, and thus that the clustering is beneficial for the predicting accuracy.

5.2.4 Operating system

Please take the assumptions regarding the electricity grid, described in 3.3, in mind. Now that the grid demand values are generated by the prediction model, the sharing potential can be determined. Table 21 shows the grid demand according to the prediction model, as well as the actual values, of May 19th 2017. The goal of the operating system is to determine direct, and indirect sharing possibilities.

Table 21 Predicted and measured values validation

	Cluster 0	*Cluster 1*	*Cluster 2*	*Cluster 3*	*#*	*Cluster 0*	*Cluster 1*	*Cluster 2*	*Cluster 3*	*#*
TS	**Predicted**					**Measured**				
2017-05-19 00:00:00	8.78	4.57	7.06	7.46	(I)	8.74	2.43	6.81	7.88	(I)
2017-05-19 01:00:00	9.50	4.48	8.37	8.10		6.81	2.45	7.18	6.49	
2017-05-19 02:00:00	9.40	5.31	8.80	6.88		7.15	4.98	6.40	8.56	
2017-05-19 03:00:00	8.40	4.61	8.58	6.85		6.32	2.68	8.76	4.56	
2017-05-19 04:00:00	8.68	3.02	4.61	6.66		-2.61	-2.29	-2.13	-0.99	(III)
2017-05-19 05:00:00[1]	0.01	-0.98	-0.33	0.77	(II)	-17.81	-11.01	-13.91	-7.47	
2017-05-19 06:00:00	-21.70	-13.13	-3.68	-5.69	(III)	-43.77	-23.60	-29.41	-14.53	
2017-05-19 07:00:00	-38.31	-21.14	-29.45	-15.35		-53.31	-29.39	-34.69	-27.76	
2017-05-19 08:00:00	-44.72	-23.84	-33.05	-31.05		-65.93	-31.96	-34.43	-32.68	
2017-05-19 09:00:00	-55.37	-25.92	-42.33	-35.76		-37.41	-24.31	-40.34	-29.38	
2017-05-19 10:00:00	-32.26	-19.05	-15.23	-17.97		-76.51	-36.18	-48.68	-36.05	
2017-05-19 11:00:00	-43.18	-27.64	-37.21	-27.80		-120.60	-57.97	-67.22	-55.11	
2017-05-19 12:00:00	-81.00	-39.89	-42.83	-38.50		-109.02	-55.00	-66.08	-56.76	
2017-05-19 13:00:00	-85.30	-40.87	-48.57	-47.61		-111.25	-52.22	-63.49	-56.34	
2017-05-19 14:00:00	-69.53	-31.14	-39.75	-40.39		-96.44	-45.00	-59.12	-53.59	
2017-05-19 15:00:00	-55.85	-23.05	-38.45	-34.36		-70.98	-35.82	-28.56	-22.60	
2017-05-19 16:00:00	-40.84	-17.12	-26.41	-24.97		-54.60	-27.99	-40.99	-33.76	
2017-05-19 17:00:00	-26.52	-14.05	-18.88	-6.05		-39.63	-19.79	-17.15	-17.17	
2017-05-19 18:00:00	6.38	2.88	7.58	5.10	(IV)	-10.99	-6.63	3.79	2.12	(II)
2017-05-19 19:00:00	11.67	4.51	15.40	9.84		7.23	5.12	11.65	5.18	(IV)
2017-05-19 20:00:00	14.13	5.51	14.79	8.86		15.45	8.19	14.24	7.60	
2017-05-19 21:00:00	14.32	5.47	12.40	8.43		16.66	4.49	9.63	8.55	
2017-05-19 22:00:00	11.67	4.78	13.53	7.75		8.62	7.51	5.93	5.61	
2017-05-19 23:00:00	11.38	4.44	10.76	8.08		9.34	3.42	5.10	6.98	

All values are in kWh

[1]*official sunrise on 19th of May 2017 on 5:40 in the morning (sunrise-and-sunset.com, 2019)*

It is assumed that the storage system is empty at the start of the day. The values of the prediction model and the measured data, show similar patterns. The day has four different sections, indicted by the roman numbers in Table 21. The first indicator, number I, is a demanding stage of the neighbourhood. This is before sunrise, where the PV-systems don't produce, electricity to cover the demand. Dependent on the status of the ESS either condition rule *1* or *2* should be executed, see 3.3.3. It would be preferable to have available stored electricity from previous' day excess electricity, so condition rule *2* is applicable.

Number II, indicates a state of the neighbourhood where direct sharing is possible. Direct sharing possibilities occur when there is excess electricity generated by the PV-systems, in certain cluster(s), and demand in other. An example of this can be found in Table 21, in the prediction columns at 5:00 in the morning. Clusters 0 and 3 have demand for electricity while clusters 1 and 2 have excess. Another example can be found on 18:00 in the afternoon in the measured columns. Clusters 0 and 1 generate a surplus of electricity, and cluster 2 and 3 are in a demanding state. In these cases the electricity can be transferred directly between the clusters. Unfortunately, the direct sharing potential, generated by the prediction model does not occur on the same moment nor in the same quantity as the actual values demonstrate. In this part condition rules *3* and *4* are applicable, dependent on the clusters.

The third indicator, number III, indicates the period where excess electricity is generated. In all clusters the electricity consumption is larger than the consumption of electricity. This electricity can be transported to the storage system to be consumed later. Condition rule number *5* has to be applied for all clusters in this section of the day.

Number IV indicates that the neighbourhood is, again, in a demanding state. However, this time the electricity that was previously generated can be consumed, this is referred to as indirect sharing. Excess electricity, which cannot be used immediately in the neighbourhood, is stored in a storage system. In the validation example, Table 21, this occurs in both the predicted and measured columns. The predicted values indicate excess electricity between 5:00 in the morning until 17:00 in the afternoon. Accumulated this excess electricity comes to be 1594 kWh. After 17:00 in the afternoon, the neighbourhood is again in demanding state, where between 18:00 and midnight 219kWh is needed. This can easily be covered by the excess generated electricity of that day. The measured values show similar patterns however, in different quantities. The condition rule applicable in this part of the day is *2*, using ESS power.

As also stated in Table 21, the sunrise, the moment the sun becomes visible at the eastern horizon, on 19th of May 2017 occurs on 5:40 AM (sunrise-and-sunset.com, 2019). However, as can be seen in the measured data columns, the solar panels already start generating electricity in the hour 4:00 to 5:00 AM. Before the sun itself is visible at the horizon, its light can already be observed and, as the smart meter data shows, they generate electricity.

Please note that efficiency loss due to transport through cables, and the storage is not taken into account in current results.

As can be observed from the values in Table 21, the total grid demand of the neighbourhood is negative, which means that there is an excess of electricity. Figure 15 shows the distribution of the excess electricity for this validation day. The validation only covers the distribution of one day.

The surplus of generated electricity can be used in three possible ways, and should be prioritized in the following order.

1. Direct sharing; it is desirable to, when possible, use the excess electricity for direct sharing first. This minimizes the loss from transporting, and reduces grid demand.
2. Indirect sharing; if direct sharing is not possible, short term storing and to use later in the neighbourhood. This decreases the grid demand and increases self-sufficiency.
3. Remaining excess. Two main possibilities for the remaining excess generated electricity.
    a. Trading; excess electricity can be traded in the demand response market.
    b. Long-term storage; the neighbourhood is generating a vast amount of excess electricity. The surplus is, in some cases, so large, that it is not realistic to store the remaining electricity, even of a single day, in an electrical storage systems, due to the high purchasing costs. Long-term storage in hydrogen, despite the low efficiency, could be an option.
    c. Send back to the grid.

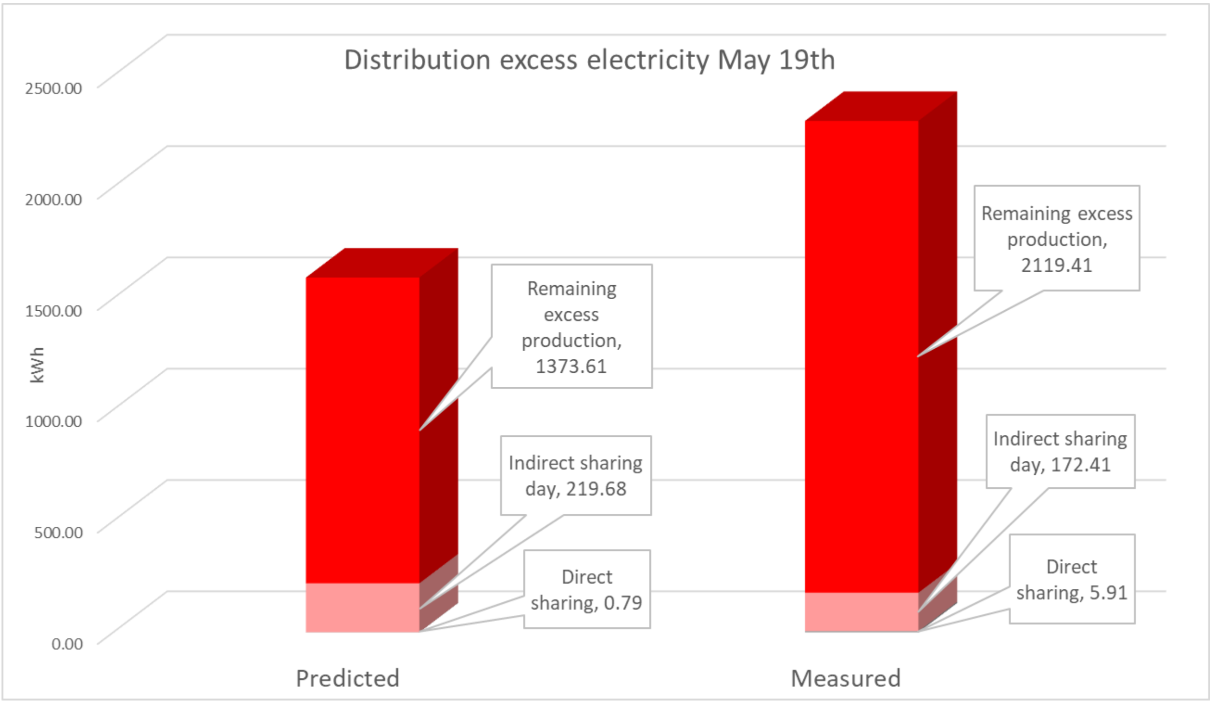


Figure 15 Total excess electricity production distribution

According to the prediction rules, suggested in 3.3.3, the following operating rules should be executed. Table 22 shows a more exact description of which clusters share electricity compared to Table 21.

Table 22 Operation schedule validation May 19th 2017

	Cluster 0	*Cluster 1*	*Cluster 2*	*Cluster 3*	*Operation commands*
TS	**Predicted**				
2017-05-19 00:00:00	8.78	4.57	7.06	7.46	Use grid power[1]
2017-05-19 01:00:00	9.50	4.48	8.37	8.10	
2017-05-19 02:00:00	9.40	5.31	8.80	6.88	
2017-05-19 03:00:00	8.40	4.61	8.58	6.85	
2017-05-19 04:00:00	8.68	3.02	4.61	6.66	
2017-05-19 05:00:00	Receive 0.01 from C1	0.01 to C0 0.77 to C3 0.20 charge ESS	0.33 charge ESS	Receive 0.77 from C1	← Operate per cluster
2017-05-19 06:00:00	-21.70	-13.13	-3.68	-5.69	Charge ESS[2]
2017-05-19 07:00:00	-38.31	-21.14	-29.45	-15.35	
2017-05-19 08:00:00	-44.72	-23.84	-33.05	-31.05	
2017-05-19 09:00:00	-55.37	-25.92	-42.33	-35.76	
2017-05-19 10:00:00	-32.26	-19.05	-15.23	-17.97	
2017-05-19 11:00:00	-43.18	-27.64	-37.21	-27.80	
2017-05-19 12:00:00	-81.00	-39.89	-42.83	-38.50	
2017-05-19 13:00:00	-85.30	-40.87	-48.57	-47.61	
2017-05-19 14:00:00	-69.53	-31.14	-39.75	-40.39	
2017-05-19 15:00:00	-55.85	-23.05	-38.45	-34.36	
2017-05-19 16:00:00	-40.84	-17.12	-26.41	-24.97	
2017-05-19 17:00:00	-26.52	-14.05	-18.88	-6.05	
2017-05-19 18:00:00	6.38	2.88	7.58	5.10	Use ESS power
2017-05-19 19:00:00	11.67	4.51	15.40	9.84	
2017-05-19 20:00:00	14.13	5.51	14.79	8.86	
2017-05-19 21:00:00	14.32	5.47	12.40	8.43	
2017-05-19 22:00:00	11.67	4.78	13.53	7.75	
2017-05-19 23:00:00	11.38	4.44	10.76	8.08	

All values are in kWh
[1]*Assuming storage system is empty, otherwise ESS will be discharged first.*
[2]*When ESS is fully charged, electricity will be put to long-term storage, or send back to the grid immediately.*

As also mentioned in the table above, the excess energy, after completely storing the ESS, could be stored for longer periods.

6. Conclusion and outlook

This chapter forms the final part of this research. The research questions will be answered and the validity of results will be discussed. Also the limitations of this study will be addressed and some recommendations will be given for future research on energy demand prediction and determining sharing potential.

6.1 Conclusion

The decentralization of the electricity grid, demands energy storage and smart energy distribution (Li & Chan, 2017). High penetration levels of intermittent energy sources, such as PV-systems and wind turbines, has increased the complexity of electricity demand forecasting (Lahouar & Slama, 2015). Smart metering has, however, given the opportunity to analyse the energy usage profiles of individual houses more thoroughly. In this study, smart metering data is analysed to expose opportunities for increasing self-sufficiency.

In order to increase self-sufficiency it is necessary to use renewable generated electricity locally, this can be done by directly sharing excess electricity between clusters, and by storing excess electricity for later use. To achieve such sharing and storing system an approach is suggested which combines different algorithms to achieve day ahead hourly electricity demand profiles of a residential district.

The novelty of this approach is using a clustering algorithm before running the prediction model. Houses with similar electricity demand profiles are clustered based on their electricity demand profiles by means of the k-means clustering technique. This method minimizes the within cluster variation based on the Euclidian distance. Empirical testing has pointed out that the demand profiles of individual houses change throughout the year, which demands for flexible clusters changing over time. Every prediction day the clustering algorithm will be run, based on the two preceding weeks, to create suitable clusters. The prediction accuracy does not increase at larger clusters, which one should usually expect. The ratings of feature importance per cluster for a single validation day differ from each other. This indicates that the clusters show different characteristics in the data, not only based on their electricity demand but also in other prediction variables. Given this information it can be concluded that the clustering adds value to the overall prediction accuracy.

The selection of the best suitable prediction model is based upon a literature review as well as empirical test. The literature review points out that there are multiple broadly used algorithms which, dependent on the case specifics, are able to accurately predict electricity demand. Most used models are, Support vector machines, Regression Trees and Artificial Neural Networks. In this empirical test a set of different algorithms is used to predict the electricity demand for several houses. The literature study, together with the empirical test has pointed out that the ensembled regression trees have the possibility to accurately predict the neighbourhoods' grid demand, given the provided data.

The prediction is executed in python with Scikit-learns' Random Forest Regressor, from their Ensembled Learning Machines module. The parameters of the random forest regressor are optimized by using a grid search algorithm which tests large variety of parameter setting combination, from the Scikit-learn module. The model is validated by predicting 24 hours of electricity demand per cluster with, 'never seen before' data. The suggested model, is capable of predicting the clusters' next day electricity demand, with an average R^2 score of 0.77 and a cv-RMSE of 0.41, based on the conducted validations. According the ASHRAE standards some improvements are necessary for practical implications (ASHRAE, 2002). Most important prediction features is the electricity consumption of the previous day on the same time. Other important features are, humidity, electricity consumption two and three days previous on the same time and ambient temperature.

In summer and spring the prediction accuracy is higher, R^2 score of 0.84 and cv-RMSE of 0.43, compared to the winter and autumn, R^2 score of 0.66 and cv-RMSE of 0.40. This is probably due to the differences in weather conditions. In summer and spring the PV generated electricity is more stable and constant, where in winter it is very dependent on the cloud coverage. Also the electricity usage in winter and autumn is higher due to the HP which provides heating in the dwellings.

The results show that the number of learners, the data sample size, does not influence the prediction accuracy. This is in contrary with the general known principle that ML models improve when the data sample increases. This observation could be used to improve the model, by selecting only the most relevant data, of the weeks similar to the prediction day. This is further described in the recommendation section, 0.

In its current form, the prediction model is capable of predicting the trend of the electricity demand. However, it is not accurate enough to predict the actual hourly demand so that an operating schedule can be determined for the coming 24 hours. It could however be used to indicate the amount of excess electricity at the end of the day. This can be very useful when trading for demand response, or deciding to use excess electricity for long-term storage.

Based on the validations conducted in this research, it can be stated that the direct sharing potential is low. The indirect sharing however is very high. With a suitable ESS the neighbourhood could be self-sufficient majority of the days of the year, assuming the validation days are representative for the month. Based on the validation days it is assumed that the neighbourhood can be completely self-sufficient for seven out of twelve months with a storage system capacity of 660kWh. Having long term storage available would increase the self-sufficiency even more, as the excess electricity largely exceeds 660kWh in spring and summer.

6.2 Discussion

This research contributes in the field of energy demand prediction as it focused on small energy consuming entities with renewables on-site. Where the majority of the publication in this field of research, compare different prediction models, this study combines two models in order to increase the prediction accuracy. Further the study, has suggested an approach to increase energy self-sufficiency of residential districts. When implemented this could help accelerating the energy transition towards a more self-sustaining society.

It is known that predicting hourly residential electricity demand is a very difficult task (Lahouar & Slama, 2015). The suggested approach, which combines k-means clustering and a random forest regressor, enables predicting the one day ahead hourly electricity demand trends fairly accurate. However the specific values do deviate too much to create upfront operating rules for sharing and storing energy on hourly basis. A critical discussion of the used model, its features and the operation system is given.

When starting this research, the aim was to predict the energy demand for individual households. However, after trying many different models and settings the prediction would not reach appropriate accuracy. This probably has two main reasons. (I) Unpredictable spikes in the demand profile. Individual houses have demand profiles with many spikes, compared to the smoother graph when multiple houses are combined. The range of demand for one household is very small, which makes the trend sensitive for changes. These spikes are probably caused by individual appliances, e.g. dryer, water cooker, vacuum cleaner. This brings us to the second reason for poor prediction in individual cases. (II) No user specific data. To make the step to individual household prediction other prediction features seem necessary. Adding information about the occupancy of the rooms of the dwelling combined with the appliances in each room, could lead to more accurate individual household predictions. Since individual households don't have smooth demand graphs and no specific user data is available, the prediction of individual households was not feasible in this study.

It has to be taken into account that the meteorological features are now based on measured weather conditions. When bringing this approach to practice, the meteorological features will be based on weather forecasts. This will have influence on the accuracy and reliability of the model.

The prediction could possibly be improved by adding occupancy data, since it can indicate more precisely when electricity will be used. This could be implemented via sensors in the houses which indicate the number of people in a dwelling. Also appliance ownership (including electric vehicles) could help improve the model accuracy, as this is used in some recent studies. A questionnaire could be used to obtain this data. Lastly, since the grid demand is very dependent on the amount of generated electricity by the PV system, solar radiation could be adopted in the prediction features.

As many studies have pointed out, see Literature review, removing the holidays and solely focussing on non-holidays, could lead to improved model fitting. However this would not have matched with the goal of suggesting a practically applicable method, which includes all type of days.

The best suitable ESS size, should be designed carefully. In this study the ESS size is solely based on the 10 validation days. An over dimensioned ESS brings high capital investment which might not be feasible. On the other hand, undersized ESS may not be able to provide the desired operational and financial benefits. Also, the size of the ESS is important for the effectiveness of frequency regulation (Gao D. , 2015). So it is very important to determine a suitable battery size.

Currently, the Dutch government has an arrangement, 'salderingsregeling', for subtracting the fed back electricity from the total grid usage in operation (Wiebes & Snel, Omvorming Salderen, 2019). This makes the feed-in compensation for excess electricity, up until the total electricity demand, equal to the cost price at the electricity supplier up until 2024. This makes purchasing and using a large ESS financially unattractive. This regulation is expected to be completely reduced to zero by the year 2031 (Wiebes & Snel, Omvorming Salderen, 2019). What exact regulations regarding feed-in compensations will be applicable by then is uncertain.

6.3 Limitations and recommendations for future research

Throughout the research some bottlenecks, shortcomings and opportunities have come across. These are translated in the following recommendations for future research regarding energy demand prediction and energy sharing.

- This research has viewed only single days for validation. It seems that on many of the validation days, the excess electricity is enough to cover the grid demand the next day, before the PV systems start generating electricity again. Looking over a longer period of time, at least two days, to see how much storage really can be used in the neighbourhood and how much surplus is there for long-term sharing and demand response trading will be useful. This would also give a better view on the actual increase of self-sufficiency of the neighbourhood. For this, automation of the validation process would be preferred.
- Predicting electricity demand and electricity production separately like in the study of (Kneiffel & Webb, 2016) is preferred. It is generally known that the importance of prediction variables for both consumption and prediction differ quite a lot. The production values are solely dependent on the weather conditions, where the consumption values are dependent on both user behaviour and weather conditions. When looking at the 'raw' consumption and production patterns, great differences can be observed. So separately predicting each pattern and later merge them could lead to increased model fitting.
- The autoregressive features of previous day seems to be the most important prediction variable. The prediction accuracy would improve if the autoregressive variables could be even with a shorter interval, ts-1hour. This could have a large impact on the prediction model. However this is not possible when predicting for one day ahead. The model could be reformed to predict the electricity demand for one hour ahead and possibly be more precise, taking both the very short term, ts-1hour, and the more long-term, historical consumption patterns into account.
- It would be useful to more thoroughly test multiple optimized prediction algorithms. The empirical test conducted in this study used default settings of some widely used models. The literature study has pointed out that there are large differences, in model accuracy, between standard and optimized models. Despite the fact that they are very common in the literature review, this study has not tested ANN related models in the empirical test for selecting a prediction model. This is due to the set time span of this study, of six months, and the needed extensive knowledge and experience to setup an optimized ANN model.
- The suggested operating rules are only suggested steps to improve energy self-sufficiency based on logic. The operating suggestions can be extended to a more advanced set of rules when taking dynamic energy pricing and trading for demand response into account. Dynamic pricing, as currently applicable in the case, gives opportunities to maximize profit/minimize costs for the usage of an ESS. To improve the financial feasibility of the ESS, the operating system could be optimized regarding

the different tariffs, charging during low tariff and discharging during high tariff. A multi objective linear optimisation program where the energy sharing is maximized while minimizing the costs could be a useful alternative. The exact application of such optimisation model is, of course, dependent on the intentions of the system owner.

- The direct sharing is very low when compared to the total grid demand. When looking at a shorter interval, 15-minutes instead of hourly might change this. However, this could also make the prediction accuracy worse.
- The operating system could be used with real time data instead of the predicted data, this assures using correct values for the sharing. This probably requires different network and electrical circuit setup.
- It is possible to use the model to see what influences climate change has, on the electricity consumption of the neighbourhood. Since the model does predict the electricity demand trends accurately. By changing the meteorological features, temperature and humidity, the change in electricity demand becomes visible. For example, running multiple prediction models with actual temperature and with increased temperature (0.5, 1.0, 1.5 and 2.0 degrees Celsius) and compare the demand profiles.
- Currently the prediction model uses all the available data, up until the training day. Using only training data of similar months and seasons, 6-10 weeks preceding on the prediction day, might improve the accuracy. The results of this study points out that the most resent consumption data is the most important for the prediction results.

Acknowledgement

Royal BAM group, provided the dataset for this study as it is one of the funders of the 'Smart Energy Systems in the Built Environment' (SES-BE) research program. This program, hosted by Eindhoven University of Technology, focuses on the development of management and control systems, the development of emerging hardware technologies to be used in buildings, the ICT-enabled energy services (monitoring and prediction, data mining, market-based and real-time control), as well as technology exploitation via new business models and a realistic simulation environment (modelling lab) for testing entrepreneurship ideas and government policies (SES-BE, 2019).

This page was intentionally left blank.

Bibliography

ACM, A. C. (2015). *Netcode Elektriciteit.* Authoriteit, Consument & Markt.

Arregi, B., & Garay, R. (2017). Regression analysis of the energy consumption of tertiary buildings. *Energy Procedia*, 9 - 14.

ASHRAE. (2002). *Measurement of Energy and Demand Savings.* Tullie Circle, Atlanta, USA: ASHRAE.

Barton, J., & Infield, D. (2004). Energy Storage and its use with Intermittend Renewable Energy. *Loughborough University, UK*.

Beckel, C., Sadamori, L., Staake, T., & Santini, S. (2014). Revealing household characteristics from smart meter data. *Energy*, 397 - 410.

Bernards, R. (2018). *Smart Planning; Integration of statistical and stochastic methods in distribution nettwork planning.* Drunen.

Biwas, R., Robinson, M., & Fumo, N. (2016). Prediction of residential building energy consumption: A neural network approach. *Energy*, 84-92.

Bouare, O. (2008). Impact of Global Warming on Rural-Urban Migration and Net Emigration in Forefront Sub-Saharan Countries. *2 African Journal of Public Affairs*.

CBS. (2018). *Hernieuwbare energie in Nederland 2017.* Den Haag: Centraal Bureau voor de Statistiek.

Costa, A., Keane, M., Torrens, J., & Corry, E. (2013). Building Operation and Energy Performance: Monitoring, analysis and optimisation toolkit. *Applied Energy*, 310 - 316.

Cucchiella, F., D'Adamo, I., & Gastaldi, M. (2017). The Economic Feasibility of Residential Energy Storage Combined with PV Panels: The Role of Subsidies in Italy. *Energies*.

Darbellay, G., & Slama, M. (2000). Forecasting the short-term demand for electricity. Do neural networks stand a better chance? *International Journal of Forecasting*, 71–83.

DeLong, E., DeLong, D., & Clarke-Pearson, D. (1988). Comparing the Areas under Two or More Correlated Receiver Operating Characteristic Curves: A Nonparametric Approach. *Biometrics*, 837-845.

Dong, B., Cao, C., & Lee, S. (2005). Applying support vector machines to predict building energy consumption in tropical region. *Energy and Buildings*, 545 – 553.

Fan, C., Xiao, F., & Zhao, Y. (2017). A short-term building cooling load prediction method using deep learning algorithms. *Applied Energy*, 222–233.

Fumo, & Biswas. (2015). Regression analysis for prediction of residential energy consumption. *Renewable and Sustainable Energy Reviews*, 332-343.

Gao, D. (2015). *Energy Storage for Sustainable Microgrid.* London: Elsevier.

Gao, X., & Wang, X. (2017). Impacts of Global Warming and Sea Level Rise on Service Life of Chloride-Exposed Concrete Structures. *Sustainability*.

HaskoningDHV, R. (2018, Dec 21). *NEARLY ZERO-ENERGY BUILDINGS - MORE INFORMATION*. Opgehaald van Royal HaskoningDHV: https://www.royalhaskoningdhv.com/en-gb/services/a-z-services/nearly-zero-energy-buildings-more-information/394

Houimli, R., Zmami, M., & Ben-Salha, O. (2019). Short-term electric load forecasting in Tunisia using artificial neural networks. *Energy Systems*.

Hunter, J., Dale, D., Firing, E., Droettboom, M., & team, M. D. (2019, May 5). *matplotlib - home*. Opgehaald van matplotlib: https://matplotlib.org/#

Huo, J., Shi, T., & Chang, J. (2016). Comparison of Random Forest and SVM for Electrical Short-term Load Forecast with Different Data Sources. *IEEE International Conference on Software Engineering and Service Science*, 1077 - 1080.

James, G., Witten, D., & Hastie, T. T. (2017). *An Introduction to Statistical Learning, with applications in R.* New York: Springer.

Junker, G., Azara, A., Lopes, R., Lindberg, K., Reynders, G., Relan, R., & Madsen, H. (2018). Characterizing the energy flexibility of buildings and districts. *Applied Energie*, 175–182.

Jurado, S., Peralta, J., Nebot, A., Mugica, F., & Cortez, P. (2013). Short-term Electric Load Forecasting Using Computational Intelligence Methods. *IEEE International Conference on Fuzzy Systems*.

Kantor, I., Rowlands, I., Parker, P., & Lazowski, B. (2015). Economic feasibility of residential electricity storage systems in Ontario, Canada considering two policy scenarios. *Energy and Buildings*, 222 - 232.

Kneiffel, J., & Webb, D. (2016). Predicting energy performance of a net-zero energy building: A Statistical Approach. *Applied Energy*, 468–483.

KNMI. (2019, April 1). *Kennis & uitleg, Zomer*. Opgehaald van KNMI: https://www.knmi.nl/kennis-en-datacentrum/uitleg/zomer

Kontokosta, C., & Tull, C. (2017). A data-driven predictive model of city-scale energy use in buildings. *Applied Energy*, 303–317.

Kuo, P., & Huang, C. (2018). A High Precision Artificial Neural Networks Model for Short-Term Energy Load Forecasting. *Energies*.

Lahouar, A., & Slama, J. (2015). Day-ahead load forecast using random forest and expert input selection. *Energy Conversion and Management*, 1040–1051.

Li, P. (2008). Energy Storage Is the Core of Renewable Energy Technologies. *IEEE NANOTECHNOLOGY MAGAZINE*.

Li, P., & Chan, C. (2017). *Thermal Energy storage analysis and design.* Arizona (USA): Elsevier.

Li, Q., Meng, Q., Cai, J., Yoshino, H., & Mochida, A. (2009). Predicting hourly cooling load in the building: A comparison of support vector machine and different artificial neural networks. *Energy Conversion and Management*, 90–96.

Li, Q., Ren, P., & Meng, Q. (2010). Prediction Model of Annual Energy Consumption of Residential Buildings. *Institute of Electrical and Electronics Engineers*, 223 - 226.

Masson, G., & Kaizuka, I. (2018). *Trends 2018 in photovoltaic applications. Survey Report of Selected IEA Countries between 1992 and 2017.* Sweden: International Energy Agency.

Mocanu, E., Nguyen, P., Gibescu, M., & Kling, W. (2016). Deep learning for estimating building energy consumption. *Sustainable Energy, Grids and Networks*, 91 - 96.

NUMFOCUS. (2019, March 27). *About NumPy*. Opgehaald van Numpy: http://www.numpy.org/#

Panasonic. (2019, 05 23). *Panasonic - Energy Solutions - Battary Storage - Harbor Plus.* Opgehaald van Harbor Plus™ Smart Battery Storage System: https://na.panasonic.com/us/energy-solutions/battery-storage/battery-storage/harbor-plustm-smart-battery-storage-system

Pandas. (2019, March 27). *The pandas project*. Opgehaald van Pandas: https://pandas.pydata.org/about.html

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 2825--2830.

Pelc, Araújo, Bell, Blanchard, Bonebrake, Chen, . . . Colwell. (2017). Biodiversity redistribution under climate change: Impacts on ecosystems and human well-being. *Science journals AAAS*.

PFS, P. F. (2019, March 27). *About*. Opgehaald van Python Foundation Software: https://www.python.org/about/

Pricewise. (2019, 07 10). *energieprijzen kWh-prijs*. Opgehaald van pricewise: https://www.pricewise.nl/energieprijzen/kwh-prijs/

Reddy, T. A., Saman, N. F., Claridge, D. E., Haberl, J. S., Turner, W. D., & Chalifoux, A. T. (1997). Baselining methodology for facility-level monthly energy use - Part 1: theoretical aspects. *ASHRAE Transactions*, 336-347.

Reynders, G., Lopes, R., Marszal-Pomianowska, A., Aelenei, D., Martins, J., & Saelens, D. (2018). Energy flexible buildings: An evaluation of definitions and quantification methodologies applied to thermal storage. *Energy & Buildings*, 372–390.

Reynolds, C., & Fels, M. (1988). Reliability Criteria for weather Adjustment of Energy Billing Data. *Center for Energy and Environmental Studies* , 236 - 251.

Robinson, C., Dilkina, B., Hubbs, J., Zhang, W., Guhathakurta, S., Brown, M., & Pendyala, R. (2017). Machine learning approaches for estimating commercial building energy consumption. *Applied Energy*, 889–904.

Rodrigues, F., Cardeira, C., & Calado, J. (2014). The daily and hourly energy consumption and load forecasting using artificial neural network method: a case study using a set of 93 households in Portugal . *Energy Procedia*, 220 – 229 .

RVO. (2018). *Electrisch Vervoer in Nederland* . Den Haag: Rijksdienst voor Ondernemend Nederland.

RVO. (2019, January 4). *Beleid electrisch rijden*. Opgehaald van rvo duurzaam ondernemen electrisch rijden : https://www.rvo.nl/onderwerpen/duurzaam-ondernemen/energie-en-milieu-innovaties/elektrisch-rijden/beleid-elektrisch-rijden

Ryu, S., Noh, J., & Kim, H. (2016). Deep Neural Network Based Demand Side Short Term Load Forecasting . *Energies*.

Scikit-learn. (2019, June 26). *RandomForestRegressor*. Opgehaald van scikit-learn: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

SES-BE. (2019, May 10). *Summary of the SES-BE program*. Opgehaald van SES-BE: https://ses-be.tue.nl/

Seyedzadeh, S., Rahimian, F., Glesk, I., & Roper, M. (2018). Machine learning for estimation of building energy consumption and performance: a review. *Visualisation in Engineering*.

SmartGrids, E. T. (2012). *Strategic Research Agenda (SRA), update of the SmartGrids SRA 2007 for the needs of 2035 STRATEGIC RESEARCH AGENDA.* Leuven: European Technology Platform SmartGrids.

Smola, A., & Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, 199 – 222.

sunrise-and-sunset.com. (2019, 06 29). *Zonsopgang en zonsondergang Amsterdam*. Opgehaald van sunrise-and-sunset : https://www.sunrise-and-sunset.com/nl/sun/nederland/amsterdam/2017/mei

Swan, L., & Ugursal, I. (2009). Modeling of end-use energy consumption in residential sector: A review of modeling techniques. *Renewable and Sustainable Energy Reviews*, 1819–1835.

SwitchExpert. (2019, 07 10). *Terugleververgoeding energie van zonnepanelen*. Opgehaald van energieleveranciers: https://www.energieleveranciers.nl/zonnepanelen/terugleververgoeding-zonnepanelen

Taylor, J., Menezes, L., & McSharry, P. (2016). A comparison of univariate methods for forecasting electricity demand up to a day ahead. *International Journey of Forecasting*, 1 - 16.

Tesla. (2019, 05 23). *Powerwall.* Opgehaald van Maak kennis met Powerwall, uw batterij voor thuis.: https://www.tesla.com/nl_NL/powerwall

Tso, G., & Yau, K. (2005). Predicting electricity energy consumption: A comparison of regression analysis, decision tree and neural networks. *Science Direct: Energy*, 1761–1768.

United-Nations. (2015). *Paris Agreement.* Paris: United Nations.

Wang, Z., Wang, Y., Zeng, R., Srinivasan, R., & Ahrentzen, S. (2018). Random Forest based hourly building energy prediction. *Energy & Buildings*, 11 - 25 .

Wiebes, E. (2019, 3 22). Stimulering Duurzame Energieproductie. *Kamerbrief (SDE+) 2018 December 21*. Den Haag, Noord Holland, Nederland: De Voorzitter van de Tweede Kamer der Staten-Generaal. Opgehaald van Rijksoverheid: https://www.rijksoverheid.nl/ministeries/ministerie-van-economische-zaken-en-klimaat/documenten/kamerstukken/2018/12/21/kamerbrief-over-stimulering-duurzame-energieproductie-sde-2019

Wiebes, E., & Snel, M. (2019, 04 25). Omvorming Salderen. Den Haag, Noord-Holland, Nederland : De Voorzitter van de Tweede Kamer der Staten-Generaal.

Xu, X., Wang, W., Hong, T., & Chen, J. (2019). Incorporating machine learning with building network analysis to predict multi-building energy use . *Energy & Buildings*, 80 - 97.

Zhao, H., & Magoulès, F. (2012). A review on the prediction of building energy consumption. *Renewable and Sustainable Energy Reviews*, 3586 – 3592.

This page was intentionally left blank.

Appendices

Appendix I – Missing values

House nr	*Total missing values consumption*	*Missing values in production data*	*Total missing values grid_consumption*	*Total missing value percentage*	*After filling Grid consumption*	*Left percentage missing values*
4776	3210	758	3968	5.65%	786	1.12%
4794	3181	763	3944	5.62%	791	1.13%
4781	2953	764	3717	5.30%	792	1.13%
4703	3148	793	3941	5.62%	793	1.13%
4783	2854	768	3622	5.16%	796	1.13%
4715	3651	774	4425	6.31%	802	1.14%
4718	4079	782	4861	6.93%	811	1.16%
4700	4407	788	5195	7.40%	816	1.16%
4732	3257	795	4052	5.77%	823	1.17%
4701	3028	798	3826	5.45%	826	1.18%
4788	3060	807	3867	5.51%	835	1.19%
4997	3294	809	4103	5.85%	837	1.19%
4804	3202	811	4013	5.72%	839	1.20%
4793	3174	812	3986	5.68%	840	1.20%
4818	3173	812	3985	5.68%	841	1.20%
4734	3169	817	3986	5.68%	845	1.20%
4733	3154	824	3978	5.67%	852	1.21%
4808	3287	826	4113	5.86%	855	1.22%
4735	3262	831	4093	5.83%	859	1.22%
4791	3276	835	4111	5.86%	863	1.23%
5006	3322	836	4158	5.93%	864	1.23%
4714	3872	845	4717	6.72%	873	1.24%
4800	3436	848	4284	6.10%	876	1.25%
4790	2979	854	3833	5.46%	882	1.26%
4796	3541	863	4404	6.28%	891	1.27%
4827	3721	872	4593	6.54%	900	1.28%
4707	3927	883	4810	6.85%	911	1.30%
7056	4243	885	5128	7.31%	914	1.30%
4756	3997	908	4905	6.99%	936	1.33%
4760	3640	912	4552	6.49%	941	1.34%
4757	3787	935	4722	6.73%	963	1.37%
4738	4197	939	5136	7.32%	967	1.38%
4759	5124	1070	6194	8.83%	1135	1.62%
4754	4852	1117	5969	8.51%	1155	1.65%
4698	5119	1157	6276	8.94%	1228	1.75%
4824	5579	920	6499	9.26%	1235	1.76%
4739	6145	1583	7728	11.01%	1629	2.32%
4745	6189	1701	7890	11.24%	1754	2.50%
4746	7816	1768	9584	13.66%	1844	2.63%
4742	6471	1804	8275	11.79%	1847	2.63%
5014	6342	1870	8212	11.70%	1899	2.71%

5016	6379	1886	8265	11.78%	1915	2.73%
4747	6875	1877	8752	12.47%	1974	2.81%
5015	6712	2000	8712	12.41%	2028	2.89%
5005	7065	885	7950	11.33%	2037	2.90%
4752	6867	1950	8817	12.56%	2055	2.93%
4805	7108	918	8026	11.44%	2090	2.98%
4807	6752	1184	7936	11.31%	2193	3.13%
4780	7870	745	8615	12.28%	2478	3.53%
4820	8283	788	9071	12.93%	2520	3.59%
4706	11804	2516	14320	20.41%	2549	3.63%
5008	8062	808	8870	12.64%	2559	3.65%
5012	8194	846	9040	12.88%	2597	3.70%
4821	8576	906	9482	13.51%	2615	3.73%
4829	9426	929	10355	14.76%	2657	3.79%
4716	7606	1778	9384	13.37%	2659	3.79%
4687	9522	2715	12237	17.44%	2816	4.01%
4689	9546	2737	12283	17.50%	2848	4.06%
4692	9406	2764	12170	17.34%	2874	4.10%
5010	9629	3017	12646	18.02%	3046	4.34%
4741	8380	1879	10259	14.62%	3253	4.64%
4758	11202	3390	14592	20.79%	3415	4.87%
5013	11834	2597	14431	20.56%	3646	5.20%
4696	13487	2798	16285	23.21%	4036	5.75%
4809	9358	2114	11472	16.35%	4063	5.79%
4798	13560	1550	15110	21.53%	4557	6.49%
4778	15809	5086	20895	29.78%	5127	7.31%
4775	10515	841	11356	16.18%	8299	11.83%
5009	3416	8970	12386	17.65%	8999	12.82%
4731	12760	1072	13832	19.71%	9961	14.19%
average	**6145**	**1444**	**7589**	**10.81%**	**2043**	**2.91%**

Appendix II – Cluster sizes

n_cluster = 6

Week 31-32		**Week 22-23**	
Cluster nr	Nr of houses	Cluster nr	Nr of houses
1	24	1	34
3	18	0	24
0	15	4	9
4	11	5	1
5	1	3	1
2	1	2	1
Week 4-5		**Week 10-11**	
Cluster nr	Nr of houses	Cluster nr	Nr of houses
5	24	4	24
1	19	1	16
0	16	3	14
2	9	0	11
4	1	5	4
3	1	2	1

Week 44-45	
Cluster nr	Nr of houses
1	40
4	19
0	8
5	1
3	1
2	1

n_cluster = 5

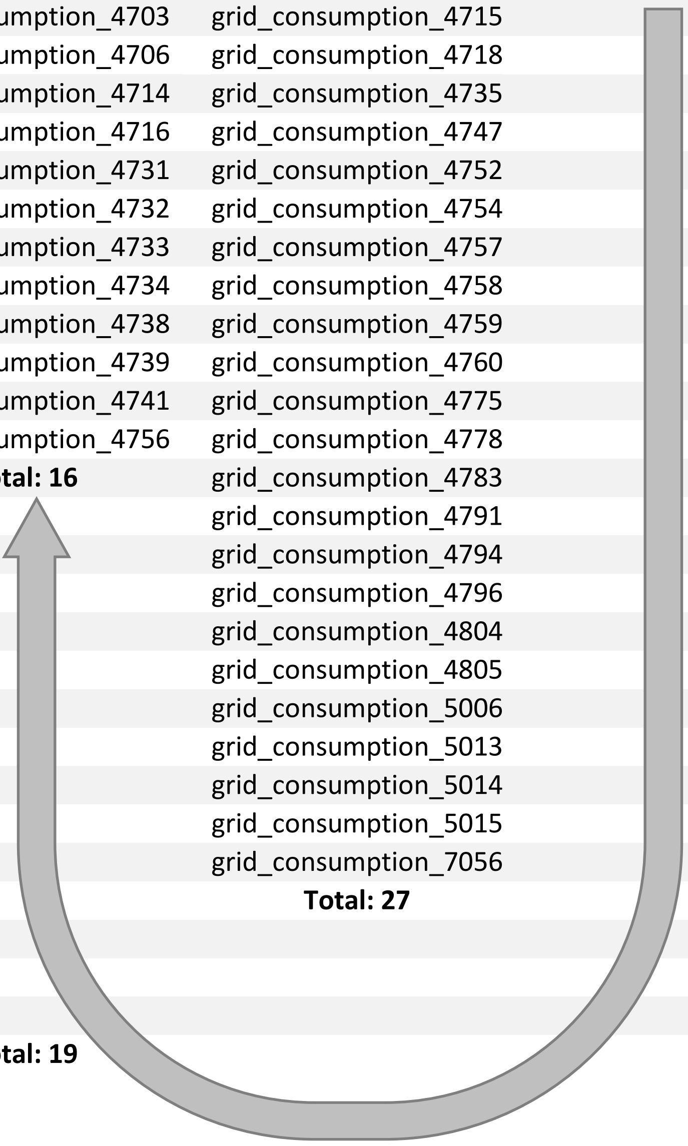
weeks 5-6		**weeks 13-14**	
Cluster	Nr of houses	Cluster	Nr of houses
0	42	1	25
3	21	0	18
2	6	2	15
4	1	4	12
1	1	3	1
week 49-50		**week 15-16**	
Cluster	Nr of houses	Cluster	Nr of houses
0	30	2	25
1	27	1	17
4	11	3	16
3	1	0	11
2	1	4	1
weeks 31-32			
Cluster	Nr of houses		
0	22		
4	17		
2	16		
3	14		
1	2		

n_clusters = 4

Week 7-8		**Week 27-28**	
Cluster nr	Nr of houses	Cluster nr	Nr of houses
2	35	1	25
0	22	3	19
1	7	2	15
3	6	0	11
Week 15-16		**Week 12-13**	
Cluster nr	Nr of houses	Cluster nr	Nr of houses
0	25	1	24
3	18	3	19
1	16	0	16
2	11	2	11
Week 31-32			
Cluster nr	Nr of houses		
0	27		
1	23		
3	17		
2	3		

Appendix III – Cluster distribution September 2017

Cluster 0	Cluster 1	Cluster 2	Cluster 3
grid_consumption_4776	grid_consumption_4687	grid_consumption_4696	grid_consumption_4742
grid_consumption_4780	grid_consumption_4689	grid_consumption_4698	grid_consumption_4745
grid_consumption_4781	grid_consumption_4692	grid_consumption_4701	grid_consumption_4746
grid_consumption_4788	grid_consumption_4700	grid_consumption_4707	**Total: 3**
grid_consumption_4790	grid_consumption_4703	grid_consumption_4715	
grid_consumption_4793	grid_consumption_4706	grid_consumption_4718	
grid_consumption_4798	grid_consumption_4714	grid_consumption_4735	
grid_consumption_4800	grid_consumption_4716	grid_consumption_4747	
grid_consumption_4807	grid_consumption_4731	grid_consumption_4752	
grid_consumption_4808	grid_consumption_4732	grid_consumption_4754	
grid_consumption_4809	grid_consumption_4733	grid_consumption_4757	
grid_consumption_4818	grid_consumption_4734	grid_consumption_4758	
grid_consumption_4820	grid_consumption_4738	grid_consumption_4759	
grid_consumption_4821	grid_consumption_4739	grid_consumption_4760	
grid_consumption_4824	grid_consumption_4741	grid_consumption_4775	
grid_consumption_4827	grid_consumption_4756	grid_consumption_4778	
grid_consumption_4829	**Total: 16**	grid_consumption_4783	
grid_consumption_4997		grid_consumption_4791	
grid_consumption_5005		grid_consumption_4794	
grid_consumption_5008		grid_consumption_4796	
grid_consumption_5009		grid_consumption_4804	
grid_consumption_5010		grid_consumption_4805	
grid_consumption_5012		grid_consumption_5006	
grid_consumption_5016		grid_consumption_5013	
Total: 24		grid_consumption_5014	
		grid_consumption_5015	
		grid_consumption_7056	
		Total: 27	



Total: 19

Appendix IV – Feature importance

Feature importance May 19th 2017

Cluster 0		Cluster 1		Cluster 2		Cluster 3	
feature_names	**feature_importance**	**feature_names**	**feature_importance**	**feature_names**	**feature_importance**	**feature_names**	**feature_importance**
tot_gridconsumed ts-1day	0.5548	tot_gridconsumed ts-1day	0.5698	tot_gridconsumed ts-1day	0.5357	tot_gridconsumed ts-1day	0.4561
Humidity	0.1059	Humidity	0.1069	Humidity	0.1222	tot_gridconsumed ts-3days	0.1701
tot_gridconsumed ts-2days	0.1038	tot_gridconsumed ts-2days	0.0916	Temperature	0.1096	Humidity	0.1308
tot_gridconsumed ts-3days	0.0521	Temperature	0.0396	tot_gridconsumed ts-3days	0.0754	tot_gridconsumed ts-2days	0.0631
Temperature	0.0341	tot_gridconsumed ts-3days	0.0371	tot_gridconsumed ts-2days	0.0423	Temperature	0.0613
tot_gridconsumed ts-4days	0.0298	tot_gridconsumed ts-5days	0.0368	tot_gridconsumed ts-4days	0.0176	tot_gridconsumed ts-4days	0.0221
tot_gridconsumed ts-5days	0.0289	tot_gridconsumed ts-4days	0.0270	tot_gridconsumed ts-7days	0.0160	tot_gridconsumed ts-7days	0.0163
tot_gridconsumed ts-7days	0.0191	tot_gridconsumed ts-7days	0.0192	tot_gridconsumed ts-6days	0.0146	tot_gridconsumed ts-5days	0.0154
tot_gridconsumed ts-6days	0.0170	tot_gridconsumed ts-6days	0.0157	tot_gridconsumed ts-5days	0.0132	tot_gridconsumed ts-6days	0.0144
tot_gridconsumed ts-14days	0.0125	Dew point temp	0.0123	tot_gridconsumed ts-14days	0.0103	Dew point temp	0.0103
Dew point temp	0.0110	tot_gridconsumed ts-14days	0.0120	Dew point temp	0.0101	tot_gridconsumed ts-14days	0.0102
part_of_day_Morning	0.0036	part_of_day_Morning	0.0050	part_of_day_Morning	0.0052	part_of_day_Morning	0.0033
season_winter	0.0032	season_winter	0.0027	part_of_day_Afternoon	0.0034	part_of_day_Afternoon	0.0023
part_of_day_Afternoon	0.0024	part_of_day_Afternoon	0.0021	season_spring	0.0027	season_winter	0.0022
season_spring	0.0016	weekday_Sunday	0.0019	part_of_day_Night	0.0020	season_autumn	0.0019
month_March	0.0016	season_spring	0.0019	season_winter	0.0019	season_spring	0.0018
weekday_Sunday	0.0015	season_autumn	0.0015	season_autumn	0.0018	part_of_day_Night	0.0014

weekday_Wednesday	0.0014	weekday_Saturday	0.0015	weekday_Sunday	0.0011	season_summer	0.0012
weekday_Saturday	0.0014	weekday_Friday	0.0014	weekday_Saturday	0.0011	weekday_Friday	0.0012
weekday_Friday	0.0013	month_March	0.0012	weekday_Tuesday	0.0011	weekday_Tuesday	0.0012
season_autumn	0.0013	weekday_Wednesday	0.0012	part_of_day_Evening	0.0010	month_January	0.0011
weekday_Tuesday	0.0013	weekday_Tuesday	0.0012	weekday_Friday	0.0010	month_June	0.0011
month_June	0.0012	weekday_Thursday	0.0011	season_summer	0.0010	weekday_Sunday	0.0011
weekday_Thursday	0.0010	month_June	0.0011	weekday_Monday	0.0009	weekday_Saturday	0.0010
weekday_Monday	0.0010	month_May	0.0010	month_March	0.0009	weekday_Wednesday	0.0010
part_of_day_Night	0.0008	weekday_Monday	0.0010	month_January	0.0008	weekday_Monday	0.0009
month_May	0.0008	month_April	0.0009	month_July	0.0008	month_March	0.0009
month_July	0.0008	month_July	0.0007	month_June	0.0008	month_April	0.0009
month_April	0.0008	part_of_day_Night	0.0006	month_April	0.0008	month_July	0.0008
season_summer	0.0006	month_September	0.0006	weekday_Wednesday	0.0008	weekday_Thursday	0.0008
part_of_day_Evening	0.0005	season_summer	0.0005	weekday_Thursday	0.0008	month_February	0.0007
month_January	0.0005	month_February	0.0005	month_February	0.0006	part_of_day_Evening	0.0007
month_September	0.0005	part_of_day_Evening	0.0005	month_May	0.0006	month_May	0.0007
month_February	0.0005	month_October	0.0005	month_October	0.0005	month_November	0.0004
month_November	0.0004	month_August	0.0005	month_September	0.0004	month_August	0.0004
month_August	0.0004	month_January	0.0004	month_August	0.0004	month_October	0.0004
month_October	0.0003	month_November	0.0004	month_November	0.0004	month_September	0.0003

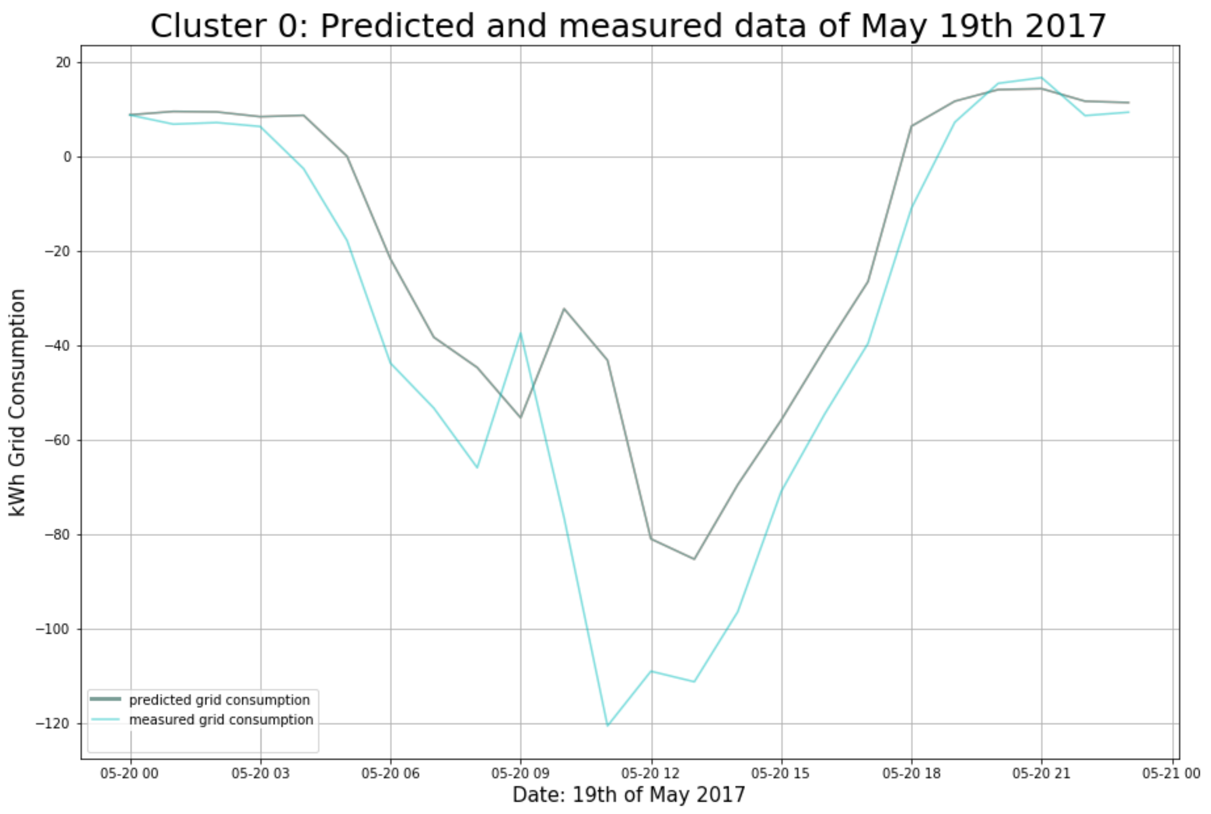
month_December	0.0002	month_December	0.0002	month_December	0.0003	month_December	0.0002

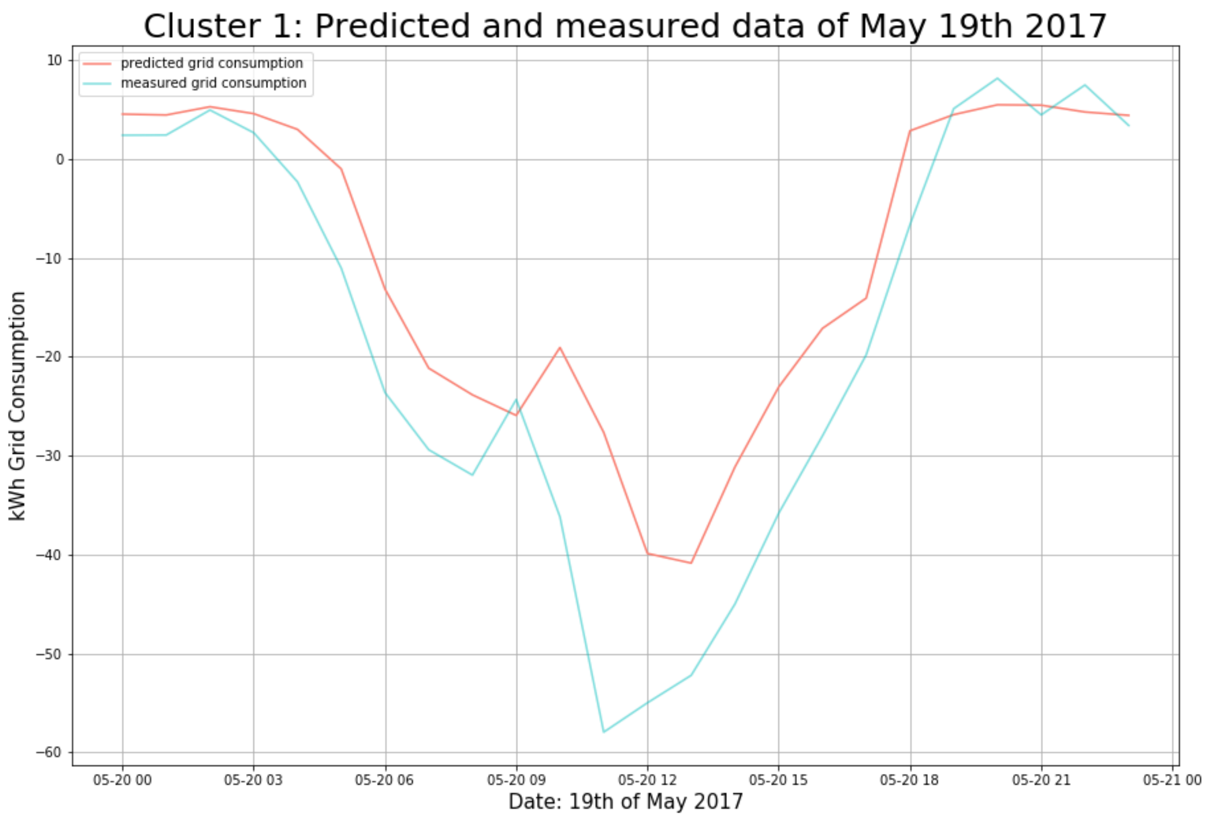
Appendix V – Hyperparameter tuning code and output

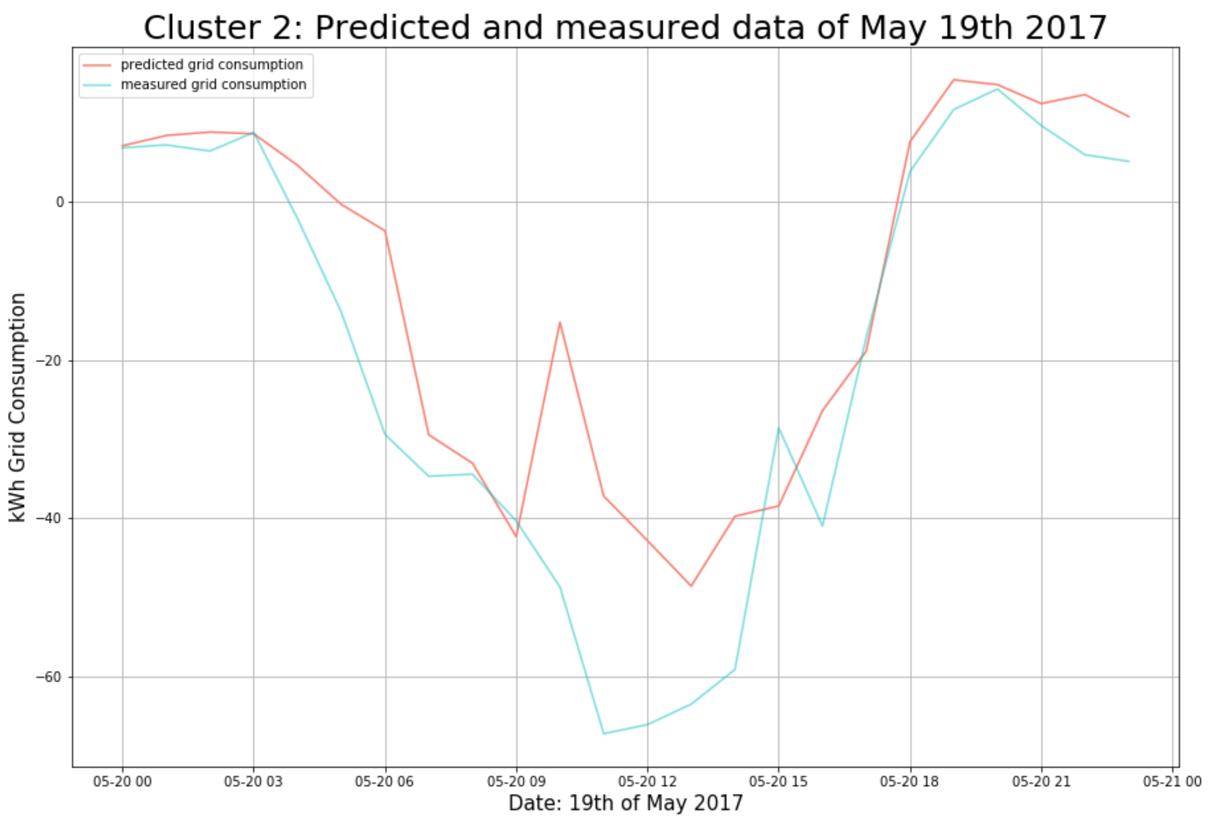
```

'''Hyperparameter tuning'''

from sklearn.model_selection import RandomizedSearchCV


# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 200, stop = 1000, num = 5)]
# Number of features to consider at every split
max_features = ['auto', 'None']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(80, 140, num = 4)]
max_depth.append(None)
# Minimum number of samples required to split a node
min_samples_split = [2, 3, 5, 10]
# Minimum number of samples required at each leaf node
min_samples_leaf = [2, 3, 4]
# Method of selecting samples for training each tree
bootstrap = [True, False]
# Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap}


# Use the random grid to search for best hyperparameters
# First create the base model to tune
rf = RandomForestRegressor()
# Random search of parameters, using 3 fold cross validation,
rf_random = RandomizedSearchCV(estimator = rf, param_distributions = random_grid,
                               n_iter = 100, cv = 3, verbose=2, random_state=42, n_jobs = -1)
# Fit the random search model
rf_random.fit(features, labels)


rf_random.best_params_
```

```
In [44]: grid_search.fit(features, labels)
    ...: grid_search.best_params_
Fitting 3 folds for each of 960 candidates, totalling 2880 fits
[Parallel(n_jobs=-1)]: Done  25 tasks      | elapsed:   21.1s
[Parallel(n_jobs=-1)]: Done 146 tasks      | elapsed:  1.6min
[Parallel(n_jobs=-1)]: Done 349 tasks      | elapsed:  4.0min
[Parallel(n_jobs=-1)]: Done 632 tasks      | elapsed: 24.8min
[Parallel(n_jobs=-1)]: Done 997 tasks      | elapsed: 36.2min
[Parallel(n_jobs=-1)]: Done 1442 tasks      | elapsed: 61.4min
[Parallel(n_jobs=-1)]: Done 1969 tasks      | elapsed: 93.5min
[Parallel(n_jobs=-1)]: Done 2576 tasks      | elapsed: 126.6min
[Parallel(n_jobs=-1)]: Done 2880 out of 2880 | elapsed: 154.8min finished
D:\Programs\Anaconda\lib\site-packages\sklearn\model_selection\_search.py:739:
DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples,), for example using ravel().
  self.best_estimator_.fit(X, y, **fit_params)
Out[44]:
{'bootstrap': True,
 'max_depth': 120,
 'max_features': None,
 'min_samples_leaf': 4,
 'min_samples_split': 3,
 'n_estimators': 350}
```

Appendix VI – Prediction and measured values plots






Cluster 2: Predicted and measured data of May 19th 2017
predicted grid consumption
measured grid consumption
kWh Grid Consumption
0
−20
−40
−60
05-20 00
05-20 03
05-20 06
05-20 09
05-20 12
05-20 15
05-20 18
05-20 21
05-21 00
Date: 19th of May 2017

Appendix VII – Code

Data pre-preparation

```
# -*- coding: utf-8 -*-
"""
Created on Thu Feb 28 16:36:02 2019

@author: vince
"""

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

df = pd.read_csv('15-min_5016.csv', delimiter=',')
knmi = pd.read_excel('KNMI_15min.xlsx')

df.replace('kWh', '', regex=True, inplace=True)
df.replace('GJ', '', regex=True, inplace=True)
df.replace({'Heat pump temp room': {'°C': ''}}, regex=True, inplace=True)
df.replace({'Heat pump sp temp room': {'°C': ''}}, regex=True, inplace=True)
df['ts'].astype(str, inplace=True)
df['ts'] = df['ts'].str.replace('Amsterdam', '')
df['ts'] = df['ts'].str.replace('T', ' ')

df['ts'] = pd.to_datetime(df['ts'])
df['g_consumption'] = df['grid_consumption_5016']
df['g_consumption'].astype(float)
df = df.set_index(df['ts'])
del df['ts']
del df['Unnamed: 0']
del df['grid_consumption_5016']
del df['Heat pump consumed kWh']
del df['Slimme Meter p1 consumed kWh high']
del df['Slimme Meter p1 consumed kWh low']
del df['Slimme Meter p1 produced kWh high']
del df['Slimme Meter p1 produced kWh low']
del df['Solar Inverter kWh produced']
del df['Boiler hottapwater consumed m3']
del df['Heat pump spaceheating delivered']
del df['Boiler supply temp']
del df['Boiler sp temp tapwater']
del df['consumed low ts-1day']
del df['consumed high ts-1day']
del df['consumed low ts-1week']
del df['consumed high ts-1week']
del df['consumed HP ts-1day']
del df['consumed HP ts-1week']
del df['production']
del df['consumption']


knmi= knmi.set_index(df.index)
df['Temperature'] = knmi['Temperature'] *0.1
df['Humidity'] = knmi['Relative humidity'] *0.1
df['Dew point temp'] = knmi['Dew point temp'] *0.1

# remove outliers consuption above and below the 2kw threshold
# fill outliers values
df['g_consumption'].replace(np.nan, 1.123456789, regex=True, inplace=True)
df['g_consumption'][(df['g_consumption']>2.0)] = np.nan
df['g_consumption'] = df['g_consumption'].fillna(method='ffill')
df['g_consumption'].replace(1.123456789, np.nan, regex=True, inplace=True)
```

```
df['g_consumption'].replace(np.nan, 1.123456789, regex=True, inplace=True)
df['g_consumption'][(df['g_consumption']<-2.0)] = np.nan
df['g_consumption'] = df['g_consumption'].fillna(method='ffill')
df['g_consumption'].replace(1.123456789, np.nan, regex=True, inplace=True)

#create the delayed data
df['grid_consumed ts-1day'] = df['g_consumption'].shift(96)
df['grid_consumed ts-2days'] = df['g_consumption'].shift(192)
df['grid_consumed ts-3days'] = df['g_consumption'].shift(288)
df['grid_consumed ts-4days'] = df['g_consumption'].shift(384)
df['grid_consumed ts-5days'] = df['g_consumption'].shift(480)
df['grid_consumed ts-6days'] = df['g_consumption'].shift(576)
df['grid_consumed ts-7days'] = df['g_consumption'].shift(672)
df['grid_consumed ts-14days'] = df['g_consumption'].shift(1344)


#Adding weekday month and season columns
month_num_to_season =   { 1:'winter',  2:'winter',
                          3:'spring',  4:'spring',  5:'spring',
                          6:'summer',  7:'summer',  8:'summer',
                          9:'autumn', 10:'autumn', 11:'autumn',
                         12:'winter'}
grouped =  df.groupby(lambda x: month_num_to_season.get(x.month))

month =     { 1:'January', 2:'February', 3:'March', 4:'April',
             5:'May', 6:'June', 7:'July', 8:'August', 9:'September',
             10:'October', 11:'November', 12:'December'}

Weekday = {1:'Sunday', 2:'Monday', 3:'Tuesday', 4:'Wednesday',
           5:'Thursday', 6:'Friday', 7:'Saturday'}

df['weekday'] = df.index.weekday_name
df['month'] = [month.get(t_stamp.month) for t_stamp in df.index]
df['season'] = [month_num_to_season.get(t_stamp.month) for t_stamp in df.index]

part_of_day = [0, 6, 12, 18, 24]
labels = ['Morning','Afternoon','Evening','Night']
hours = df.index.hour
df['part_of_day'] = pd.cut(hours-6+24 *(hours<6),bins=part_of_day,
                                                 labels=labels,right=False)

categorical = df[['weekday', 'month', 'season', 'part_of_day']]
numerical = df[['Heat pump temp room', 'Heat pump sp temp room',
                'Temperature', 'Humidity', 'Dew point temp', 'grid_consumed ts-1day',
                'grid_consumed ts-2days', 'grid_consumed ts-3days',
                'grid_consumed ts-4days', 'grid_consumed ts-5days',
                'grid_consumed ts-6days','grid_consumed ts-7days',
                'grid_consumed ts-14days', 'g_consumption']]

categorical = pd.get_dummies(categorical)

data_pred = pd.concat([numerical, categorical], axis=1)
data_pred = data_pred.set_index(df.index)
data_pred.isna().sum()
'''create the hourly dataset'''
#create set for hourly data consumption, these have to be summed
hour_data_cons = data_pred[['grid_consumed ts-1day',
                'grid_consumed ts-2days', 'grid_consumed ts-3days',
                'grid_consumed ts-4days', 'grid_consumed ts-5days',
                'grid_consumed ts-6days','grid_consumed ts-7days',
                'grid_consumed ts-14days', 'g_consumption']]
```

```
hour_data_cons =  hour_data_cons.resample('60min').sum()

hour_data_average = data_pred[['Heat pump temp room', 'Heat pump sp temp room',
                          'Temperature', 'Humidity', 'Dew point temp',
                          'weekday_Friday', 'weekday_Saturday', 'weekday_Sunday',
                          'weekday_Monday', 'weekday_Tuesday', 'weekday_Wednesday',
                          'weekday_Thursday',
                          'month_January', 'month_February', 'month_March',
                          'month_April', 'month_May', 'month_June', 'month_July',
                          'month_August', 'month_September', 'month_October',
                          'month_November', 'month_December',
                          'season_summer', 'season_winter', 'season_spring',
                          'season_autumn',
                          'part_of_day_Morning', 'part_of_day_Afternoon',
                          'part_of_day_Evening', 'part_of_day_Night']]

hour_data_average =  hour_data_average.resample('60min').mean()

hourly_data = pd.concat([hour_data_cons, hour_data_average], axis=1)
hourly_data.to_excel('pred_data-5016.xlsx')
```

Missing values and clustering set

```
# -*- coding: utf-8 -*-
"""
Created on Thu Mar 28 15:38:41 2019

@author: vince
"""

import pandas as pd
import numpy as np

df = pd.read_csv('data_BEN-7056.csv', delimiter=';')

df['ts'] = df['ts'].str.replace('Amsterdam', '')
df['ts'] = df['ts'].str.replace('T', ' ')
df['ts'] = pd.to_datetime(df['ts'])

df.replace('kWh', '', regex=True, inplace=True)
df['Slimme Meter p1 consumed kWh low']
                    = df['Slimme Meter p1 consumed kWh low'].astype(float)
df['Slimme Meter p1 consumed kWh high']
                    = df['Slimme Meter p1 consumed kWh high'].astype(float)
df['Heat pump consumed kWh'] = df['Heat pump consumed kWh'].astype(float)
df['production'] = df['Solar Inverter kWh produced'].astype(float)

#replacing the missing value with past day's same time value

df['consumed low ts-1day'] = df['Slimme Meter p1 consumed kWh low'].shift(96)
df['consumed high ts-1day'] = df['Slimme Meter p1 consumed kWh high'].shift(96)
df['consumed HP ts-1day'] = df['Heat pump consumed kWh'].shift(96)

df['Slimme Meter p1 consumed kWh low'] = df['Slimme Meter p1 consumed kWh low']
                                        .fillna(df['consumed low ts-1day'])
df['Slimme Meter p1 consumed kWh high'] = df['Slimme Meter p1 consumed kWh high']
                                    .fillna(df['consumed high ts-1day'])
df['Heat pump consumed kWh'] = df['Heat pump consumed kWh'].fillna(df['consumed HP ts-1day'])
```

```
#replacing the missing value with past week's same time value
df['consumed low ts-1week'] = df['Slimme Meter p1 consumed kWh low'].shift(672)
df['consumed high ts-1week'] = df['Slimme Meter p1 consumed kWh high'].shift(672)
df['consumed HP ts-1week'] = df['Heat pump consumed kWh'].shift(672)

df['Slimme Meter p1 consumed kWh low'] = df['Slimme Meter p1 consumed kWh low'].fillna(df['consu
df['Slimme Meter p1 consumed kWh high'] = df['Slimme Meter p1 consumed kWh high'].fillna(df['co
df['Heat pump consumed kWh'] = df['Heat pump consumed kWh'].fillna(df['consumed HP ts-1week'])

df['Slimme Meter p1 consumed kWh low'].isna().sum()
df['Slimme Meter p1 consumed kWh high'].isna().sum()
df['Heat pump consumed kWh'].isna().sum()

df['consumption'] = df['Slimme Meter p1 consumed kWh low'] + df['Slimme Meter p1 consumed kWh h
df['consumption'].isna().sum()

#forward fill any still missing values, limit =1
df['production'] = df['production'].fillna(method='ffill', limit=1)

df['grid_consumption_7056'] = df['consumption'] - df['production']

'''
#one time creation of the cluster table!
df_cluster = df['grid_consumption_7056']
df_cluster = pd.DataFrame(df_cluster)

df_cluster = df_cluster.set_index(df.ts)
'''

df_cluster = df_cluster.T
#cluster_data = cluster_data.iloc[1:]
df_cluster.columns = df['ts']

cluster_data = cluster_data.append(df_cluster)

cluster_data.to_csv('cluster_data_def.csv')
```

Clustering

Creating accumulated cluster sets example

```
  # -*- coding: utf-8 -*-
  """
  Created on Thu Mar 21 15:09:03 2019

  @author: vince
  """

  import pandas as pd

  #cluster 1 week 7-8 2017
  df = pd.read_excel('pred_data-4696.xlsx')
  df2 = pd.read_excel('pred_data-4742.xlsx')
  df3 = pd.read_excel('pred_data-4745.xlsx')
  df4 = pd.read_excel('pred_data-4746.xlsx')
  df5 = pd.read_excel('pred_data-4805.xlsx')
  df6 = pd.read_excel('pred_data-5005.xlsx')
  df7 = pd.read_excel('pred_data-5013.xlsx')
  df8 = pd.read_excel('pred_data-4687.xlsx')
  df9 = pd.read_excel('pred_data-4689.xlsx')
  df10 = pd.read_excel('pred_data-4692.xlsx')
  df11 = pd.read_excel('pred_data-4700.xlsx')
  df12 = pd.read_excel('pred_data-4703.xlsx')
  df13 = pd.read_excel('pred_data-4706.xlsx')
  df14 = pd.read_excel('pred_data-4714.xlsx')
  df15 = pd.read_excel('pred_data-4716.xlsx')
  df16 = pd.read_excel('pred_data-4731.xlsx')
  df17 = pd.read_excel('pred_data-4732.xlsx')
  df18 = pd.read_excel('pred_data-4733.xlsx')
  df19 = pd.read_excel('pred_data-4734.xlsx')
  df20 = pd.read_excel('pred_data-4738.xlsx')
  df21 = pd.read_excel('pred_data-4739.xlsx')
  df22 = pd.read_excel('pred_data-4741.xlsx')
  df23 = pd.read_excel('pred_data-4756.xlsx')

df.isna().sum()
df_accumulated = df
df_accumulated['tot_grid_consumption'] = (df['g_consumption'] + df2['g_consumption'] + df3['g_consumption']
                                    + df4['g_consumption'] + df5['g_consumption'] + df6['g_consumption']
                                    + df7['g_consumption'] + df8['g_consumption'] + df9['g_consumption']
                                    + df10['g_consumption'] + df11['g_consumption']+ df12['g_consumption']
                                    + df13['g_consumption'] + df14['g_consumption'] + df15['g_consumption']
                                    + df16['g_consumption'] + df17['g_consumption'] + df18['g_consumption']
                                    + df19['g_consumption'] + df20['g_consumption']
                                    + df21['g_consumption'] + df22['g_consumption']
                                    + df23['g_consumption']  ).astype(float)


df_accumulated['tot_gridconsumed ts-1day'] = df_accumulated['tot_grid_consumption'].shift(96)
df_accumulated['tot_gridconsumed ts-2days'] = df_accumulated['tot_grid_consumption'].shift(192)
df_accumulated['tot_gridconsumed ts-3days'] = df_accumulated['tot_grid_consumption'].shift(288)
df_accumulated['tot_gridconsumed ts-4days'] = df_accumulated['tot_grid_consumption'].shift(384)
df_accumulated['tot_gridconsumed ts-5days'] = df_accumulated['tot_grid_consumption'].shift(480)
df_accumulated['tot_gridconsumed ts-6days'] = df_accumulated['tot_grid_consumption'].shift(576)
df_accumulated['tot_gridconsumed ts-7days'] = df_accumulated['tot_grid_consumption'].shift(672)
df_accumulated['tot_gridconsumed ts-14days'] = df_accumulated['tot_grid_consumption'].shift(1344)

df_accumulated = df_accumulated.set_index(df_accumulated['ts'])
del df_accumulated['grid_consumed ts-1day']
del df_accumulated['grid_consumed ts-2days']
del df_accumulated['grid_consumed ts-3days']
del df_accumulated['grid_consumed ts-4days']
del df_accumulated['grid_consumed ts-5days']
del df_accumulated['grid_consumed ts-6days']
del df_accumulated['grid_consumed ts-7days']
del df_accumulated['grid_consumed ts-14days']
del df_accumulated['g_consumption']

del df_accumulated['ts']

df_accumulated.to_excel('cluster1_week7-8_17.xlsx')
```

K-means clustering example

```
# -*- coding: utf-8 -*-
"""
Created on Fri Apr  5 11:40:12 2019

@author: vince
"""

import pandas as pd

from sklearn import cluster

cluster_data = pd.read_csv('cluster_data_complete.csv', delimiter=',')
cluster_data = cluster_data.set_index(cluster_data['Unnamed: 0'])
del cluster_data['Unnamed: 0']

clustering = cluster.KMeans(n_clusters=4, n_init=100, max_iter=1000)

cluster_data.fillna(0, inplace=True)
cluster_data['cluster'] = clustering.fit_predict(cluster_data[cluster_data.columns[52900:54244]

weeks = cluster_data.columns[52900:54244]
weeks = pd.DataFrame(weeks)
T_cluster = cluster_data.T

Results_table = cluster_data['cluster']
Results_table = pd.DataFrame(Results_table)
Results_table = Results_table.set_index(cluster_data.index)
Results_table['cluster'].value_counts()
Results_table.to_excel('cluster_distribution-June-19.xlsx')
```

Prediction

Hyperparameter tuning

```
'''Hyperparameter tuning'''

from sklearn.model_selection import RandomizedSearchCV


# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 200, stop = 1000, num = 5)]
# Number of features to consider at every split
max_features = ['auto', 'None']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(80, 140, num = 4)]
max_depth.append(None)
# Minimum number of samples required to split a node
min_samples_split = [2, 3, 5, 10]
# Minimum number of samples required at each leaf node
min_samples_leaf = [2, 3, 4]
# Method of selecting samples for training each tree
bootstrap = [True, False]
# Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap}


# Use the random grid to search for best hyperparameters
# First create the base model to tune
rf = RandomForestRegressor()
# Random search of parameters, using 3 fold cross validation,
rf_random = RandomizedSearchCV(estimator = rf, param_distributions
                               = random_grid, n_iter = 100, cv = 3, verbose=2,
                                   random_state=42, n_jobs = -1)
```

```
# Use the random grid to search for best hyperparameters
# First create the base model to tune
rf = RandomForestRegressor()
# Random search of parameters, using 3 fold cross validation,
rf_random = RandomizedSearchCV(estimator = rf, param_distributions
                               = random_grid, n_iter = 100, cv = 3, verbose=2,
                                   random_state=42, n_jobs = -1)
# Fit the random search model
rf_random.fit(features, labels)

rf_random.best_params_
```

Prediction example

```
# -*- coding: utf-8 -*-
"""
Created on Wed Mar 20 07:57:41 2019

@author: vince
"""

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sklearn as sk
from sklearn.tree import DecisionTreeClassifier
from sklearn import preprocessing
from sklearn.tree import export_graphviz


data = pd.read_excel("cluster2-3_proof-of-concept-June19.xlsx")

train_data = data.iloc[0:13561]
val_data = data.iloc[13561:13585]

train_labels = train_data['tot_grid_consumption']
train_labels = pd.DataFrame(train_labels)
train_labels = train_labels.set_index(train_data['ts'])
train_features = train_data
train_features = train_features.set_index(train_data['ts'])

#del train_features['ts.1']
del train_features['ts']
del train_features['tot_grid_consumption']

train_features[['Temperature', 'Humidity' , 'Dew point temp', 'tot_gridconsumed ts-1day',
                'tot_gridconsumed ts-2days', 'tot_gridconsumed ts-3days',
                'tot_gridconsumed ts-4days', 'tot_gridconsumed ts-5days',
                'tot_gridconsumed ts-6days', 'tot_gridconsumed ts-7days',
                'tot_gridconsumed ts-14days']] = train_features[['Temperature',
                     'Humidity' , 'Dew point temp', 'tot_gridconsumed ts-1day',
                     'tot_gridconsumed ts-2days', 'tot_gridconsumed ts-6days',
                     'tot_gridconsumed ts-7days', 'tot_gridconsumed ts-14days'
                     ]].astype(float)

train_features[['weekday_Friday', 'weekday_Saturday', 'weekday_Sunday',
                             'weekday_Monday', 'weekday_Tuesday', 'weekday_Wednesday',
                             'weekday_Thursday',
                             'month_January', 'month_February', 'month_March',
                             'month_April', 'month_May', 'month_June', 'month_July',
                             'month_August', 'month_September', 'month_October',
                             'month_November', 'month_December',
                             'season_summer', 'season_winter', 'season_spring',
                             'season_autumn',
                             'part_of_day_Morning', 'part_of_day_Afternoon',
                             'part_of_day_Evening', 'part_of_day_Night']] =
                               train_features[['weekday_Friday', 'weekday_Saturday',
                             'weekday_Sunday', 'weekday_Monday', 'weekday_Tuesday',
                             'weekday_Wednesday', 'weekday_Thursday', 'month_January',
                             'month_February', 'month_March', 'month_April', 'month_May',
                             'month_June', 'month_July', 'month_August', 'month_September',
                             'month_October', 'month_November', 'month_December',
                             'season_summer', 'season_winter', 'season_spring','season_autumn',
                             'part_of_day_Morning', 'part_of_day_Afternoon',
                             'part_of_day_Evening', 'part_of_day_Night']].astype('category')
```

```
val_labels = val_data['tot_grid_consumption']
val_labels = pd.DataFrame(val_labels)
val_labels = val_labels.set_index(val_data['ts'])
val_features = val_data
val_features = val_features.set_index(val_data['ts'])

del val_features['ts']
del val_features['tot_grid_consumption']

val_features[['Temperature', 'Humidity', 'Dew point temp', 'tot_gridconsumed ts-1day',
              'tot_gridconsumed ts-2days', 'tot_gridconsumed ts-3days',
              'tot_gridconsumed ts-4days', 'tot_gridconsumed ts-5days',
              'tot_gridconsumed ts-6days', 'tot_gridconsumed ts-7days',
              'tot_gridconsumed ts-14days']] = val_features[['Temperature',
              'Humidity' , 'Dew point temp', 'tot_gridconsumed ts-1day',
              'tot_gridconsumed ts-2days', 'tot_gridconsumed ts-3days',
              'tot_gridconsumed ts-4days', 'tot_gridconsumed ts-5days',
              'tot_gridconsumed ts-6days', 'tot_gridconsumed ts-7days',
              'tot_gridconsumed ts-14days']].astype(float)


val_features[['Temperature', 'Humidity', 'Dew point temp', 'tot_gridconsumed ts-1day',
              'tot_gridconsumed ts-2days', 'tot_gridconsumed ts-3days',
              'tot_gridconsumed ts-4days', 'tot_gridconsumed ts-5days',
              'tot_gridconsumed ts-6days', 'tot_gridconsumed ts-7days',
              'tot_gridconsumed ts-14days']] = val_features[['Temperature',
              'Humidity' , 'Dew point temp', 'tot_gridconsumed ts-1day',
              'tot_gridconsumed ts-2days', 'tot_gridconsumed ts-3days',
              'tot_gridconsumed ts-4days', 'tot_gridconsumed ts-5days',
              'tot_gridconsumed ts-6days', 'tot_gridconsumed ts-7days',
              'tot_gridconsumed ts-14days']].astype(float)

val_features[['weekday_Friday', 'weekday_Saturday', 'weekday_Sunday',
                          'weekday_Monday', 'weekday_Tuesday', 'weekday_Wednesday',
                          'weekday_Thursday',
                          'month_January', 'month_February', 'month_March',
                          'month_April', 'month_May', 'month_June', 'month_July',
                          'month_August', 'month_September', 'month_October',
                          'month_November', 'month_December',
                          'season_summer', 'season_winter', 'season_spring',
                          'season_autumn',
                          'part_of_day_Morning', 'part_of_day_Afternoon',
                          'part_of_day_Evening', 'part_of_day_Night']]
            = val_features[['weekday_Friday', 'weekday_Saturday', 'weekday_Sunday',
                          'weekday_Monday', 'weekday_Tuesday', 'weekday_Wednesday',
                          'weekday_Thursday',
                          'month_January', 'month_February', 'month_March',
                          'month_April', 'month_May', 'month_June', 'month_July',
                          'month_August', 'month_September', 'month_October',
                          'month_November', 'month_December',
                          'season_summer', 'season_winter', 'season_spring',
                          'season_autumn',
                          'part_of_day_Morning', 'part_of_day_Afternoon',
                          'part_of_day_Evening', 'part_of_day_Night'
                          ]].astype('category')
```

```
  from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor


  random_forest = RandomForestRegressor(
                 n_estimators=100, criterion='mse', max_depth=None,
                 min_samples_split=2, min_samples_leaf=1,
                 min_weight_fraction_leaf=0.0, max_features='auto',
                 max_leaf_nodes=None, min_impurity_decrease=0.0,
                 min_impurity_split=None, bootstrap=True, oob_score=False,
                 n_jobs=-1, random_state=None, verbose=0, warm_start=False)

  random_forest.fit(train_features, train_labels)
  model_pred = random_forest.predict(val_features)

  Random_tree_R2 = sk.metrics.r2_score(val_labels, model_pred)


MSE = sk.metrics.mean_squared_error(val_labels, model_pred)
MAE = sk.metrics.mean_absolute_error(val_labels, model_pred)
RMSE = np.sqrt(MSE)

np.mean(abs(model_pred))
np.mean(abs(val_labels))

print('MSE:', MSE, 'MAE', MAE, 'RMSE:', RMSE, 'abs_mean:',
                                      np.mean(abs(val_labels)))

comparison = pd.DataFrame(val_labels)
comparison['model_pred'] = model_pred
comparison.to_excel('comparison_June19_cluster2-3.xlsx')

feature_importance = random_forest.feature_importances_
feature_names = train_features.columns
feature_importance_table = pd.DataFrame({'feature_names':feature_names,
                                         'feature_importance':feature_importance})
number_features_used = random_forest.n_features_

feature_importance_table.to_excel('Feature_importance-cluster2-3.xlsx')

plt.figure(figsize=(15,10))
plt.plot(val_data['ts'], model_pred, color='salmon', alpha=1.0,
         label='predicted grid consumption')
plt.plot(val_data['ts'], val_labels, color='c', alpha=0.5,
         label='measured grid consumption')
plt.title('Cluster 3: Predicted and measured data of June 19th 2017', size=25)
plt.grid()
plt.legend()
plt.xlabel('Date: 19th of June 2017', size=15)
plt.ylabel('kWh Grid Consumption', size=15)


labels['tot_grid_consumption'].corr(features['tot_gridconsumed ts-1day'])
labels['tot_grid_consumption'].corr(features['tot_gridconsumed ts-14days'])

val_mean_abs = abs(val_labels)
abs_model_pred = abs(model_pred)
MSE_abs = sk.metrics.mean_squared_error(val_mean_abs, abs_model_pred)
RMSE_abs = np.sqrt(MSE_abs)
np.mean(val_mean_abs)


errors = abs(model_pred - val_labels)
mean_error = np.mean(errors)
mean_val_labels = np.mean(abs(val_labels))
mape = 100 * (mean_error/mean_val_labels)
accuracy = 100 - mape
print('Model Performance')
print('Average Error: {:0.4f} degrees.'.format(np.mean(errors)))
print('Accuracy = {:0.2f}%.'.format(accuracy))
```