Master's thesis

2020

Kari Walstad

Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electric Power Engineering

NTNU
Norwegian University of
Science and Technology

Kari Walstad

Electric Utility Customer Segmentation from Advanced Metering Systems Data

Development of a computer programme for shape-based time series segmentation

June 2020

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

Electric Utility Customer Segmentation from Advanced Metering Systems Data

Development of a computer programme for shape-based time series segmentation

Kari Walstad

Energy and Environmental Engineering
Submission date: June 2020
Supervisor: Vijay Venu Vadlamudi

Norwegian University of Science and Technology
Department of Electric Power Engineering

Abstract

This is a master's thesis written for the Department of Electric Power Engineering at the Norwegian University of Science and Technology in collaboration with Lyse Elnett and the Centre for Intelligent Electricity Distribution. In this master's thesis, a computer based tool is developed for the segmentation of the customer base of Distribution System Operators (DSO) based of Advanced Metering System (AMS) time series data of the DSO customers.

The Norwegian Regulatory Authority for Energy have required all Norwegian DSOs to install AMS-meters at every point of measurement in the distribution utility grid by 1 January 2019. An AMS-meter is able to capture a wide variety of data types in real time. Analysis and evaluation of this data has the potential to benefit DSOs in several ways. However, as was confirmed through contacting several DSOs, such as Elvia, BKK Nett and Lyse Elnett, up to now DSOs have only to a limited extend exploited the potential benefits of using AMS data. In the report, four potential areas of research within the use of AMS-data for DSOs were identified. Through an Analytical Hierarchy Process one of these four areas was selected for further study. The area of use for AMS-data which was selected for this report, was the segmentation of utility grid customers based on AMS time series data of customer electricity consumption.

A synthesis computer programme for customer segmentation was developed in the coding language Python, using shape-based clustering (i.e. amplitude, offset and time invariant), and a Cluster Validation Index (CVI) algorithm. Additionally, an option to perform outlier analysis of the AMS input data was included in the programme.

Following the development of a conceptual framework for the computer programme, the actual algorithms used in the customer segmentation model were selected based on the merits of each specific algorithm. The programme was also written in such a way that the different algorithms could be changed/replaced by the user. The selected clustering algorithm was the K-Shape algorithm and the selected CVI was the Silhouette algorithm. Dynamic Time Warping was chosen as the algorithm used in the different outlier analysis methods.

The assessment of the developed customer segmentation programme and the underlying methodology was first done through tests on a known data set to check whether the results that were known beforehand to be correct would be produced. The developed segmentation programme was shown to produce good results when tested on a known data set. Further, the included outlier analysis methods were shown to be able to sort out predefined time series outliers provided user evaluation and input. The outlier analysis part is developed to a certain point and has potential be further improved.

Following this, an assessment was made on the basis of actual AMS-data sets provided by Lyse Elnett. When tested on the first AMS-data set, the programme showed a tendency to prefer segmenting based on similar periodicity of the time series. Additionally, AMS-data was more challenging for the algorithm to cluster than the known data set, possibly because the AMS-data set was more homogeneous with more similarly shaped and less discernible time series groups. Outlier analysis was shown to improve the programme performance by removing irregular (i.e. flat) time series.

Based on the second AMS-data set, a comparison with the current standard method of customer segmentation utilized by Norwegian DSOs was also performed. The developed customer segmentation method was shown to produce a better partition compactness of the AMS-data set than the standard DSO method when measured with a CVI. It should be kept in mind that the main objective of the standard DSO segmentation method may not necessarily be on segmenting for similar consumption shapes, as opposed to the developed segmentation programme.

Some potential improvements to the developed programme were also identified. One of these was the potential for the programme to take into account the standard DSO segmentation, in order to include amplitude and offset information in the analysis. Another area of improvement was the language in which the programme was written. Python is a relatively high-level coding language, and the efficiency of the programme and thereby calculation time may be improved if a lower-level language such as C++ were utilized.

Sammendrag

Denne masteroppgaven er skrevet for Institutt for elkraftteknikk ved Norges teknisk-naturvitenskapelige universitet (NTNU), Trondheim, i samarbeid med Lyse Elnett AS og Centre for Intelligent Electricity Distribution (CINELDI). Tema for masteroppgaven er utvikling av et datamaskinbasert verktøy for segmentering av kunder til nettselskaper basert på tidsserier fra AMS-måledata (Advanced Metering Systems) av kundenes strømforbruk.

Reguleringsmyndigheten for Energi (RME) i Norge har stilt som krav at alle norske nettselskaper skal ha installert "smarte" strømmålere, det vil si AMS-målere, innen 1, januar 2019 ved alle aktuelle målepunkt i sitt distribusjonsnettverk. AMS-målere kan registrere et spekter av parametere i sanntid. Tilgang på slike data kan ha mange fordeler for nettselskapene. Likevel har undersøkelser gjennomført i forbindelse med denne oppgaven avdekket av nettselskapene i liten grad anvender AMS-data. I rapporten er det identifisert fire anvendelsesområder for AMS-data som ville vært gunstige for nettselskapene å ta i bruk. Gjennom en "Analytical Hierarchy Process" ble ett av områdene valgt ut for videre studie i denne rapporten. Området som ble valgt er segmentering av nettselskapenes kunder basert på tidsserier av AMS-målinger av strømforbruk.

Et dataprogram satt sammen av ulike komponenter for segmentering av nettselskapenes kunder ble utviklet. Programmet er laget i programmeringsspråket Python og segmenterer kunder basert på tidsserienes form (shape-based clustering), uavhengig av amplitude, offset og tid (amplitude, offset, and time invariant), samt en Cluster Validation Index (CVI) algoritme. I tillegg ble det lagt inn i programmet metoder for å skille ut spesielle avvikende AMS-tidsserier basert på styring fra bruker.

Etter at den konseptmessige rammen for dataprogrammet var etablert, ble algoritmene benyttet i dataprogrammet valgt basert på fordeler og ulemper for hver enkelt. Programmet er skrevet på en slik måte at algoritmene lett kan byttes ut og erstattes med andre av brukeren. Den valgte segmenteringsalgoritmen heter "K-Shape" og den valgte metoden for å måle godhet av segmenteringen, CVI, heter "Silhouette algorithm". "Dynamic Time Warping" er valgt som algoritme for de ulike metodene for analyse av avvikende tidsserier.

Evalueringen av det utviklede dataprogrammet ble først gjennomført basert på kjente datasett for å kontrollere at modellen var i stand til å produsere et resultat som på forhånd var kjent. Dataprogrammet gjennomførte dette på en god måte. Deretter ble det gjort en evaluering av programmet basert på AMS-tidsserier gjort tilgjengelig av Lyse Elnett. I rapporten er det vist at programdelen for analyse av avvikende tidsserier er i stand til å sortere ut forhåndsdefinerte avvikende tidsserier. Programdelen er utviklet til et visst punkt, men kan utvikles videre.

Deretter ble programmet testet basert på to AMS-datasett fra Lyse Elnett. Testingen med det første datasettet viste at det utviklede programmet har en tendens til å segmentere alle tidsserier med lignende periodisitet i samme segmentgruppe og derved prefere et lite antall segmenter. AMS-dataene viste seg å være mer utfordrende for programmet å segmentere enn settet med kjente data, etter som AMS-

dataene inneholdt grupper med tidsserier som var mindre distinkte og tidsserier som kunne anses som irregulære. Programdelen for å sortere ut avvikende tidsserier ble anvendt med positivt resultat gjennom å separere ut irregulære tidsserier (flate tidsserier) fra datasettet.

Testing av programmet ble også gjort på det andre datasettet fra Lyse Elnett i form av en sammenligning av segmenteringsresultatene fra kjøringer av dataprogrammet opp mot standardsegmenteringen som nettselskapene per i dag benytter. Det utviklede dataprogrammet produserte en bedre segmentering enn standardsegmenteringen målt med CVI. Det er likevel viktig å her være klar over at nettselskapenenes standardmetode for segmentering ikke nødvendigvis er basert på at kunder med likt profil for sitt strømkonsum skal grupperes i samme kategori, hvilket er basis for det dataprogrammet utviklet for denne rapporten.

Det ble også identifisert noen forbedringspunkter for det utviklede dataprogrammet. Et av disse er å gjøre det mulig for programmet å ta den eksisterende standardsegmenteringen som nettselskapene benytter med i betraktningen, for også å kunne ta hensyn til amplitude og offset av tidsseriene i segmenteringens. Et annet forbedringspunkt er knyttet til programmeringsspråket som er benyttet for programmet. Python er et relativt høynivå-programmeringsspråk og ved å benytte et mer lavnivå-språk, som for eksempel C++, kan effektiviteten til programmet økes og derved tiden for gjennomføringen av analysene reduseres betraktelig.

Contents

Abbreviations

AHP	Analytic Hierarchy Process
AMS	Advanced Metering Systems
API	Application Programming Interface
CSV	Comma-Separated Values
CVI	Cluster Validation Index
DMS	Distribution Management System
DSO	Distribution System Operator
DTW	Dynamic Time Warping
EEA	European Economic Area
EMS	Energy Management System
EU	European Union
HAN	Home Area Network
ICT	Information and Communications Technology
ID	Identifier
IT	Information Technology
NCC	Normalized Cross-Correlation
NIS	Network Information System
NVE	Norges Vassdrags- og Energidirektorat
OD1	Outlier Detection method 1

OD2	Outlier Detection method 2
OD3	Outlier Detection method 3
OED	Olje- og Energidepartementet
OMS	Outage Mnagement System
RME	Reguleringsmyndigheten for Energi
SBD	Shape-Based Distance
SCADA	Supervisory Control And Data Acquisition
SE	State Estimation
STL	Seasonal-Trend decomposition using LOESS
TSO	Transmission System Operator

Definitions

Algorithm

Algorithm is defined by the University of Oxford [10] as:

> "A process or set of rules to be followed in calculations or other problem-solving operations, especially by a computer."

Application Programming Interface (API)

Application Programming Interface is defined by the global research and advisory firm, Gartner Inc. [13] as:

> "An application programming interface (API) is an interface that provides programmatic access to service functionality and data within an application or a database."

Cache

Cache is defined by Gartner Inc. [19] as:

> "A cache is defined as a temporary storage area for instructions and data near a computer's central processing unit (CPU), usually implemented in high-speed memory. It replicates information from main memory or storage in a way that facilitates quicker access, using fewer resources than the original source."

Demand response

Demand response is defined by Gartner Inc. [23] as:

> "Demand response (DR) is broadly defined as a measure for reducing energy load in response to supply constraints, generally during periods of peak demand."

Electricity congestion

Electricity congestion is a consequence of the utility grid transmission constraints, characterized by a finite utility grid capacity which prevents the delivery of power from an associated set of power transactions [82].

Gradient descent

Gradient descent is an optimization algorithm which is used to minimize some function. This is done by iteratively proceeding in the direction of steepest descent as defined by the negative of the gradient [59].

Information and Communications Technology (ICT)

Information and Communications Technology (ICT) is an umberella term for all the various medias and technologies employed in the task of communicating information [34].

Internet of Things

Internet of Tings is defined by Gartner Inc. [38] as:

> "The Internet of Things (IoT) is the network of physical objects that contain embedded technology to communicate and sense or interact with their internal states or the external environment."

Linear regression

Linear regression is defined by the Machine Learning Glossary [81] as:

> "Linear Regression is a supervised machine learning algorithm where the predicted output is continuous and has a constant slope. It's used to predict values within a continuous range, (e.g. sales, price) rather than trying to classify them into categories (e.g. cat, dog)."

Logistic regression

Logistic regression is defined by the Machine Learning Glossary [45] as:

> "Logistic regression is a classification algorithm used to assign observations to a discrete set of classes. Unlike linear regression which outputs continuous number values, logistic regression transforms its output using the logistic sigmoid function to return a probability value which can then be mapped to two or more discrete classes."

Metrology

Metrology is defined by the International Bureau of Weights and Measures (BIPM) [97] as:

> "The science of measurement, embracing both experimental and theoretical determinations at any level of uncertainty in any field of science and technology."

Measurement redundancy

Measurement redundancy is an important concept in State Estimation, because increasing the number of measurements made on a system beyond the minimum (i.e., the number of degrees of freedom in the system) can reduce the effect of measurement errors [20]. This increase in measurement above the number that is required to uniquely determine the state of the system is measurement redundancy. The more the redundancy, the better it is to process bad data successfully with the help of a state estimator and get a reasonable estimate of the state. The concept of measurement redundancy reflects the independence degree of an estimated value on the measured value.

Natural monopoly

Natural monopoly is defined by the Organisation for Economic Co-operation and Development (OECD) [55] as:

> "A natural monopoly exists in a particular market if a single firm can serve that market at lower cost than any combination of two or more firms.
>
> Natural monopoly arises out of the properties of productive technology, often in association with market demand, and not from the activities of governments or rivals (see monopoly).
>
> Generally speaking, natural monopolies are characterized by steeply declining long-run average and marginal-cost curves such that there is room for only one firm to fully exploit available economies of scale and supply the market."

Neural networks

Neural networks are defined by the Machine Learning Glossary [58] as:

> "Neural networks are a class of machine learning algorithms used to model complex patterns in data sets using multiple hidden layers and non-linear activation functions. A neural network takes an input, passes it through multiple layers of hidden neurons (mini-functions with unique coefficients that must be learned), and outputs a prediction representing the combined input of all the neurons."

Power-sensitive information

Power-sensitive information is defined by NVE [14, 96] as:

> "Power-sensitive information is specific and in-depth information about plants, functions, systems, and other power supplies that can be used to inflict damage or disrupt the supply of power if the information becomes known to unauthorized persons."

Pseudo random number generator

Pseudo random number generator may be defined as [17]:

> "A pseudo random number generator is a deterministic algorithm which approximates a true random number generator"

Reproducibility

Reproducibility in science is defined by the U.S. National Academies of Sciences, Engineering, and Medicine [72] as:

> "[...] obtaining consistent computational results using the same input data, computational steps, methods, code and conditions of analysis."

Script

Scripts (in computer programming) may be thought of as a sequence of instructions that are interpreted or carried out by another program [74].

sklearn

`sklearn` (or `scikit-learn`) is a popular machine learning library developed for the coding language Python [79].

State Estimation (SE)

State Estimation is an analysis function used in analysis of electric power system. Using redundant measurements, a state estimator estimates the current state of a power system [39].

Time series

Time series are defined by the University of Oxford [92] as:

> "A series of values of a quantity obtained at successive times, often with equal intervals between them."

True random number generator

True random number generator may be defined as [17]:

> "A true random number generator is a physical process which outputs each valid string independently with equal probability"

List of Figures

List of Tables

1 Introduction

1.1 Background

The Norwegian power system will in the coming years undergo changes due to an increasing *electrification* of the transportation sector, a *decentralisation* of power production meaning large-scale integration of distributed/renewable energy resources and electrical storage, and *digitalization* i.e. the increased connectivity of electronic devices and the conversion of information to a digital format [4].

One aspect of the digitalization of the Norwegian power system is the installation of Advanced Metering Systems (AMS) at each point of measurement in the distribution grid. The AMS-meter is able to capture a wide variety of data types in real time. There exist many potential applications of AMS-data, but Norwegian DSOs are not yet utilizing the opportunity to a large extent. This became apparent while working on the specialization project [96], and became the inspiration for the topic of this Master's thesis (from here on termed the report). It became clear that Norwegian DSOs today have to a lesser extent exploited the opportunity to use AMS-data, while other market players already are utilizing such data.

This report is written for the Department of Electric Power Engineering at the Norwegian University of Science and Technology (NTNU) in collaboration with Lyse Elnett and the Centre for Intelligent Electricity Distribution (CINELDI), and looks into potential applications of AMS-data. The idea was presented to Lyse Elnett, which consented to provide AMS consumption data for analysis in the report.

The initial task of this work was to identify an area for analysis of benefit to DSOs, which could be based on the provided AMS consumption data. It was decided to develop a computer-based tool to perform said analysis. The ambition was to base the development of the tool on freely accessible programs and algorithms. Furthermore, it was a wish that the tool could be used by DSOs or other interested parties after the completion of this report.

1.2 Scope

As mentioned in Section 1.1, the initial ideas for this report emerged while working on the specialization project [96]. The objective of this study is to illustrate that opportunities and benefits within certain areas for DSOs can be achieved from the analytical use of AMS-data by applying existing and openly available advanced data analysis tools. Such tools should be possible to utilize by DSO engineers with a power systems background and an information technology interest.

The report will be divided into two parts, Part I and Part II, where Part I will focus on developing the background for the problem formulation, such as the fundamentals of the Norwegian power system, AMS-meters and their application, as well as the use of AMS-data by Norwegian DSOs. An area for applying analysis of AMS-data will be selected, which will be further investigated in Part II of the report.

Part II will be concerned with the development and assessment of a computer-based programme for the selected area of analysis of AMS-data. The background for selecting the respective advanced data analysis tool components for the computer based programme shall be presented. Furthermore, a method for evaluating potential outlier data will be sought included in the programme. Thereafter the developed programme will be assessed based on both known/verifiable input data as well as actual AMS-data provided by Lyse Elnett.

Suggestions for improvements of the developed programme shall also be included.

It will not be within the scope of this report to perform "after-analysis" of customer groups in AMS-data sets. It will also not be within the scope of the report to study the consumption behaviour of different customer types. Furthermore, it is not within the scope to perform tests of several different advanced data analysis tool component types.

1.3 Author's note to contributors

I (the author) would like to thank everyone who has contributed to the writing of this report.

Firstly, a note of thanks should be handed to my supervisor, Vijay Venu Vadlamudi, for support and for giving me the opportunity to be explorative and creative in the writing of this report.

Secondly, a note of thanks should be handed to my contacts at Lyse Elnett. Particularly Aina Romani Dalmau Serigstad for supplying the bulk of the AMS-data analysed in this report, and Siri Torgersen Ravndal for helping me out of a challenging situation which arose close to the due date of the report.

Thirdly, a note of thanks should be handed to all companies which answered enquiries and surveys in relation to this report; Elvia (Hafslund), BKK Nett, Skagerak Nett, Agder Energi Nett, Norges Vassdrags og Energidirektorat, Reguleringsmyndigheten for Energi, Energisalg Norge, `tslearn` and Trønderenergi.

Lastly, a note of thanks should be handed to my father, Jan-Erik Walstad, for helping me with the editorial aspect of this report with such enthusiasm.

2 Reader's guide

This report is divided into two main parts; **Part I** and **Part II**.

2.1 Part I: Fundamentals and problem formulation

The initial ideas for this master's thesis emerged during work on the author's specialization project [96]. During work with this report it appeared that Norwegian DSOs currently have not to a significant extent exploited the opportunity to use AMS-data, while such data is already utilized by other market players, e.g. Smartly and NODES AS. **Part I** of the report provides the knowledge background which is considered necessary for the research topic for this report.

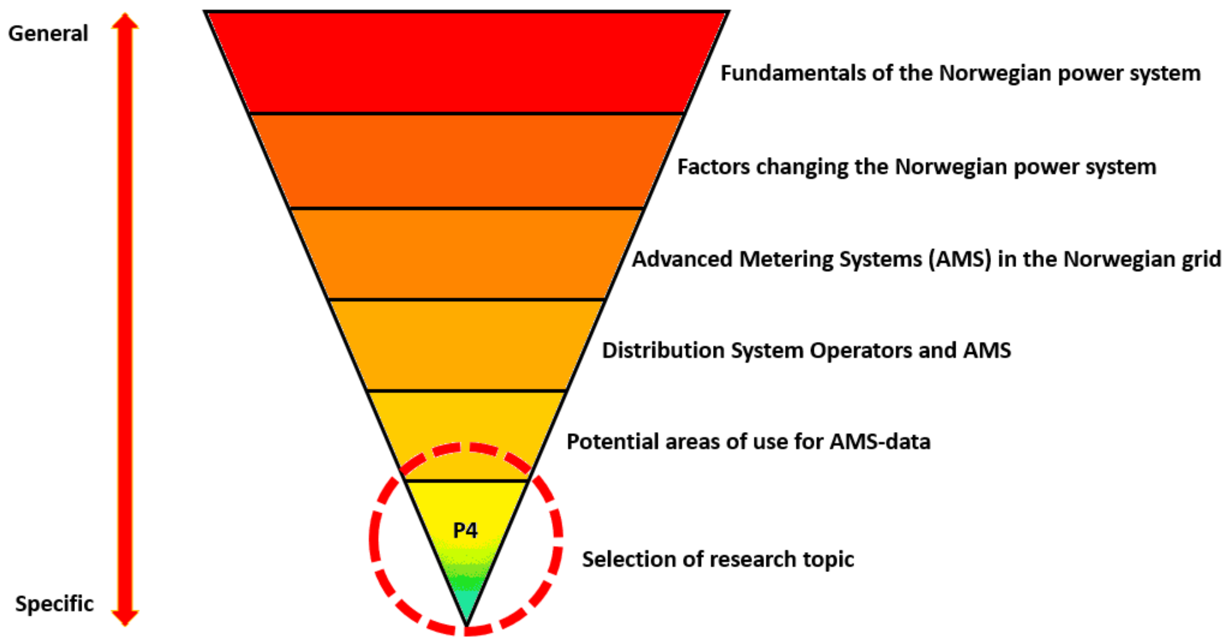

General
Specific
Fundamentals of the Norwegian power system
Factors changing the Norwegian power system
Advanced Metering Systems (AMS) in the Norwegian grid
Distribution System Operators and AMS
Potential areas of use for AMS-data
P4
Selection of research topic

Figure 1: Illustration of Part I content of the report.

The following report Sections are included in **Part I**:

Section 3: Provides a fundamental understanding of the Norwegian power system, as well as an overview of the different factors influencing change in the Norwegian power system.

Section 4: Describes Advanced Metering Systems (AMS), and their use by Norwegian DSOs.

Section 5: Looks into how the Norwegian Distribution System Operators (DSO) are currently utilizing AMS-data, as well as some potential future areas of use. The topic of research for this report, P4, is also selected in this Section.

2.2 Part II: Method development and assessment

Part II of the report starts by explaining the conceptual framework for the development of a computer-based programme for addressing the selected topic of research, P4 customer segmentation. Following this, the theoretical background for the various parts of the developed customer segmentation programme is explained. Following this, the customer segmentation programme is assessed and tested. The assessment is first performed with known data sets to see whether the programme produces results which are known beforehand to be correct. Thereafter the customer segmentation programme is tested on AMS-data provided by Lyse Elnett, before it is finally compared with the current standard way that DSOs segment their customers into groups.

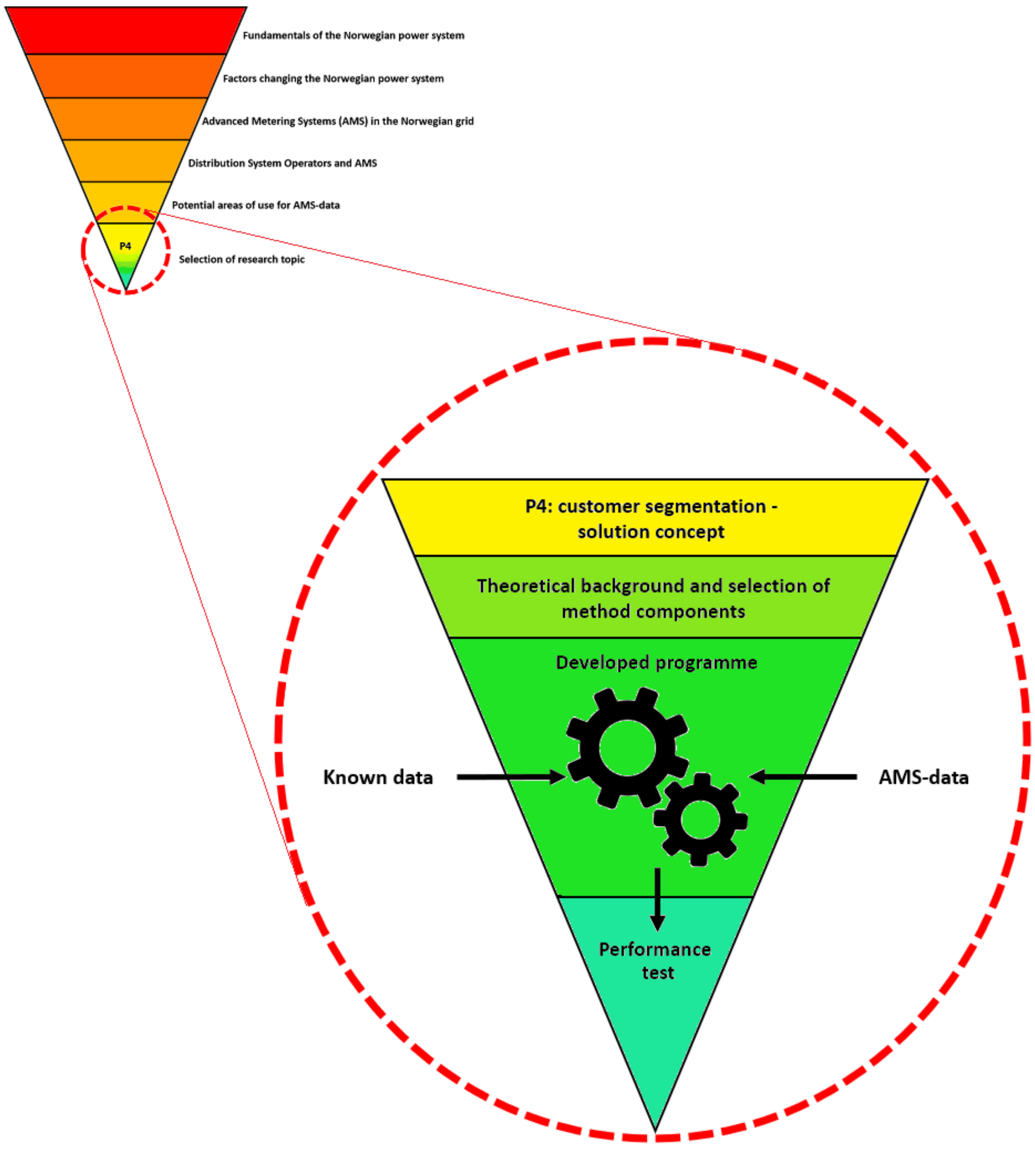

Fundamentals of the Norwegian power system
Factors changing the Norwegian power system
Advanced Metering Systems (AMS) in the Norwegian grid
Distribution System Operators and AMS
Potential areas of use for AMS-data
P4
Selection of research topic
P4: customer segmentation - solution concept
Theoretical background and selection of method components
Developed programme
Known data
AMS-data
Performance test

Figure 2: Illustration of Part II content of the report.

The following report Sections are included in **Part II**:

Section 6: The approach to developing the P4 solution concept is described, as well as an introduction to Python.

Section 7: The theory behind the P4 solution concept is presented, and an algorithm for each part of the P4 solution concept is selected. Additionally, methods for refining the data set is explained.

Section 8: The developed programme for shape-based customer segmentation is presented.

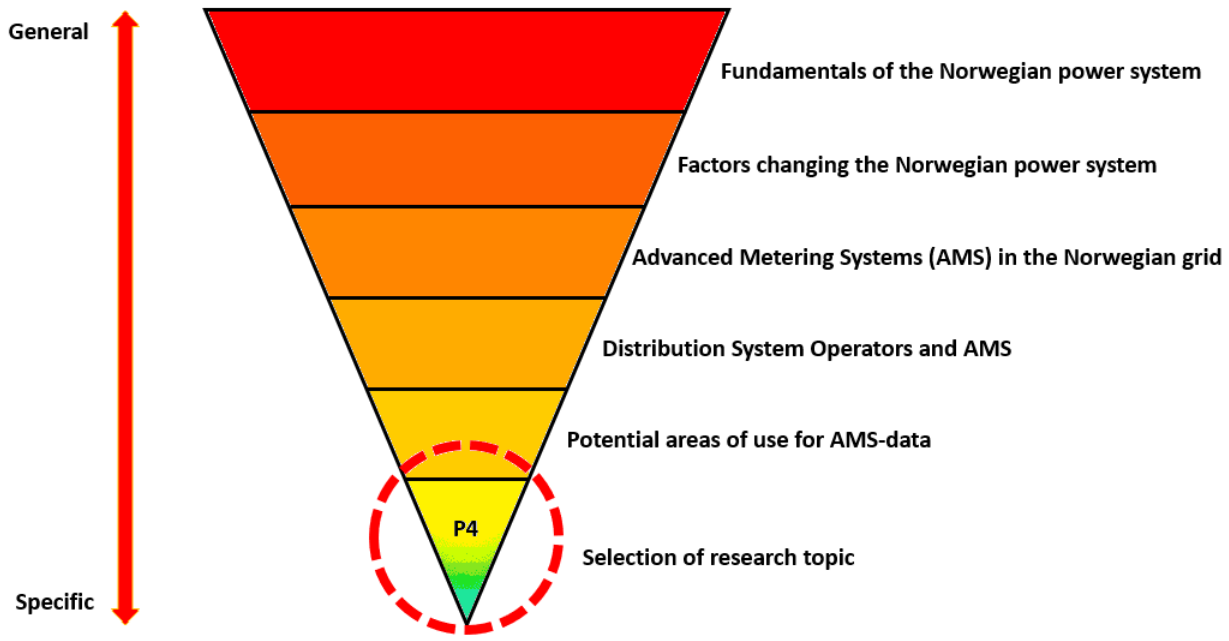
Section 9: Assesses the developed customer segmentation programme with know time series data.

Section 10: Assesses the developed customer segmentation programme with AMS time series data. A comparison is done between the developed customer segmentation programme and the standard DSO customer segmentation method.

Section 11: Report conclusions.

Section 12: Potential improvements on the developed customer segmentation programme, suggestions for further work and general notes on the writing of the report.

Part I: Fundamentals and problem formulation


General
Specific
Fundamentals of the Norwegian power system
Factors changing the Norwegian power system
Advanced Metering Systems (AMS) in the Norwegian grid
Distribution System Operators and AMS
Potential areas of use for AMS-data
P4
Selection of research topic

3 Fundamentals of the Norwegian power system

This Section explains the fundamentals of the Norwegian power system and power market, which represents a basis for the topic of this report. Additionally some of the important current factors which are influencing and changing the Norwegian power system are discussed, to give the reader an impression of how the Norwegian power system could change in the coming years.

3.1 Regulatory authorities

The regulation of Norwegian Distribution System Operators (DSO) is achieved at three different levels. The Royal Norwegian Ministry of Petroleum and Energy (in Norwegian called Olje- og Energidepartementet, OED) is at the authoritative top, followed by the Norwegian Water Resources and Energy Directorate (Norges Vassdrags- og Energidirektorat, NVE). Then follows the Regulatory Authority for Energy (Reguleringsmyndigheten for Energi, RME). This hierarchy is illustrated in Figure 3.

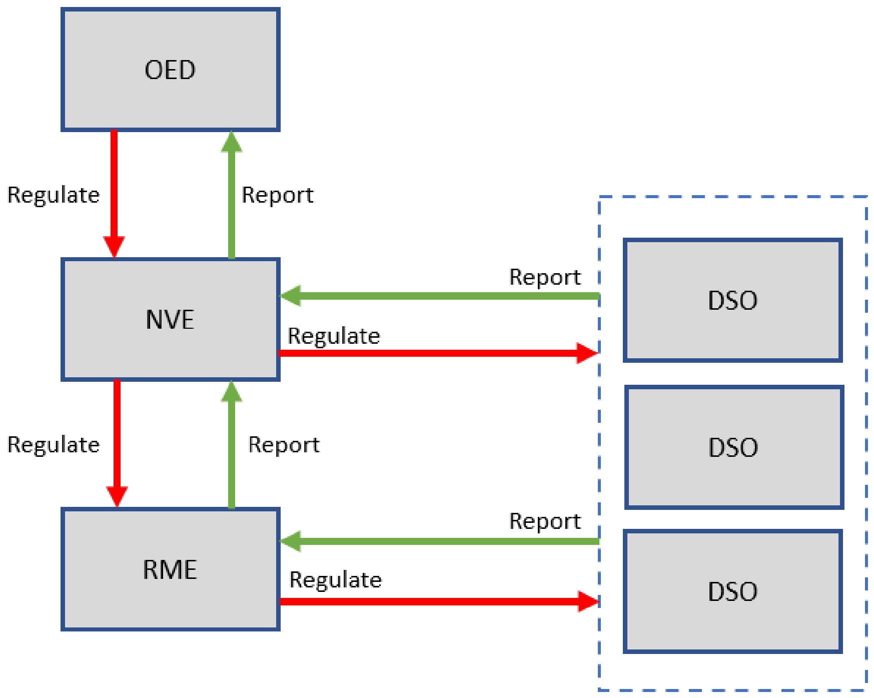


Figure 3: Hierarchy of the Norwegian regulatory authorities, in relation to DSOs.

OED is the Norwegian ministry responsible for energy, including its production, transport and trade. The OED reports to the legislature, i.e. Stortinget (Parliament) [63].

NVE is the Norwegian governmental directorate which regulates the national water resources and the related energy supply. NVE is organised under OED, and has administrative responsibility for the Watercourse Regulation Act (1917), Industrial

Concession Act (1917), Energy Act (1990) and Water Resources Act (2000). The directorate may issue new regulations, and is obliged to consider regulations issued by the European Union which affect any regulations within its purview [62].

From November 1, 2019, the RME has been designated by the OED as a regulatory authority pursuant to Section 2-3 of the Energy Act and Section 4 of the Natural Gas Act. These tasks shall be performed by RME as an independent regulatory authority [71]. RME does in some cases report directly to OED, but not concerning the Energy Act as NVE holds such administrative responsibility [62]. Therefore, as seen in Figure 3, RME does not report directly in the hierarchy to OED concerning regulation of DSOs.

3.2 The Energy Act (1990)

The Energy Act (Energiloven) describe the rules which govern the generation, conversion, transmission, trading, distribution and use of energy [46]. The Act ensures that the generation, conversion, transmission, trading, distribution and use of energy are conducted in a way which efficiently promotes the interests of society [66].

An area licence is needed for anyone who wishes to build, own and operate an electrical installation for distribution of electric energy between voltage levels within a specific location. The holder of an area licence is obligated to provide electrical energy to the customers within the geographical area in which the licence is applicable, as well as affiliation of electric production facilities. In Norway, only DSOs are granted local area licences. Statnett is the Norwegian Transmission System Operator, and is in this capacity granted the national area licence [46, 66, 96].

To trade electrical energy, a trading licence for electrical energy is needed. Norwegian DSOs are not granted licences for trade in electrical energy. Hence, DSOs are not allowed to own or be owned by entities operating in production or sale of electrical energy [46], §4-6. Trading licences are granted to power producers and power companies. Trade in electrical energy occurs in the power market, which is discussed in Section 3.4.

3.3 The power system

The electric utility grid in the Norwegian power system is divided into three voltage levels. These are the high voltage grid (Sentral- or Transmisjonsnettet in Norwegian) with voltage levels between 132 and 420 kV, the medium voltage grid (Regionalnettet in Norwegian) with voltage levels between 33 and 132 kV, and the low voltage grid (Distribusjonsnettet in Norwegian) with voltage levels between 230 V and 22 kV [42].

The high voltage grid is operated by the Transmission System Operator (TSO) Statnett. The low voltage grid, in addition to some parts of the medium voltage grid is operated by DSOs. Which parts of the medium voltage grid that a DSO may operate is determined by Statnett. Statnett is wholly owned by the state, while DSOs are wholly owned by the municipalities in which they operate the low/medium voltage utility grids [42].

Power production companies are not necessarily state- or municipality owned. Their ownership will largely depend on how much power they produce annually, and from which sources. Power companies which sell power from the power production companies to various customers are also privatised to some extent [42].

Statnett is responsible for securing the instantaneous balance between the production and consumption of power in Norway. This in essence means that Statnett is responsible for keeping the electric frequency in the entire utility grid balanced at approximately 50 Hz [84].

3.4 The power market

The Energy Act lays the foundation for market-based electric power production and trade, and for clear regulation of the participants in the power market. The power market aims to ensure that resources are utilised efficiently, and that electrical power stays cost-efficient for the electric power customer [41].

In 1991 Norway introduced market-based trading of electric power. From the very start, the Norwegian power market was accessible to a wide variety of participants. Early on, the power exchange was operated by Statnett Marked AS, and today is operated by Nord Pool AS. The power exchange has become an important and integral part of the Norwegian power market. Nord Pool has since evolved to become the worlds first international power exchange, and now operates power trading markets in 14 countries in the Nordic, Baltic and continental European area [41].

The power market may be divided into two segments, the *wholesale power market*, and the *end-user power market* [41].

In the wholesale power market, large volumes of electrical power are traded every day. Large customers (consumers) which operate in the wholesale power market may choose to buy power from power companies, Nord Pool or directly from power producers. Smaller customers (consumers), which can exclusively operate in the end-user power market, can only buy power from power companies. Figure 4 illustrates this [41].

The structure of the Norwegian power market, and the interaction of the different market players may be visualised as in Figure 4.

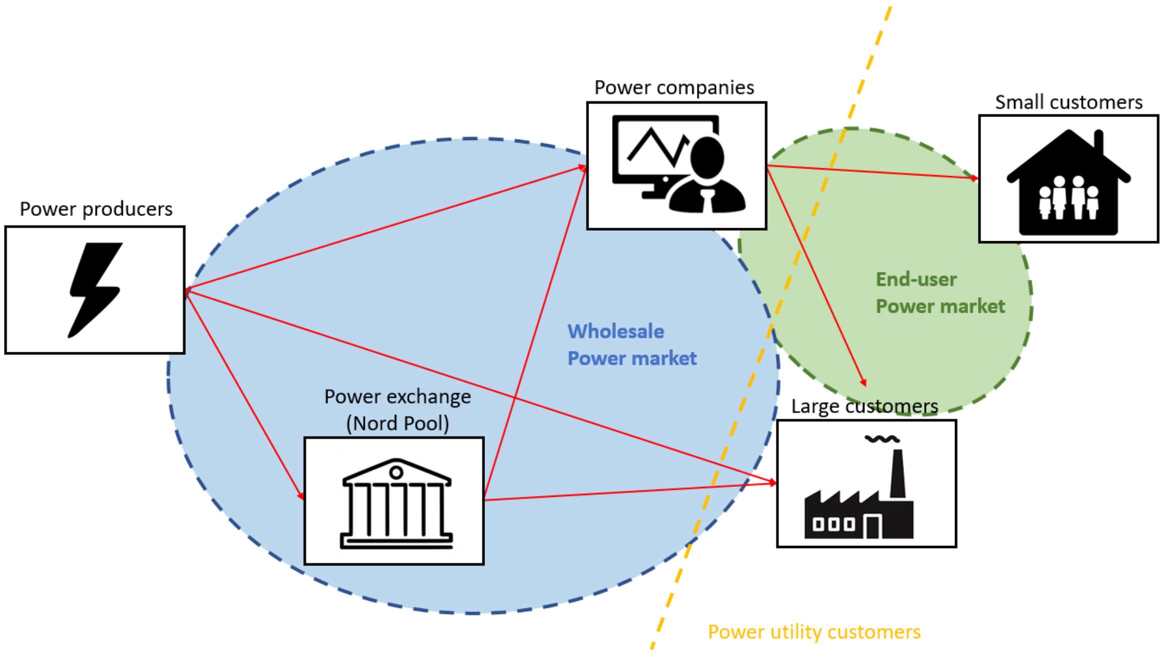


Figure 4: The general structure of the power market and its main players, adapted from [41].

The wholesale power market comprises of the day-ahead market, the continuous intraday market and the balancing markets. Today the TSO Statnett is the only Norwegian market player that can buy power in the balancing market at Nord Pool [73].

DSOs are required to buy power to cover the losses in their respective distribution grids, and to provide power to specific utility grid customers that are not affiliated with any power companies [43, 56]. DSOs fall into the category of "Large customers" in Figure 4 that buy power from power companies. This has been explicitly verified for this report by contacting NVE and some of the largest DSOs in Norway, i.e. Hafslund, BKK nett, Skagerak nett and Lyse Elnett, in addition to power companies such as Trønder Energi and Energisalg Norge.

3.5 Distribution System Operators

A Distribution System Operator (DSO) is defined in Norway by Energi Norge [54] as:

> a natural or legal person responsible for the operation, maintenance and if necessary, development of a distribution network in a given area and, where appropriate, electric connection to other networks, as well as a responsibility to ensure the network's ability for the long term to meet a reasonable demand for the distribution of electric power.

Norwegian DSOs are directly regulated by NVE and RME. DSOs are subject to extensive regulation as they are considered natural monopolies in their distribution area. Development of utility distribution grids is associated with high fixed costs and low variable costs. Thus, in Norway, it is not considered economically optimal to have more than one market player for a particular distribution area [65].

In Norway there exists around 136 DSOs, of which 103 have less than 10'000 customers each [42]. Lyse Elnett is one of the larger DSOs with over 150'000 affiliated customers.

3.5.1 Economic regulations for DSOs

Because DSOs are natural monopolies in their licensed network areas, their incomes are strictly regulated. RME is in charge of the economic regulation of DSOs and therefore decides how much income each DSO is allowed to make. Once a year RME calculates an "allowed income" for each DSO. At the end of each year RME checks whether the realised income has been higher or lower than the allowed. If a DSO has made a surplus income, the surplus, including interest, must be paid back to the utility customers in the form of reduced network costs the following year. If a DSO has made a deficit compared with the allowed income, the deficit can be recuperated by increasing the network costs in the following year [64].

In addition, RME measures the efficiency of DSOs. More efficient DSOs are rewarded, and less efficient DSOs are penalised. The more efficient DSOs are rewarded with a higher allowed income, while the less efficient DSOs are penalised with a lower allowed income. The larger the allowed income, the more a given DSO may charge utility grid customers in network costs, thereby achieving a higher return [64].

3.5.2 DSO IT systems

The most important IT systems used by Norwegian DSOs today (2020) are Supervisory Control And Data Acquisition (SCADA), Energy Management System (EMS), Distribution Management System (DMS), Outage Management System (OMS) and Network Information Systems (NIS) systems. Appendix I contains the results of research done into the operation and hierarchy of the different IT systems carries out in preparation for this report. More information on all the above mentioned DSO IT systems may be found in Appendix I.

3.6 A changing power system

In a bid to keep Norway at the forefront of technological development related to the electric power system, NVE has developed new regulations to act as a stimulus for technological change. These regulations point towards a growing sentiment in the industry to digitalize and modernize the electric power system. NVE's [60] yearly report of 2019 highlighted some of the main development trends for the future Norwegian power system. The motivation for many of these stem from legislation that the Norwegian government [36] passed in 2015, stating that it would reduce the country's carbon emissions by at least 40% by 2030, compared to 1990. In addition, OED and NVE have published aims towards 2040 for electrification of the transport sector and decentralization of power production [60].

Electrification, decentralization and digitalization are three concepts that are central in describing what is predicted to influence how electric power production, distribution/transmission and consumption behaviour will change in the future [47].

This Section will highlight some regulations and aims of government bodies such as OED, NVE and the EU that may either introduce new market players to the power market, or may change behaviour of existing players.

3.6.1 Electrification of transport

The transport sector is responsible for around one third of Norway's carbon emissions, and since transport is not included in the European Union's emissions trade (cap and trade), large reductions in emissions from the transport sector will be required [51]. This will mean an increase in demand for electric power for transport of around 30-40 TWh per year (Statnett [32] estimate).

3.6.2 Decentralization of power production

NVE [3] estimates that 19-38 TWh of wind power will be installed in Norway between 2019 and 2040, and that solar power production will increase, but by an unpredictable amount. More information on the predicted development of renewable and decentralized power production in Norway towards 2040 may be found in resource [3].

3.6.3 EU regulations on electricity balancing

In 2017 the European commission [21] published a set of new regulations, "Establishing guidelines on electricity balancing", which are mandatory for all members of the European Economic Area (EEA, known as EØS in Norwegian) and EU member states to implement [24]. These new regulations will give DSOs the ability to manage local electricity congestion in the grid of their lisenced distribution area [3]. This is contradictory to the current practice, where the TSO Statnett has sole responsibility for all electricity congestion in the Norwegian utility grid. A consequence of the new EU regulations may be that DSOs in the future will buy power in the balancing markets, and/or the emergence of new type of players in the Norwegian power market i.e. "aggregators".

3.6.4 The aggregator role

A commercial player that sells demand response, and uses a commercial control system for the demand response actuation is termed an aggregator [85]. The aggregator is a new role in the Norwegian power market. In order to operate as an aggregator NVE will need to grant the commercial player a trading licence for aggregator activities [44].

3.6.5 Information and Communication Technology (ICT) in the utility grid

A variety of equipment with Information and Communications Technology (ICT) functionality is being installed in both transmission and distribution utility grids in Norway. The TSO Statnett has for example put automatic system circuit breakers to use in the transmission grid, to comply with §21 in the regulation on system responsibility of the power system [28]. Advanced Metering Systems (AMS) is also now mostly installed in all relevant measurement points in the utility grid, more on this in Section 4.

As more and more ICT equipment is being installed in the utility grid, the grid may increasingly be considered a "smart grid". The defining quality of a smart grid is its ability to detect and react to local and/or global changes in its state by the use of ICT technology with a multi-directional flow of information [90].

3.6.6 Elhub

NVE [53] has given Statnett the task of developing "Elhub". The function of Elhub will be to collect and store AMS-data from the Norwegian distribution utility grid. More detailed information on AMS-meters and AMS-data may be found in Section 4.

Data in Elhub is made accessible to specific market players in the Norwegian power market like DSOs, aggregators and power companies. An intention of the development of Elhub is to provide stimulus for innovation in, and optimisation of, the power market and power system. Elhub will also ensure information security for the AMS-data preventing adversaries gaining insight into and/or control over AMS-data [53].

4 Advanced Metering Systems

An Advanced Metering System (AMS) meter is sometimes called a “smart meter”. The main purpose of an AMS-meter is to record electric power consumption in its measurement point, and communicate the recorded information to the affiliated utility company. However, AMS infrastructure allows for two-way communication between a utility/power company and the AMS-meter over an ICT network. This two-way communication is required of the AMS-meter due to its additional functionality, as described in Section 4.1.1 i.e. the functionality which makes the AMS a “smart meter”.

4.1 Regulations related to AMS-meters

The Regulations on Power Trade and Utility Grid Services (Forskrift om kraftomsetning og nettjenester) regulates electrical energy measurement and settlement, billing of network services and electrical energy, the neutrality of utility grid operators, etc [61].

Chapter 4 “Advanced metering systems” was included into the Regulations on Power Trade and Utility Grid Services in 2011. RME is responsible for regulation of Chapter 4. The inclusion of Chapter 4 was done to, inter alia, provide necessary information for utility grid customers to control their own electrical power consumption, and to contribute to increased opportunity for DSOs to optimize operation of their utility grid [61].

According to Chapter 4, §4-1, utility grid companies are obligated to install AMS-meters at every point of measurement in the utility grid, with two exceptions given in §4-1 a) when electricity consumption at a point of measurement is low and unpredictable, or b) when installation of a meter represents a significant and verifiable disadvantage to the utility grid customer. Following §4-5 all measurement points should have AMS-meters installed within the 1st of January, 2019 [61], i.e. all AMS-meters should have been installed at the time of writing this report (spring 2020).

4.1.1 Further details on AMS regulations

According to §4-2, the sampling frequency of an AMS should be at least once per 60 minutes, with the possibility to increase the frequency to once every 15 minutes (§4-2 a)). All AMS-meter are required to comply with open standards for communication (§4-2 b)). An AMS-meter should also be able to communicate with other measuring devices, and should retain measured data in case of a power outage (§4-2 c)-d)). The AMS-meter should be able to both limit and/or cut power flow, and measure flow of active and reactive power at its point of measurement (§4-2 e) & h)). It should also be able to receive information on electric power prices and tariffs, and transmit data on control or short circuit to ground signal errors (§4-2 f)). Finally, an AMS should provide adequate information security such that measurement and control functions are not obtained by an adversary (§4-2 g)) [61].

Following §4-3, an AMS-meter should be capable of storing measurement data until such data has been transmitted to the relevant DSO, at least the duration of time until the due date of the current billing period [61].

According to §4-4, power companies should be able to send price information, and DSOs should be able to send tariff information to AMS-meters. As such, a utility grid customer will be able to read electric power price, and tariff information from their respective AMS-meter [61].

By §4-6, a utility grid customer should have local access to their respective AMS-meter measurement information.

4.2 AMS-meters in the Norwegian distribution utility grid

During the rollout of AMS-meters in the Norwegian distribution utility grid, a variety of different meter models were installed. This section will present some of the different types of AMS-meters installed. At any given location in the distribution utility grid, the selected type of meter would depend on the voltage level at that location, number of phases, and whether the selected point of measurement was at a utility grid customer or at a transformer.

According to RME, three AMS producers share the Norwegian market at distribution level customer measurement points. These are Aidon (ca. 50%), Kamstrup (ca. 25%) and Kaifa (ca. 25%). There exists other AMS-vendors that supply AMS-meters for utility customers but they constitute around 1% of all installed AMS-meters and will therefore not be discussed any further in this report [93]. Consequently all AMS-meters installed by DSOs after §4-1 mentioned in Section 4.1 are assumed provided by one of the three above mentioned companies. Figure 5 shows the three types of AMS-meters from the three producers which are most commonly installed in small-scale utility grid customers.

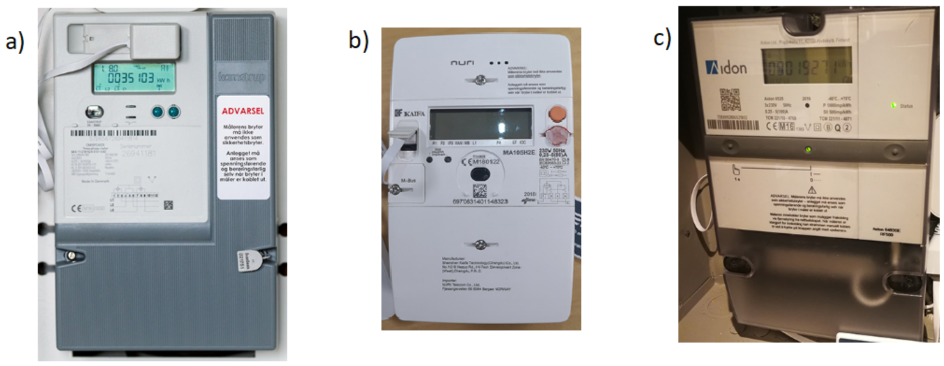


Figure 5: AMS-meters from the three producers a) Kamstrup, b) Kaifa and c) Aidon [22].

Figure 6 shows a map of the different DSO license areas in Norway. The DSOs commonly have exclusive contracts with one of the three AMS vendors. The coloured areas on the map indicates some of the areas where the vendors have exclusively provided AMS-meters (areas which are not coloured lacked public information on which AMS vendor that provided AMS-meters. This map is unfortunately partially outdated as for example Hafslund and Eidsiva Nett have merged in 2019-2020 to form Elvia, which is not displayed on the map in Figure 6. The most up to date (March 2020) map of Norwegian DSO licence areas may be found in reference [57]). All in all, around 42 Norwegian DSOs use AMS-meters produced by Aidon, 37 use AMS-meters produced by Kamstrup, and 26 use AMS-meters produced by Kaifa [93].

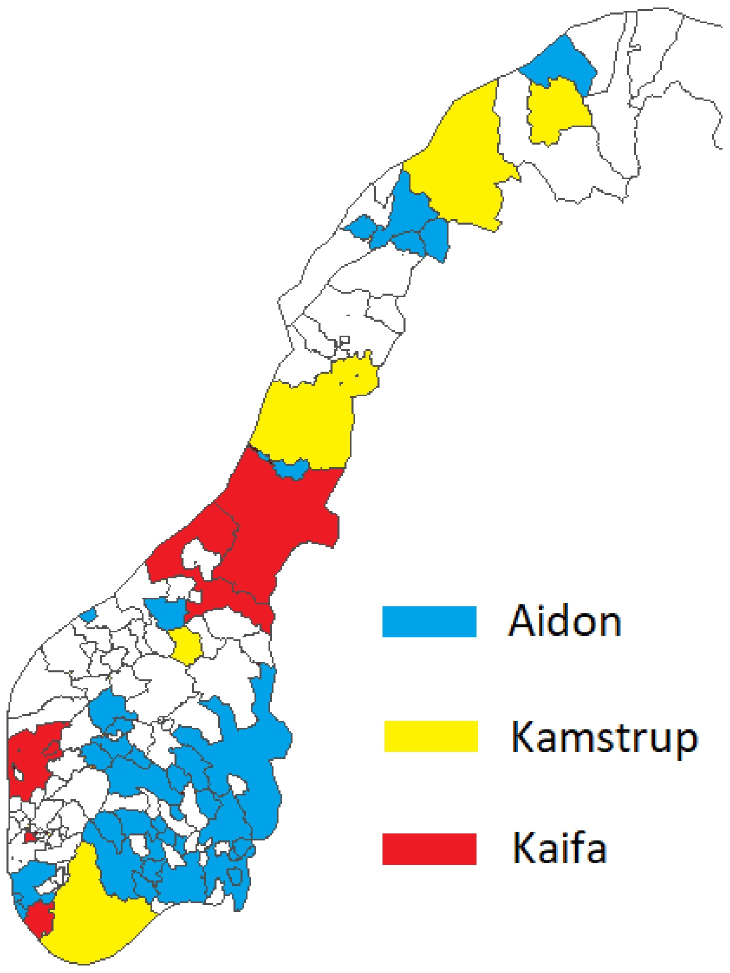


Figure 6: Map of the three AMS-vendor's AMS locations among Norwegian DSOs.

4.2.1 AMS, DSO and Elhub interaction

Through dialogue with BKK nett and Lyse Elnett conducted for this report, the interaction between DSO, AMS-meters and Elhub has been outlined.

All AMS-data is sent to Elhub in real-time through an intermediary information and telecommunications entity or company. After each day's (24h) settlement has been calculated, the AMS-data is sent from Elhub to the relevant DSO. DSOs may also access AMS-data in near real-time from their connected AMS-meters, which may be updated every three hours. It is possible that other DSOs manage this interaction slightly differently.

4.3 AMS-meters

A typical AMS-meter will include a combination of software, hardware and calibration systems. Each AMS-meter should include systems for metrology, information security, and communication both between components of the AMS-meter, and between the meter and external units [16].

A smart meter system may include; a time-keeping device (clock), a data communications module, a computing and processing unit, tamper detection, transformer driver and voltage reference. Further the meter will contain an analog-to-digital converter. When a voltage level is measured, it will be processed by the computing and processing unit, stored in the correct format where a time stamp is added, and sent over an ICT network to a DSO [16]. In the case of AMS-meters installed in Norway, communication between an AMS-meter and an external unit happens through a radio-mesh network of limited bandwidth [35].

To comply with §4-3, Section 4.1.1, an AMS-meter should have an emergency backup power source in the form of a battery. An anti-tampering circuitry should also be added. This is done to comply with §4-2 g) as mentioned in Section 4.1. Extra functionality such as a display may be added to the AMS-meter according to §4-4 [16, 61].

To illustrate what specifications an AMS-meter installed in the Norwegian distribution grid has, it may be helpful to look at the three series of meters that Aidon installs. Since all AMS-meters installed must comply with Chapter 4. of the Regulations on Power Trade and Utility Grid Services, as mentioned in Section 4.1, the AMS-meters produced by Kamstrup and Kaifa should be equivalent to the ones produced by Aidon. The different types of Aidon AMS-meters are [37]:

- The Aidon 6510-series take single-phase measurements of electric power consumption for utility grid customers, at a nominal voltage level of 1x230 V.
- The Aidon 6520- and 6530-series take thee-phase measurements of electric power consumption for utility grid customers, at a nominal voltage level of 3x230 V, or 3x230 V/400 V.
- The Aidon 6540- and 6550-series take thee-phase measurements of electric power consumption at a transformer, at a nominal voltage level of 3x230 V, or 3x230 V/400 V.

More detailed information on each of the Aidon AMS-meter series may be found in reference [69].

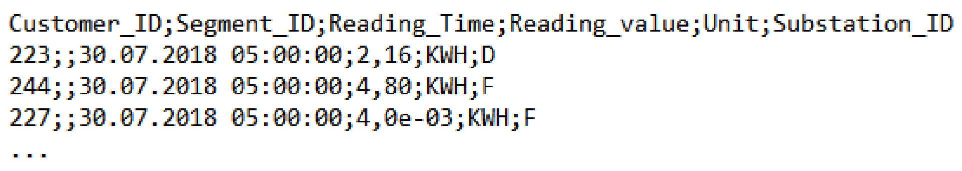
AMS-meters also include an access point termed a Home Area Network (HAN) port. The HAN-port may be used as an interface with the AMS-meter to which commercial actors can develop various services for the meter owners. To activate the HAN-port of an AMS-meter, the relevant utility grid customer will need to contact the DSO of which the AMS-meter is affiliated. It is expected that in the future the HAN-port may enable utility grid customers to participate in for example commercial demand response activities controlled by aggregators [86].

4.4 AMS time-series data

AMS time series data may include all measurement values that the AMS-meter is able to measure. The AMS-data may include physical measurement values (voltage, current, active power, reactive power etc.), AMS-meter IDs, customer segment IDs, timestamps, and possibly an ID which identifies which substation the AMS-meters is connected to. These data types are listed column-wise, and are labeled in the first row of each column.

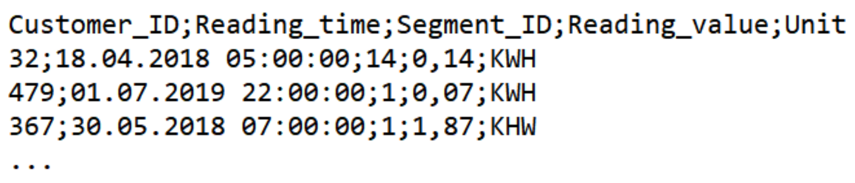
Each column will be separated by a delimiter, for example a semicolon ";". Date values in the timestamps are given in a European date format, which means DD.MM.YYYY, as apposed to the American date format which is of the form MM.DD.YYYY. Time values in the timestamps are given by HH.MM.SS, although the granularity is not required to be more than minute-wise (Section 4.1.1). Decimal places are marked with a comma ",".

Below are two examples of AMS-data from different DSOs (Lyse Elnett in Figure 7, and Elvia in Figure 8). They are both obtained from Aidon meters, as may be understood by studying the map in Figure 6.



```
Customer_ID;Segment_ID;Reading_Time;Reading_value;Unit;Substation_ID
223;;30.07.2018 05:00:00;2,16;KWH;D
244;;30.07.2018 05:00:00;4,80;KWH;F
227;;30.07.2018 05:00:00;4,0e-03;KWH;F
...
```

Figure 7: Example of AMS data collected by AMS-meters affiliated with the DSO Lyse Elnett.



```
Customer_ID;Reading_time;Segment_ID;Reading_value;Unit
32;18.04.2018 05:00:00;14;0,14;KWH
479;01.07.2019 22:00:00;1;0,07;KWH
367;30.05.2018 07:00:00;1;1,87;KHW
...
```

Figure 8: Example of AMS data collected by AMS-meters affiliated with the DSO Elvia.

5 Norwegian DSOs and their use of AMS-data

While working on the specialization project [96] it became apparent that Norwegian DSOs today have the unexploited opportunity to use AMS-data, while some market players already are utilizing such data. Historically AMS-data has primarily been available for consumers with large electric power consumption, typically above 100 MWh/year. As described in Section 4.1 AMS meters are now required for all utility grid consumers enabling collection of this new data. Further, new technology has become easily available for processing large volumes of data, e.g. machine learning and "big data", which this report seeks to investigate.

5.1 Current DSO use of AMS-data

In the context of this report, selected Norwegian DSOs were contacted to research to which extent and purposes they were using their collected AMS-data. Of the contacted DSOs, three answered enquiries before the due date of this report. These were Elvia, Lyse Elnett and BKK Nett. With Elvia's 900000, BKK Nett's 245000 and Lyse Elnett's 150000 utility grid customers, these are some of the largest DSOs in Norway.

Firstly the DSOs were asked how they have implemented the use of AMS-data currently. These areas of implementation are listed as I1 to I6:

- **I1:** Automatic detection of faults in the LV grid.
- **I2:** Automatic detection of ground faults.
- **I3:** Billing.
- **I4:** Detecting wrongly installed AMS-meters.
- **I5:** Tap change transformers.
- **I6:** Aggregate AMS-data to substation level when substation measurements are missing.

Figure 9 shows across how many of the three DSOs that the different areas of AMS-data use (I1-I6) have been implemented.

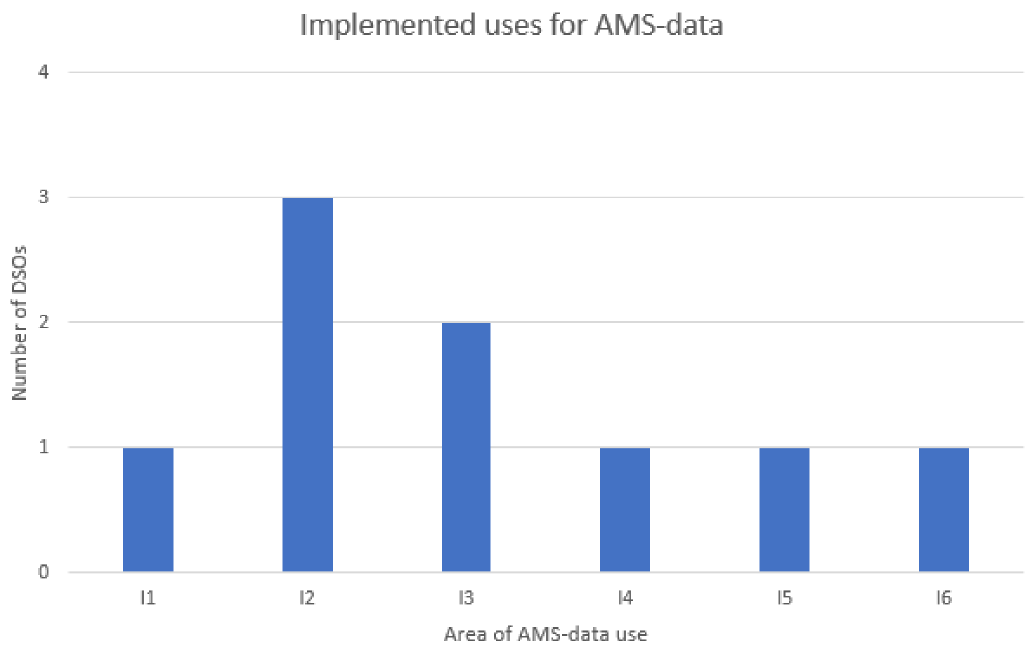


Figure 9: Areas of use for AMS-data which the DSOs Elvia, BKK Nett and Lyse Elnett have implemented.

Secondly, the DSOs were asked which areas of AMS-data use they were planning to implement in the near future. The areas of use that the DSOs are actively planning to implement are listed as I7-I12. Some of the I1-I6 areas of use are not yet implemented by all three DSOs, as may be seen in Figure 9, but are currently being planned for implementation by one or more of the other DSOs.

- I7: Planning transformer size requirements in transformer- and substations.
- I8: Improve voltage quality.
- I9: Customer power consumption prediction.
- I10: Fault and anomaly detection in DMS.
- I11: State Estimation.
- I12: Customer segmentation.

Figure 10 shows across how many of the three DSOs that the different areas of AMS-data use (I1-I12) are planned for implementation.

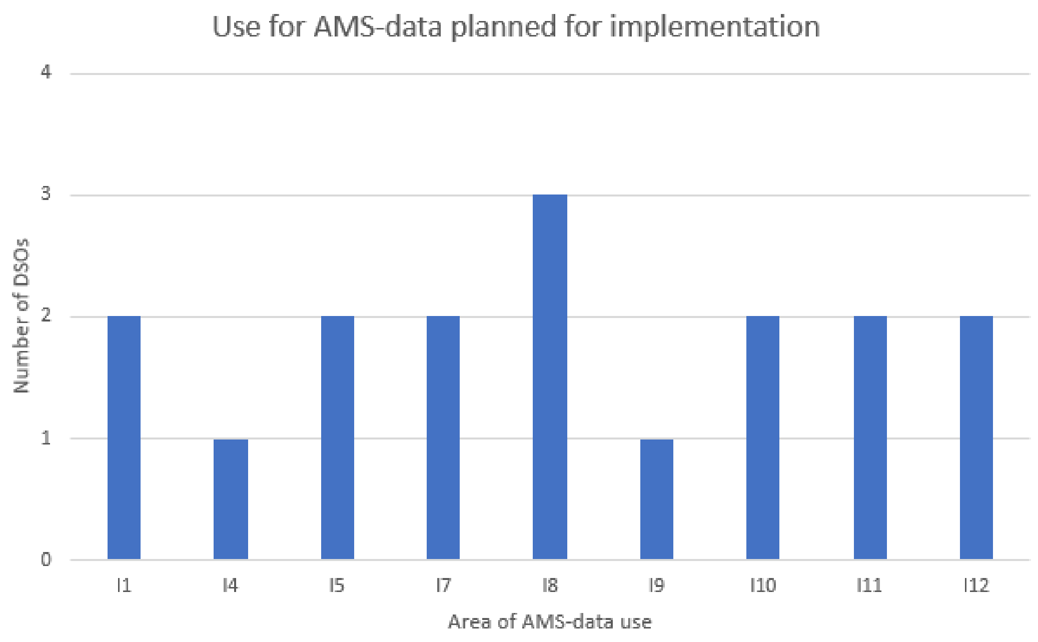


Figure 10: Areas of use for AMS-data which the DSOs Elvia, BKK Nett and Lyse Elnett are planning to implement.

5.2 Potential benefits of more extensive AMS-data use for DSOs

Norwegian DSOs have several reasons to increase their use of utility customer AMS-data and substation AMS-data. Through AMS-data analysis a DSO may gain both economic advantages in addition to improve their ability to remain independent (further outlined below).

As stated in Section 3.5.1, "inefficient" DSOs are granted a lower allowed income for the following year. Therefore, Norwegian DSOs are incentivised to become more efficient. For relating the term *efficiency* to an organization, Investopedia [15] offers a useful definition:

> Efficiency signifies a peak level of performance that uses the least amount of inputs to achieve the highest amount of output. Efficiency requires reducing the number of unnecessary resources used to produce a given output including personal time and energy. It is a measurable concept that can be determined using the ratio of useful output to total input. It minimizes the waste of resources such as physical materials, energy, and time while accomplishing the desired output.

One area where application of AMS-data could improve efficiency of DSOs is related to grid development. Use of AMS-data may also help DSOs in for example billing customers correctly. These examples could help a DSO to become more efficient, and result in the DSO being granted a larger allowed income by RME.

The future status of many DSOs is uncertain because of the new EU [21] regulations on electricity balancing, described in Section 3.6.3, which gives DSOs in Norway the ability to manage local electricity congestion.

In order to manage local electricity congestion, i.e. balance electricity locally in their licensed areas, DSOs will have to buy "balancing power". Though the Norwegian regulations are not in place yet (2020), it is currently expected that a DSO will only be able to buy balancing power which may be activated within its own licensed distribution area [8].

This may represent a future incentive for DSOs to merge in order to ensure that their licensed distribution areas are large enough to incorporate the necessary balancing power reserves [8]. Further, the challenge for DSOs of local electricity balancing may be mitigated with the help of aggregators. Although demand response provided by aggregators is useful in theory, there are still unresolved issues related to the actual implementation of it, particularly for the purposes mentioned in this Section [96]. However, if a DSO is able to better analyze the power consumption of its customers with the help of AMS-data, the DSO could gain an understanding of whether demand response for electricity balancing is achievable within its licence area. If the DSO can demonstrate that demand response with an aggregator could provide adequate electricity balancing, the DSO may not be required to merge.

5.3 Potential challenges for DSOs to actively apply AMS-data in their operations

There are a variety of challenges facing DSOs addressing the issue of unused AMS-data in their company. The most important are:

- An absence of in-house competence may mean that DSOs lack ambition to pursue the use of AMS-data by themselves.
- A number highly competitive IT-companies geared towards the power system are emerging (Cognite, Powel, Rejlers, etc.). There is now a tendency in the industry towards decompartmentalization of DSO data, and development of Application Programming Interface (API) layers on top of the decompartmentalized data which will set the stage for IT-developers to build analysis software on top of the open APIs.
- Power companies may in some instances have a greater incentive to work on utilization of AMS-data than DSOs. This is because, as mentioned in Section 3.4, DSOs have given power companies the responsibility of calculating grid losses and buying the correct volume of power to cover them. This has been concluded from conversations with Lyse Elnett, NVE, Hafslund, BKK Nett and Agder Energi Nett.
- The IT systems (described in Appendix I) that DSOs use today are reliable and have been used for decades. Additionally, it is costly for a DSO to change an IT system. For these reasons DSOs may hesitate to ask the IT system vendors to provide additional AMS functionality.

- The new EU regulations on electricity balancing have not yet been written into NVE's regulations for the Norwegian power system.
- The Norwegian power system's exposure to decentralized/volatile power production has not yet become large enough for DSOs to feel the need for more sophisticated grid management than is in operation today.

Despite these challenges DSOs are motivated to put AMS-data to use for reasons mentioned in Section 5.2. The services provided by companies such as Powel, Cognite and Rejler will not come without economic costs. Exactly how high this cost is is out of the scope of this report to estimate, but it is possible that DSOs could save money by implementing some solutions themselves.

5.4 Potential areas of use for AMS-data to be applied in this study

In this section some possible areas of use of AMS-data for DSOs are listed. This section is not intended to cover all areas of use, but rather some selected areas which may have the potential to be utilized by Norwegian DSOs today.

5.4.1 P1: State Estimation

A future decentralization of power production, as described in Section 3.6.2, may mean a large infeed of power in the low voltage grid. This could cause issues in the way State Estimation (SE) is currently calculated at DSO control centres, and create a need for SE at the lower voltage grid (400/230 V to 22 kV). SE is an analysis function used for electric power systems, which estimates the current state of a power system using redundant measurements [39]. In Norway, SE is computed in the Energy Management System (EMS) of the DSO, with grid state data collected by a Supervisory Control And Data Acquisition (SCADA) system [96]. The classical SE algorithms work well with SCADA-collected data at high (132 kV to 400 kV) and medium (50 kV to 132 kV) voltage levels, but would not work well on AMS-data as they contain new variables and typically a negative measurement redundancy [1, 94]. As electricity production becomes more decentralised and perhaps as transport is electrified, DSOs can benefit from using AMS-data to compute SE for the grid at lower voltage. This has already been successfully demonstrated, and some examples may be found in references [94, 78, 95].

It should be mentioned that the Distribution Management System (DMS) of a Norwegian DSO typically provides the same functionality as an Energy Management System (EMS), but for lower voltage levels. Therefore it may seem plausible that DMS could have the ability to compute SE for the lower voltage grid. However, there is an important difference in the interaction between SCADA and EMS, and SCADA and DMS. Norwegian DSO's SCADA systems typically share their server with the DSO's EMS, but not the DMS. This means that the data collected by the SCADA system will need to pass through a firewall and/or a SCADA "demilitarized zone" to enter the DMS. This would result in a time lag in calculations done by the

DMS, including SE. More on the hierarchy and interaction of Norwegian DSO's EMS, DMS and SCADA systems may be found in Appendix I.

5.4.2 P2: Outage management

AMS-data may be used in outage management for both outage detection/verification, and outage restoration [52]. All Norwegian DSOs have an Outage Management System (OMS) available as a package in their EMS as well as their DMS. Due to the "small" size of most Norwegian DSOs, they generally only have an OMS activated in their DMS. This is because their medium voltage grid (50 kV to 132 kV) is not large or complex enough to necessitate an OMS. The low voltage grid (400/230 V to 22 kV) is however often complex enough to warrant a need for an OMS. However, OMS systems activated in the DMS of Norwegian DSOs only have access to data from the SCADA system [96]. This means that the OMS of today can only do calculations based on data from substations in the low voltage grid.

During a power outage the absence of signals from AMS-meters may be detected and used to map the area effected by the outage. Additionally, AMS-meters store data during an outage which may be analyzed after the incident has been resolved, (§4-2 c)-d)) of Section 4.1.1. The ability of AMS-meters to function as circuit breakers for connected utility customers, and the ability to both detect ground faults and measure flow of active and reactive power (all mentioned in Section 4.1.1, §4-2 e) & h)) potentially enables more advanced outage control than a DSOs OMS system can provide today [7]. There exists many proposed methods for outage management with the use of AMS-meters, some of which may be found in references [7, 52, 50]

5.4.3 P3: Power consumption prediction

Predicting power consumption in the utility grid may have many useful applications for DSOs. As described in Section 3.4, DSOs have the responsibility to buy power in the power markets to cover grid losses and to supply utility grid customers who are not customers of a power company. If a given DSO is able to accurately predict the consumption of utility customers the loss in the utility grid could more accurately be predicted, and a more correct volume of power bought in the day-ahead market. Predicting customer consumption may also have applications for demand response, and for deciding whether or not to use an aggregator on certain occasions.

In the past, linear or logistic regression models have commonly been used to predict customer consumption and grid losses. This is a conclusion from conversations with NVE, TrønderEnergi, Energi Salg Norge and Lyse Elnett realted to this report. Linear and logistic regression models are defined as supervised machine learning models, described in Section 7.2.1, where model parameters are fit to a training data set. However, more accurate results may be obtained with more advanced regression models which are based on logistic regression concepts, such as neural networks. In these newer models, like the one developed by TrønderEnergi Kraft AS in Norway, a supervised learning algorithm is trained with AMS-data to provide predictions which show a 30% (or more) increase in accuracy when compared to the old linear/logistic regression models [6]. Neural networks have become a hot topic of research due to their accuracy and versatility. Therefore, many research papers may be found on neural networks for predicting time series data, some of which are given in references [6, 29].

5.4.4 P4: Customer segmentation

Norwegian DSOs can use AMS-data to study the electricity consumption patterns of their customers. Before installing AMS-meters at every customer, Norwegian DSOs could only collect time series consumption measurements from substations and customers with large power consumption in the low voltage grid. Now, DSOs posses a wealth of time series consumption data which they can analyze.

The goal of a customer segmentation analysis for this report would be to divide the customers into groups of similar consumption patterns. Knowing the approximate behaviour of customers in specific segments can for example be useful for grid developers. If the developers know the typical profiles of customers which will be connected to the grid, they can more accurately predict if the transformers in the substations or transformer stations situated "upstream" of the new customers have the necessary capacity. Customer segmentation can also prove useful for DSOs for implementing tariff systems specifically designed for the defined customer segments. Additionally, knowing the consumption behaviour of customers may represent a basis on which aggregators can implement demand response in the future. One pilot project looking into demand response with AMS-data and aggregators is the Elnett21 project where Lyse Elnett is a partner [33].

5.5 Selected area of AMS-data use to be studied in this report

One of the areas P1 to P4 of AMS-data use should be selected for further study in this report. When evaluating the P1 to P4 study areas, they were assessed in an Analytic Hierarchy Process (AHP). The AHP is a tool used in decision making which aids the decision maker in taking the best decision given a set of options and some evaluation criteria. It reduces complex decisions to a series of pairwise comparisons by setting up an AHP matrix, where each decision option is evaluated against each criteria [76].

Following the AHP method, the decision options P1 to P4 were evaluated according to the following four criteria:

- C1: Feasibility of analyzing P1-P4 based on the available AMS-data.
- C2: Perceived benefit of solution of P1-P4 for DSOs.
- C3: Availability of analysis tools (algorithms).
- C4: My (the author's) knowledge of the topic.

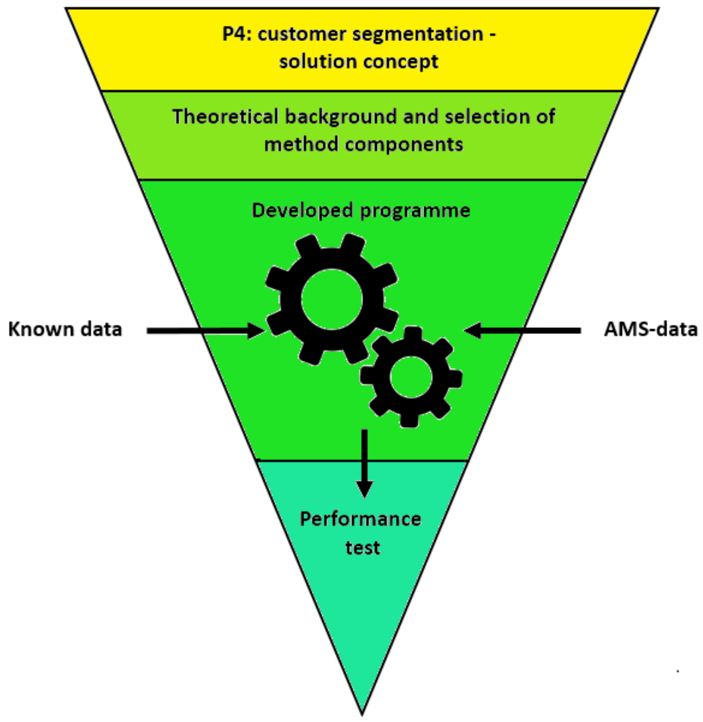
The potential areas of study P1-P4 were scored in an AHP matrix, Table 1, according to criteria C1-C4 on a scale from 1-5, where a higher score indicates a more favourable evaluation. The potential area of study with the highest score was chosen for further research. The results of the evaluation were as follows:

Table 1: Potential areas of study P1-P4 scored against criteria C1-C4.

	P1	**P2**	**P3**	**P4**
C1:	0	0	5	5
C2:	0	2	3	4
C3:	0	3	5	5
C4:	1	2	3	4
SUM:	*1*	*7*	*16*	*18*

From Table 1 it is clear that P4, customer segmentation is preferred and was therefore chosen as the area of AMS-data use to be studied in further detail.

Part II: Method development and assessment

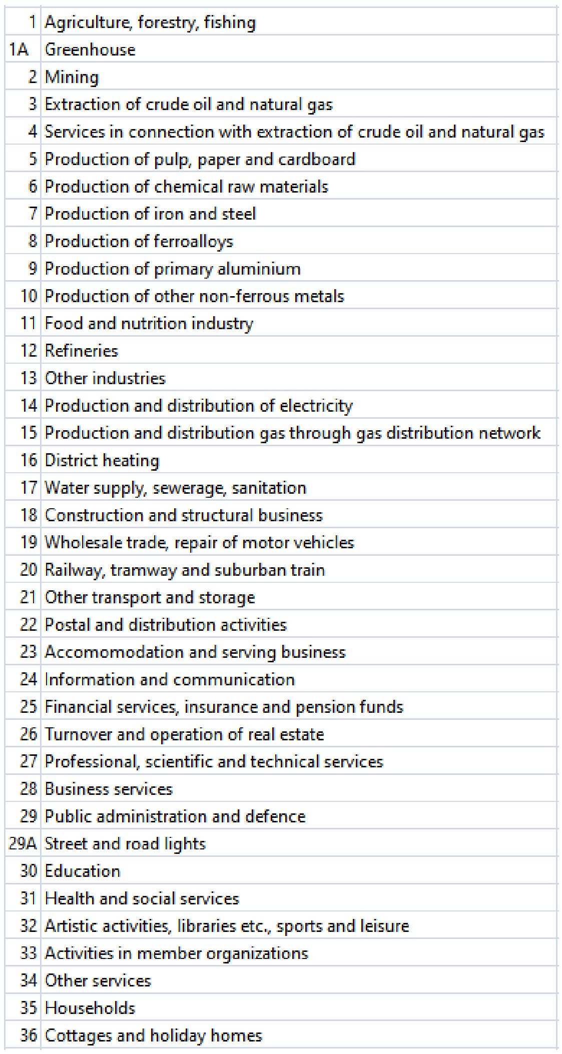

P4: customer segmentation - solution concept
Theoretical background and selection of method components
Developed programme
Known data
AMS-data
Performance test

6 Selecting a method for customer segmentation (P4)

This Section describes how developing a solution to the selected topic of research, P4, customer segmentation, was approached. The solution will comprise of a computer programme, and the reader is in this Section introduced to the coding language Python which is chosen as the language for programme implementation.

6.1 Current method for customer segmentation

Traditionally, utility grid customers have been assigned to customer segments at the time of connection. This means that customers are assigned to segments *before* their consumption behaviour has been observed. The list of different segments to which customers traditionally have been assigned is shown in Figure 11.



1	Agriculture, forestry, fishing
1A	Greenhouse
2	Mining
3	Extraction of crude oil and natural gas
4	Services in connection with extraction of crude oil and natural gas
5	Production of pulp, paper and cardboard
6	Production of chemical raw materials
7	Production of iron and steel
8	Production of ferroalloys
9	Production of primary aluminium
10	Production of other non-ferrous metals
11	Food and nutrition industry
12	Refineries
13	Other industries
14	Production and distribution of electricity
15	Production and distribution gas through gas distribution network
16	District heating
17	Water supply, sewerage, sanitation
18	Construction and structural business
19	Wholesale trade, repair of motor vehicles
20	Railway, tramway and suburban train
21	Other transport and storage
22	Postal and distribution activities
23	Accomomodation and serving business
24	Information and communication
25	Financial services, insurance and pension funds
26	Turnover and operation of real estate
27	Professional, scientific and technical services
28	Business services
29	Public administration and defence
29A	Street and road lights
30	Education
31	Health and social services
32	Artistic activities, libraries etc., sports and leisure
33	Activities in member organizations
34	Other services
35	Households
36	Cottages and holiday homes

Figure 11: The different categories that a utility grid customer connected to the Norwegian low or medium voltage grid can be assigned to.

However, as outlined in Section 3.6, the Norwegian power system is currently undergoing changes. One of which is the introduction of AMS-meters, explained in Section 4. AMS-data open up for the possibility of segmenting a DSO customer base based on the measured electricity consumption behaviour of utility grid customers. In other words, assigning customers to segments *after* their consumption behaviour has been observed.

6.2 Approach to solution development

As an approach to developing a method for customer segmentation based on the use of AMS-data, the problem was divided into sub-problems as shown in Figure 12.

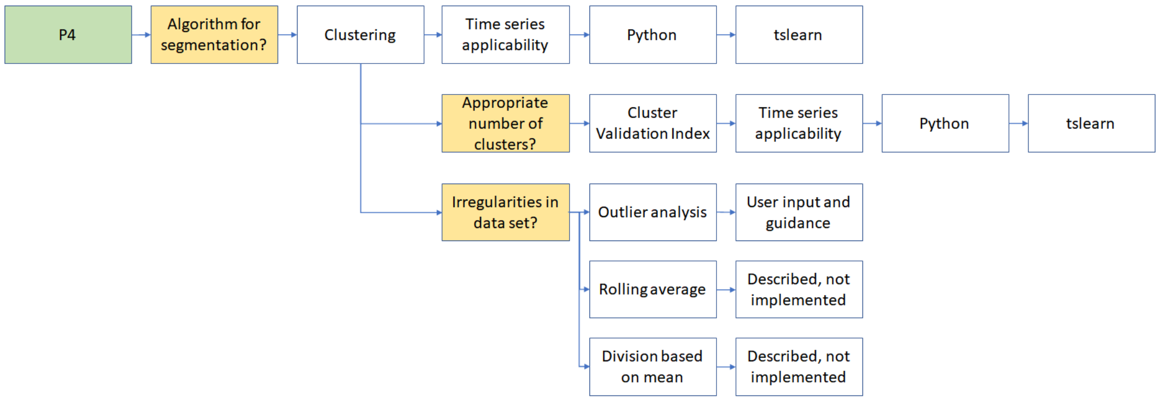

P4
Algorithm for segmentation?
Clustering
Time series applicability
Python
tslearn
Appropriate number of clusters?
Cluster Validation Index
Time series applicability
Python
tslearn
Irregularities in data set?
Outlier analysis
User input and guidance
Rolling average
Described, not implemented
Division based on mean
Described, not implemented

Figure 12: The approach to selecting a method for P4, customer segmentation.

Firstly, the feasibility of actually developing a customer segmentation method was assessed. It was questioned if there already existed accessible algorithms which could perform segmentation of time series data. As a starting point I (the author) attended an online Machine Learning course from Stanford's Coursera.

Developing a method would be feasible if there existed machine learning libraries in accessible coding languages specifically designed for time series data. After some research the `tslearn` library for Python was discovered. `tslearn` is a machine learning library specifically written for time series, and is freely accessible to the public.

Secondly, based on the available Python libraries, the development of a customer segmentation computer programme was evaluated. After attending the Machine Learning course it became clear that the algorithm type best fit for this task was clustering (more on clustering in Section 7.2.2 and Section 7.3). After clustering was selected as the algorithm type of choice, it became apparent that an additional type of algorithm should be implemented in order establish an appropriate numbers of clusters for a given time series data set. A Cluster Validation Index was chosen for this purpose, further explained in Section 7.4.

Lastly, although not considered a main focus of this report, functionality to identify irregularities in the data set was added. Refining the data set based on removing outliers, rolling averages and dividing the data set into subsets based on mean values was evaluated. Outlier analysis was the functionality implemented in the programme code. Outlier analysis is based on guidance and input from the user. Potential implementation of rolling averages and division based on mean values are however described in Section 7.5.4.

6.2.1 Aspirational properties of the developed customer segmentation method

The customer segmentation method would aspire to have the following properties:

- Be computer-based and developed as part of the work related to this report.
- Be scalable to handle large amounts of data.
- Be freely accessible to anyone who wishes to use it for their own purposes. This would mean that it should be based on a freely accessible software such as Python.
- Be modular, which means that a user may easily apply additional functionality to the solution, or change the existing algorithms without necessarily changing the *structure* of the programme.
- Be able to demonstrate feasibility and promising results.
- Be described in this report in an accessible manner to the reader.

6.3 Using Python

As described above, Python was the programming language selected for the customer segmentation programme. This Section gives an introduction to installation of Python, Anaconda (Python interpreter), as well as the Python libraries relevant for the report.

Python is a high level programming language. Anaconda is an open-source distribution of the Python programming language for scientific computing, with a simple and efficient package management system. To use the code in this report it is necessary to install Python, and preferable to install Anaconda.

6.3.1 Installing Anaconda and Python

Start by downloading the newest version of Python here (https://www.python.org/), followed by the newest version of Anaconda here (https://www.anaconda.com/).

The Anaconda package should include the an program called "Spyder". Check that you have Spyder installed by searching for it amongst your installed programs.

Figure 13 shows the logos of the three programs that are used in this report, which should be installed.

Figure 13: The logos of Python, Spyder and Anaconda.

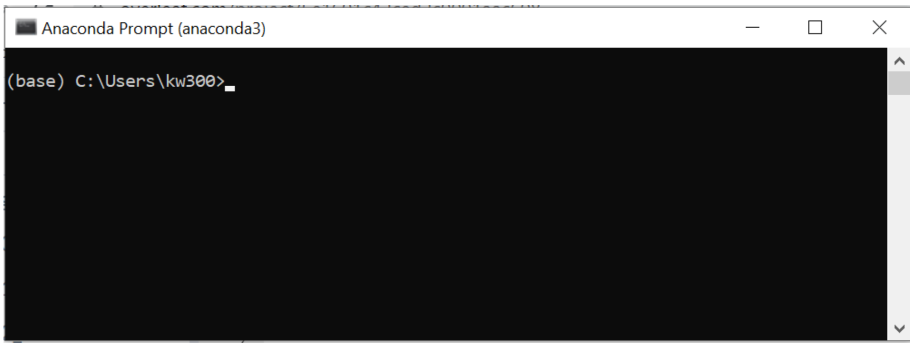
6.3.2 Python libraries

A Python library is a collection of functions and methods which allows a programmer to perform actions and computations without having to write the code for the function/operation from the bottom up. Using libraries in Python is highly time cost-efficient and is done to a large extent in this report. The central libraries in this report are `numpy`, `tslearn`, `pandas` and `matplotlib`. Anaconda has `numpy` and `matplotlib` already pre-installed, so only the `tslearn` and `pandas` libraries need to be installed separately.

Installing `pandas`

`pandas` is a software library for the Python language which is particularly well suited for manipulation of data structures and time series data.

To install `pandas` open the Anaconda Prompt. This is a command prompt window for Anaconda which looks like a black window, as shown in Figure 14. The Anaconda Prompt may be found by searching for "Anaconda Prompt" in your installed programmes.



```
(base) C:\Users\kw300>_
```

Figure 14: The Anaconda Prompt window.

Enter the following command into the Anaconda Prompt window, and press enter:

```
pip install pandas
```

Installing `tslearn`

`tslearn` is a machine learning software library for the Python language which is specifically developed for time series type data.

To install `tslearn`, open the Anaconda Prompt window as described in Section 6.3.2 and shown in Figure 14. Enter the following command into the Anaconda Prompt and press enter:

```
conda install -c conda-forge tslearn
```

6.3.3 Text files and Python files

Text files are computer files that only contain text, with no special formatting such as bold text, italic text, images etc. On computers operated by a Windows system, text files are identified by the ".txt" file extension. Since all code in this report is written on a Windows-based computer, text files are referred to as .txt files.

A Python file is a script file written in the Python programming language. Python files are identified by the ".py" file extension. These files can only be run in a Python interpreter, which is in this report chosen to be Spyder.

7 Theoretical background and description of methods for the customer segmentation programme

In order to understand the customer segmentation method explored in this report it is useful to look into the theoretical background of its different components.

Three types of algorithms represent the building blocks of the programme as explained in Section 6.2. These are; distance metrics, clustering algorithms and Cluster Validation Indexes (CVI). Each of these building blocks are explained in this section. One specific distance metric, clustering algorithm and CVI is selected as the algorithm of choice for the customer segmentation programme, as the report is not primarily concerned with exploring the subtle differences and advantages of various distance metrics, clustering algorithms and CVIs. However, the customer segmentation programme is modular, which means that the selected distance metrics, clustering algorithm and CVI may be changed by a future user.

7.1 Distance measures relevant for the selected algorithms

In this Section the distance metrics which are used in the algorithms and methods central to this report are described. Since the report is concerned with time series data, the distance metrics described are especially adapted to these types of data.

The relevant distance metrics for this report are:

- **Euclidean distance:** Is not well suited for calculating distance between time series, but is included in this report because it is a often a component of other distance metrics which are useful for calculating time series distance. One such metric is Dynamic Time Warping.
- **Dynamic Time Warping:** Although first designed for speech recognition in 1978, Dynamic Time Warping (DTW) has since become one of the most used time series similarity metrics because it is able to capture time series similarity under time distortions [77]. This means that two time series do not have to be perfectly aligned in time for DTW to measure their similarity, which makes DTW far more powerful than for example pure Euclidean distance that does not take time distortion into account [98]. DTW is in this report used for data refinement, and in the selected CVI.
- **Shape-based distance:** This distance metric is used in the calculation of clusters in the clustering algorithm, K-Shape, which is central to this report.

These three distance metrics are explained in further detail in the following Subsections.

7.1.1 Euclidean distance

Euclidean distance is the mathematical formulation of a “straight line” between two points. The Euclidean distance between points x and y is the same as the length of the line segment connecting them. This distance may be defined by Equation (1)

[26].

$$d_{x,y} = \sqrt{(x-y)^2} \tag{1}$$

In Cartesian coordinates, given the set of points on a two-dimensional plane $\boldsymbol{x} = (x_1, x_2, ..., x_n)$ and $\boldsymbol{y} = (y_1, y_2, ..., y_n)$, the Euclidean distance between $\boldsymbol{x}$ and $\boldsymbol{y}$ is given by the Pythagorean formula, Equation (2) [26].

$$d_{x,y} = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{2}$$

7.1.2 Dynamic Time Warping

Given two normalized time series $\boldsymbol{x}$ and $\boldsymbol{y}$ where:

$$\boldsymbol{x} = (x_0, x_1, ..., x_{n-1}), n \in \mathbb{N}$$
$$\boldsymbol{y} = (y_0, y_1, ..., y_{m-1}), m \in \mathbb{N}$$

the aim of DTW is to calculate an optimal alignment between two time series $\boldsymbol{x}$ and $\boldsymbol{y}$ which achieves minimum "global cost" while ensuring time continuity. The global cost is the summation of the cost between each point x_i and y_j in the alignment. The cost function, denoted $C(x_i, j_j)$ is typically calculated by squared Euclidian distance, Equation (1), denoted d [98], as shown in Equation (3).

$$C(x_i, j_j) = (x_i - y_j)^2 = d(x_i, y_j)^2 \tag{3}$$

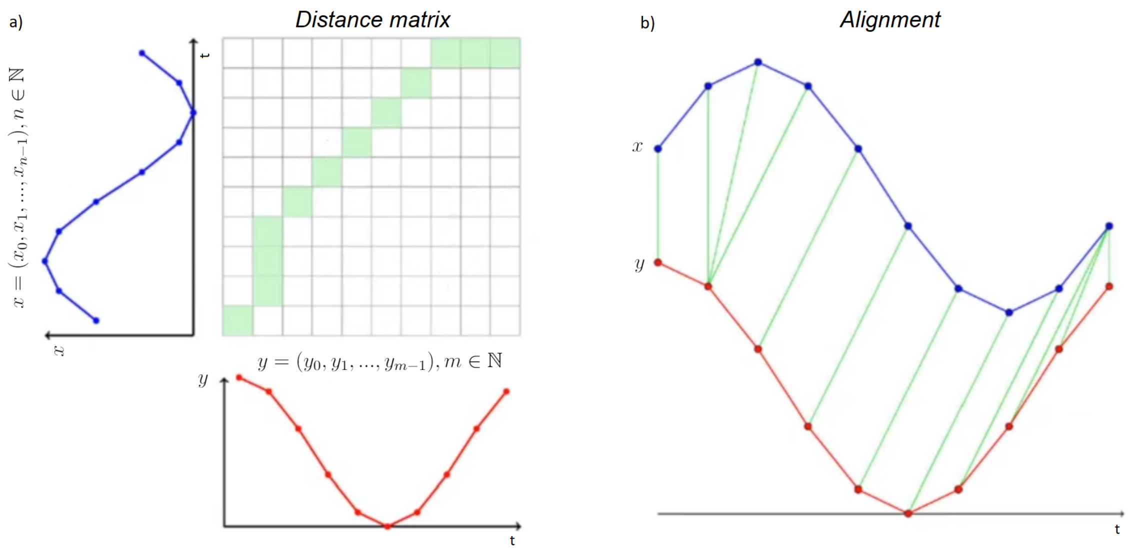

a)
Distance matrix
$x = (x_0, x_1, ..., x_{n-1}), n \in \mathbb{N}$
t
x
$y = (y_0, y_1, ..., y_{m-1}), m \in \mathbb{N}$
y
t
b)
Alignment
x
y
t

Figure 15: a) Dynamic Time Warping distance matrix illustration, with b) the calculated optimal alignment illustrated.

The DTW calculation method may be illustrated as shown in Figure 15 a), where time series $\boldsymbol{x}$ and $\boldsymbol{y}$ are aligned along the sides of an nxm matrix called the *distance* *matrix*. Each element in the distance matrix will contain the distance cost between each corresponding point x_i and y_j of the time series. The *optimal alignment path* which is the "path" through the distance matrix with the lowest cost is marked in green in Figure 15 b). The optimal alignment path can be thought of as a temporal alignment for $\boldsymbol{x}$ and $\boldsymbol{y}$ such that the Euclidian distance between $\boldsymbol{x}$ and $\boldsymbol{y}$ is minimal [98].

Mathematical formulation

DTW is formulated as the following optimization problem:

$$DTW(\boldsymbol{x}, \boldsymbol{y}) = \min_{\pi} \sqrt{\sum_{(i,j)\in\pi} d(x_i, y_j)^2} \tag{4}$$

Where $d(x_i, y_j)^2$ is the distance between the two points x_i and y_j, and $\pi = [\pi_0, \pi_1, ..., \pi_K]$ is the optimal alignment path as illustrated in Figure 15 that satisfies the following properties [88]:

- $\pi = [\pi_0, \pi_1, ..., \pi_K]$ is a list of index pairs, $\pi_k = (i_k, j_k)$, with $0 \leq i_k < n$ and $0 \leq j_k < m$
- $\pi_0 = (0, 0)$ and $\pi_K = (n-1, m-1)$
- for all $k > 0$, $\pi_k = (i_k, j_k)$ is related to $\pi_{k-1} = (i_{k-1}, j_{k-1})$ as follows:
  - ☐ $i_{k-1} \leq i_k \leq i_{k-1} + 1$
  - ☐ $j_{k-1} \leq j_k \leq j_{k-1} + 1$

The optimal global cost of DTW, i.e. the DTW distance is calculated recursively by:

$$D(x_i, y_j) = C(x_i, y_j) + \min \begin{cases} D(x_i, y_{j-1}) \\ D(x_{i-1}, y_j) \\ D(x_{i-1}, y_{j-1}) \end{cases} \tag{5}$$

Where $D(i, j)$ is an element of the cumulative cost matrix with index $[i, j]$ (nxm distance matrix) [98]. $C(x_i, y_j)$ is the cost function as defined in Equation (3).

7.1.3 Shape-based distance

Shape-based distance is the distance metric which the clustering algorithm central in this report uses. Firstly, the term *shape-based* should be defined. An algorithm may be thought of as shape-based when it is amplitude, offset and shift/time invariant. This means that the algorithm should recognize the similarity of two given time series irrespective of amplitude, offset and time shift. Hence, the shape-based distance metric described in this section and used in shape-based clustering is in theory amplitude, offset and time shift invariant [68].

Amplitude and offset invariance

Amplitude and offset invariance is achieved by normalizing the data set. This means each time series is recomputed such that its standard deviation σ' is equal to one, and its mean μ' is equal to zero, i.e. each sequence $\boldsymbol{x}$ is transformed into $\boldsymbol{x}' = \frac{\boldsymbol{x}-\mu}{\sigma}$ where μ is the mean and σ is the standard deviation of $\boldsymbol{x}$ [68].

Time shift invariance

Time shift invariance is computed with a cross-correlation measure, which is a statistical measure that determines the similarity of two sequences $\boldsymbol{x} = (x_1, x_2, ..., x_n)$ and $\boldsymbol{y} = (y_1, y_2, ..., y_n)$ even if they are not properly aligned. Cross-correlation is computed by sliding $\boldsymbol{x}$ over $\boldsymbol{y}$ and computing the inner product of each shift s of $\boldsymbol{x}$. When all possible shifts s are considered, $s \in [-n, n]$, the cross-correlation sequence $CC_w(\boldsymbol{x}, \boldsymbol{y}) = R_{w-m}(\boldsymbol{x}, \boldsymbol{y})$, $w \in \{1, 2, .., 2n-1\}$ is computed with Equation (6) [68].

$$CC_w(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} \sum_{l=1}^{n-k} x_{l+k} \cdot y_l & k \geq 0 \\ R_{-k}(\boldsymbol{x}, \boldsymbol{y}) & k < 0 \end{cases} \tag{6}$$

The goal is to compute the position w at which $CC_w(\boldsymbol{x}, \boldsymbol{y})$ is maximised. The optimal shift will then be given by $s = w - n$. CC_w is then normalized, to produce NCC_c in Equation (7) [68].

$$NCC_c(\boldsymbol{x}, \boldsymbol{y}) = \frac{CC_w(\boldsymbol{x}, \boldsymbol{y})}{\sqrt{R_0(\boldsymbol{x}, \boldsymbol{x}) \cdot R_0(\boldsymbol{y}, \boldsymbol{y})}} \tag{7}$$

Shape-based distance calculation

After sequence $\boldsymbol{x} = (x_1, x_2, ..., x_n)$ and $\boldsymbol{y} = (y_1, y_2, ..., y_n)$ are transformed to enable amplitude and offset invariance, the position w where $NCC_c(\boldsymbol{x}', \boldsymbol{y}')$ is maximised is computed. The shape-based distance SBD between sequences $\boldsymbol{x}'$ and $\boldsymbol{y}'$ is then given by Equation (8) [68].

$$SBD(\boldsymbol{x}', \boldsymbol{y}') = 1 - \max_{w} \{NCC_c(\boldsymbol{x}', \boldsymbol{y}')\} \tag{8}$$

SBD will take a value between 0 and 2, where 0 indicates a perfect similarity between sequences $\boldsymbol{x}'$ and $\boldsymbol{y}'$.

7.2 Introduction to machine learning

Machine learning may be thought of as the study of computer algorithms that automatically improve with experience. The term was originally coined in 1959 by Arthur Samuel, who at IBM developed an algorithm for the board game "checkers" which improved its performance with practice [48]. In 1997 Tom Mitchell [18, 59] defined "machine learning" as:

> "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

This definition is today through common consensus considered a formal definition of machine learning in academia [59]. The "computer program" for machine learning in the definition does its computation with an *algorithm*.

Machine learning problems are generally assigned to one of two broad categories; supervised or unsupervised learning problems.

7.2.1 Supervised machine learning

In supervised machine learning the algorithm is given a data set to practice on where the correct output for the data set is already known. This data set is termed a training set. Algorithm parameters are then fit to the training set, typically through an optimization method like gradient descent [59]. This process is illustrated in Figure 16.

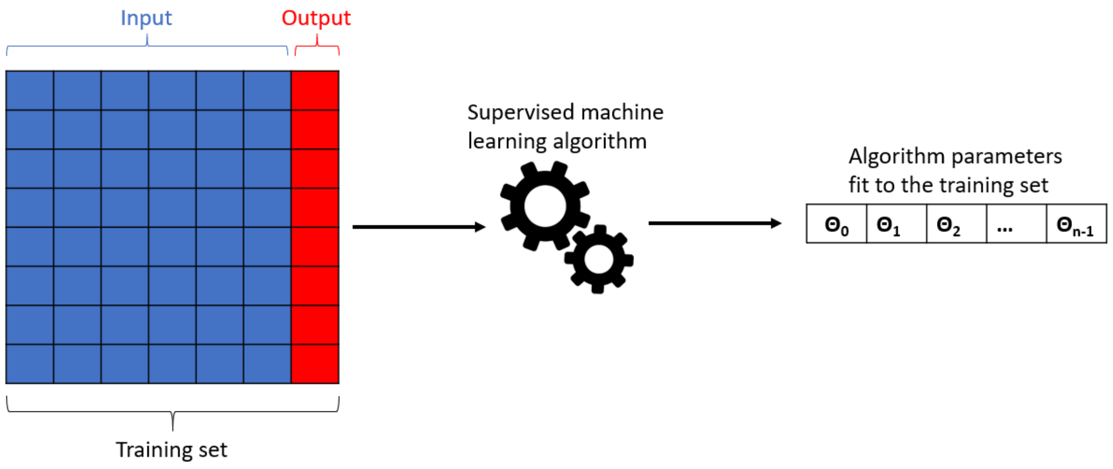

Input
Output
Training set
Supervised machine learning algorithm
Algorithm parameters fit to the training set
Θ_0
Θ_1
Θ_2
...
Θ_{n-1}

Figure 16: Supervised machine learning algorithm fitting parameters to a training set.

When the algorithm parameters have been fit to the training set, the algorithm may be given another data set where the relationship between the input and output of the data in the new data set is assumed to resemble the relationship between the input and output of the data in the training set. This way, if the parameters of the supervised machine learning algorithm are fit well, the algorithm may be able to predict the correct output of the data in the new data set [59]. This is illustrated in Figure 17.

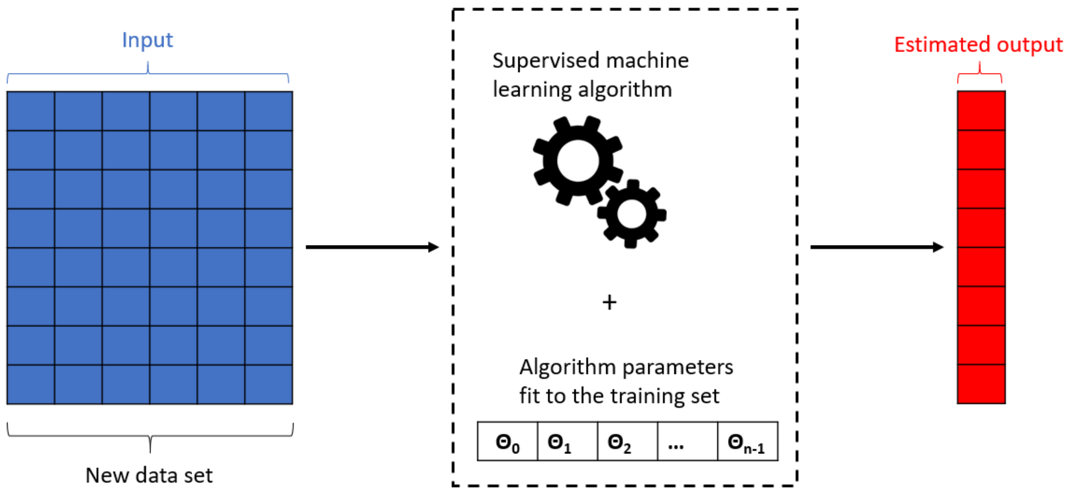

Input
New data set
Supervised machine learning algorithm
+
Algorithm parameters fit to the training set
Θ_0
Θ_1
Θ_2
...
Θ_{n-1}
Estimated output

Figure 17: Supervised machine learning algorithm predicting the output for a new data set with the parameters fit to a training set.

Supervised machine learning problems may be further categorized into either regression or classification problems [59] (not developed further in this report).

7.2.2 Unsupervised machine learning

In unsupervised machine learning the algorithm will attempt to derive structure from data where it is not know beforehand what the output should look like [59]. This is illustrated in Figure 18 where a set of input data is fed into the machine learning algorithm, and the output data depends on the type of algorithm used. Figure 18 may give the impression that an unsupervised machine learning algorithm always outputs a data set of the same size as the input data set, which may not be the case. Output data set size may vary depending on the algorithm.

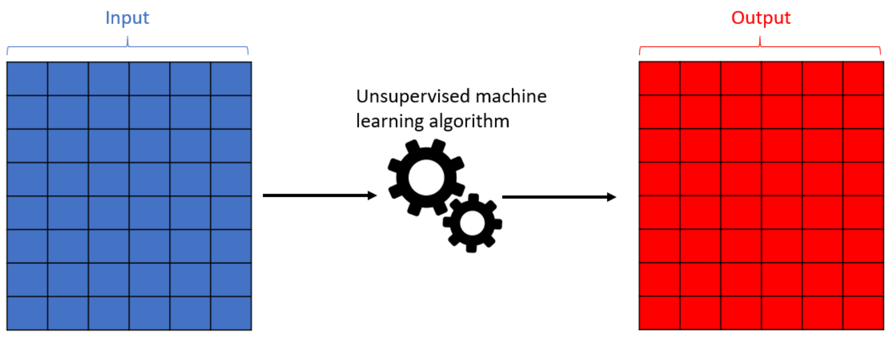


Figure 18: Illustration of unsupervised machine learning

Unsupervised machine learning problems are sometimes classified into clustering and non-clustering type problems [59].

Clustering algorithms derive structure in a data set based on relationships among the variables in the data. This machine learning technique is central to this report, and is explained in further detail in Section 7.3. Figure 19 gives a simple illustration of the result of a clustering algorithm which has been instructed to find two clusters in a data set of variables with attributes X and Y.

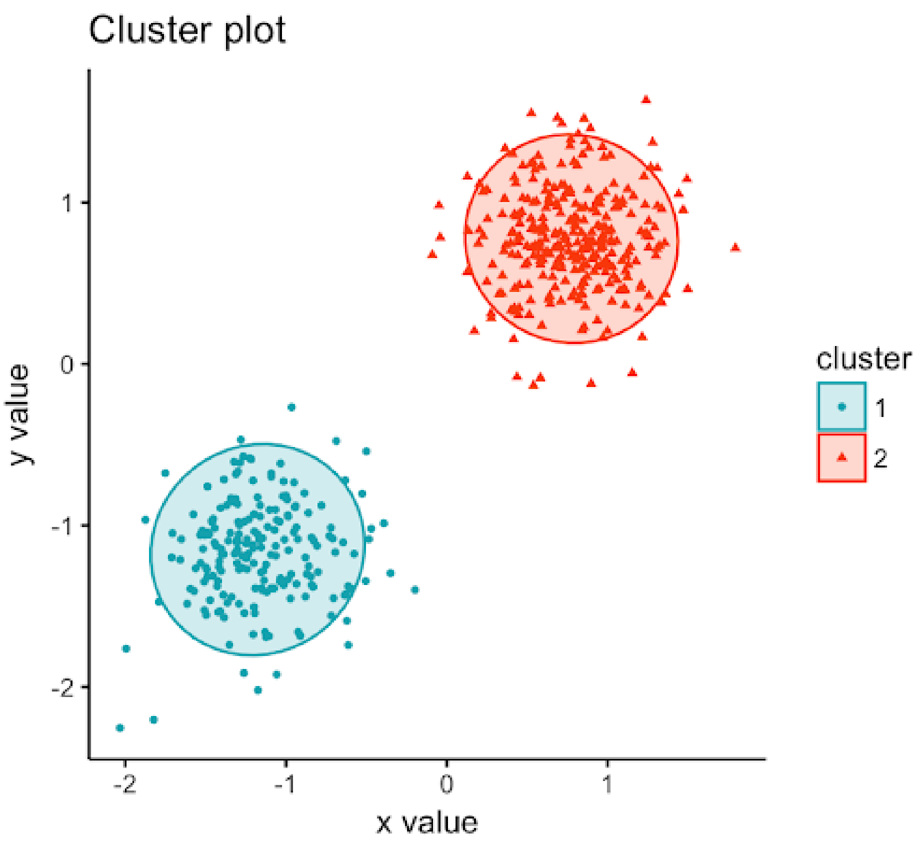


Figure 19: The result of a clustering algorithm [40].

Non-clustering algorithms are another type of algorithms which may find structure in "messy" data, and may be used in for example speech recognition.

7.3 Clustering

Clustering may be thought of as the task of grouping a set of objects such that objects in the groups are more "similar" to each other than to objects in other groups. As the notion of what constitutes a desirable "cluster" and what does not varies according to both the aims of the algorithm user and to the type of data which is to be clustered, there exists a myriad of different clustering algorithms [91].

In this report, a centroid- and shape-based clustering approach was selected. Centroid-based because then each time series cluster can be represented by a single average time series (the centroid), which is both visually illustrative and informative. Shape-based because it was considered desirable to be able to capture similarities in customer consumption profiles despite time shifts and amplitude/offset variations. This means the clustering algorithm should for example be able to recognize the similarity of two household's electrical consumption pattern even if they have different consumption levels, and even if one household leaves the house to go to work/school at 07:00 AM in the morning, and the other one at 09:00 AM. The centroid- and the shape-based clustering approach is illustrated in in Figure 20, where the red line is the cluster centroid, and the clusters are formed despite of time shifts, amplitude and offset variations (amplitude/offset variations are not shown in the pictures as the time series have been normalised). It was also important to select a clustering algorithm which could handle large data sets, as Norwegian DSOs usually have at least a couple tens of thousands of customers.

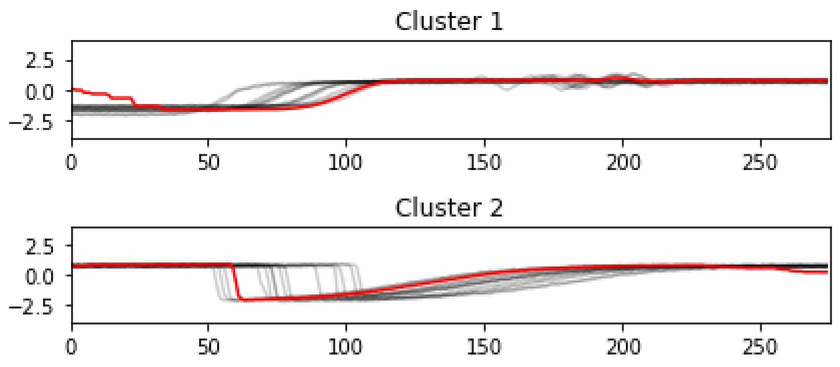


Figure 20: Cluster- and shape-based time series clustering [88].

To that end the centroid- and shape-based clustering algorithm called "K-Shape" was selected as the algorithm of choice in this report. The K-Shape algorithm was also favourable with respect to data size scalability as the algorithm run time scales linearly to the number of time series [68]. `tslearn` does include many clustering algorithms besides K-Shape for time series clustering, so the `tslearn KShape` clustering algorithm in the Python programme given in Appendix III and IV may therefore easily be changed for another clustering algorithm from `tslearn`. All available time series clustering algorithms in `tslearn` may be found in reference [88].

The reader is discouraged from using clustering algorithms not specifically designed for clustering time series data, such as the algorithms found in Python's popular `sklearn` library. This is because they do not use distance metrics suitable for comparing time series.

7.3.1 K-Shape

K-Shape is a centroid- and shape-based clustering algorithm applicable to time series data. The shape-based feature is achieved by using shape-based distance as the algorithm's distance metric. Shape-based distance is, as explained in Section 7.1.3, both amplitude, offset and shift invariant [68].

The clusters are computed by minimizing an objective function, which is the sum of squared distances between each time series to its respective cluster centroid. Given

K clusters this can be mathematically represented by Equation (9) **??**.

$$P^* = min \sum_{j=1}^{K} \sum_{\boldsymbol{x}_i \in p_j} dist(\boldsymbol{x}_i, \boldsymbol{c}_j)^2 \tag{9}$$

where p_j is the partition of the total data set which includes all time series in cluster j, $\boldsymbol{c_j}$ is the centroid of cluster j and $dist$ is the Shape-based distance metric as used in Section 7.1.3.

The centroid of a cluster may be though of as the multidimensional average of a cluster, and is in the K-Shape algorithm computed iteratively until convergence. The number of clusters that the K-Shape algorithm computes must be given as input to the algorithm. Throughout this report, the number of clusters that the `tslearn KShape` algorithm is instructed to compute is denoted `K`. The cluster centroid computation process occurs as follows [68]:

- All time series in the data set get randomly assigned to a cluster, and the `K` number of cluster centroids are computed with Equation (10).

  Then, an iterative process starts where in each iteration the following steps are executed:

  - ☐ For each time series in the data set, compute its similarity with the use of shape-based distance as described in Section 7.1.3 to all `K` centroids, and place it in the cluster with the most similar centroid.
  - ☐ Recalculate the cluster centroids with Equation (10).

The above steps are repeated until no time series are moved from one cluster to another, or when a pre-specified number of iterations are performed [68].

Centroid computation method

The cluster centroids in the K-Shape algorithm are computed using Equation (10) [68].

$$\boldsymbol{\mu}_k^* = \sum_{\boldsymbol{x}_i \in P_k} NCC_C(\boldsymbol{x}_i, \boldsymbol{\mu}_k^*)^2 \tag{10}$$

7.3.2 `tslearn KShape` algorithm result stability

As described in Section 7.3.1, the K-Shape algorithm will repeat the cluster centroid recalculation and time series reassignment steps until either a pre-specified number of iterations are performed, or until no time series are changed from one cluster to another. In the `tslearn KShape` algorithm used in this report, this pre-specified number of iterations is given by the input variable `max_iter` [88].

The `tslearn KShape` algorithm also includes the input variable `n_init`, which determines the number of times that the `KShape` algorithm will be run with different centroid seeds, i.e. the initial random placement of clusters. The result is then the best output of `n_init` consecutive runs of the `KShape` algorithm, in terms of inertia [88].

In this report, the `tslearn KShape` algorithm is considered to be in a "stable zone" when, for a constant `K`, its produces consistent results (i.e. a stable CVI), for `max_iter` or `n_init` above a certain value. It will be shown in Section 9 that the stability of the `KShape` algorithm is influenced by the value of `n_init`, not `max_iter`. Therefore, the formal definition of `KShape` algorithm stability used in this report is when, for a constant `K`, the `KShape` algorithm produces consistent reults (i.e. a stable CVI) for a threshold value of `n_init` and above.

For large data sets there may exist very many possible centroid seeds. For this reason, large data sets may need a large value for `n_init` to reach a stable zone for the `KShape` algorithm, as many different sentroids seeds can be tested.

7.3.3 Cluster usefulness

The usefulness of a cluster partition will vary according to the type of data that is clustered, and according to the aims of the user. However, there exists some characteristics which may be beneficial as guidelines for assessing cluster partition usefulness [91]. These are:

- **Compactness:** The clusters should be homogeneous within partitions, and heterogeneous between partitions.
- **Differentiable:** The cluster partitions should be conceptually distinguishable.
- **Substantial:** The clusters should be large enough so that the algorithm user is able to gain some use or knowledge from it.
- **Stable:** The cluster partitions should be stable over time.

Compactness can be assessed with the use of CVIs, and substantiality is managed through outlier analysis in Section 7.5 or through the choice of `K`. Cluster differentiability and stability will, however, need to be assessed by the user of the customer segmentation programme developed for this report. This may be done by visually inspecting the clusters and centroids, and by testing the AMS-data over different time intervals. Stability over time of the K-Shape algorithm is demonstrated in section 9.2.5.

7.4 Cluster Validation Index

A Cluster Validation Index (CVI) aims to validate how good a given cluster partition is. In this report a CVI will be used to evaluate the partition compactness for a given number of clusters `K`, with the goal of assisting the identification of an optimal number of partitions, `K_optimal`, for a given time series data set.

There exists no one perfect CVI, so ideally multiple CVIs should be evaluated to determine the optimal partition [91, 5]. However, since this report deals with time series not many attractive options for CVIs exist. Most CVIs which are robust and accessible for Python may be found in the `sklearn` machine learning library. Unfortunately `sklearn` does not work very well with time series since distance metrics are not adapted for time series and is commonly calculated with Euclidean distance, which is not time invariant. `tslearn` builds on `sklearn`, and is particularly constructed to deal with time series. However, `tslearn` only includes one CVI, `silhouette_score` which uses DTW as its distance metric. Fortunately the Silhouette Index is regarded as a highly accurate CVI which can in many cases achieve satisfactory results on its own [12, 11, 75].

The author and developer of the `tslearn` library, Romain Tavenard, was contacted in the context of this report concerning CVIs for time series. He confirmed that `sklearn` CVIs are ineffective when used on time series data. `tslearn` would not immediately be developing more CVIs adapted to time series, but would be happy to receive contributions regarding CVIs to add to the `tslearn` library in addition to the `silhouette_score` function which already exists. These statements indicate that further CVIs may be developed at a later date. Since the customer segmentation programme is modular it will be possible to include other CVIs for time series once they become available in `tslearn` or any other Python library.

7.4.1 Silhouette analysis

Silhouette analysis is the computation of a measure which describes how similar an object is to its own cluster, compared to other clusters [75].

Suppose a data set DS contains object i where $i \in [1, 2, ..., N]$. Suppose object i belongs to cluster group A, after DS which contains i has been run through a clustering algorithm [75].

When cluster A contains other objects apart from i, it is possible to calculate the average dissimilarity of object i to all other objects in A. Dissimilarity is computed with a distance measure d, most commonly Euclidean distance, Equation (1) [75]. The Silhouette algorithm from `tslearn` used in this report uses DTW as a distance measure, not Euclidean distance due to its weaknesses when calculating distance between time series as discussed in Section 7.1.

The following steps are applied to a clustered data set for Silhouette analysis. First calculate:

$a(i)$ = average dissimilarity (distance) of i to all other objects of A

$$a(i) = \frac{1}{\mid A(i) \mid -1} \sum_{A(i), i \neq j} d(i, j) \tag{11}$$

where $\mid A(i) \mid$ is the number of data points in the cluster A assigned to the ith data point. $a(i)$ can be visually represented as the average distance of all lines within A in Figure 21. Every black point in Figure 21 represents an object in the data set DS [75].

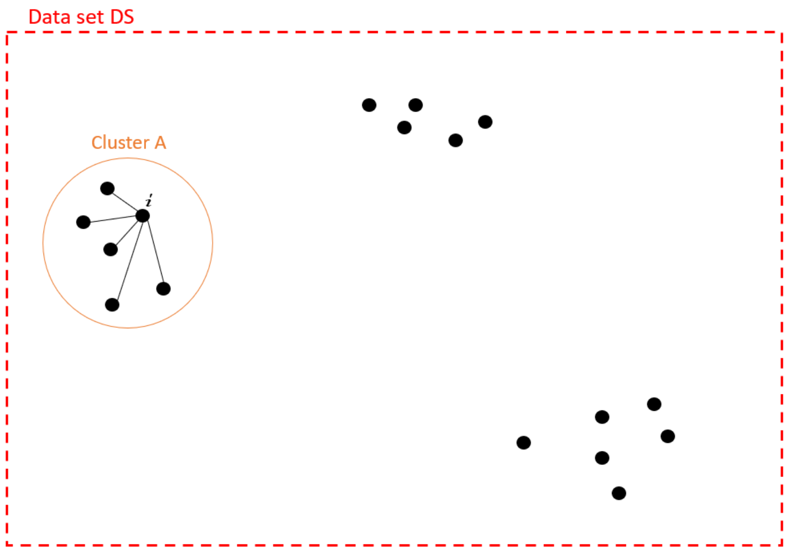


Figure 21: Silhouette analysis first step: calculate dissimilarity between object i in cluster A to all other objects in cluster A, as derived from [75].

Now consider any cluster different from A, denoted cluster C, and compute:

$d(i,c)$ = average dissimilarity of i to all objects of C

$$b(i) = \frac{1}{\mid C(j) \mid} \sum_{j \in C(j)} d(i,j) \tag{12}$$

where $\mid C(j) \mid$ is the number of data points in the cluster C. $d(i,c)$ can be visually represented as the average distance of all lines between i and the objects in cluster C in Figure 22.

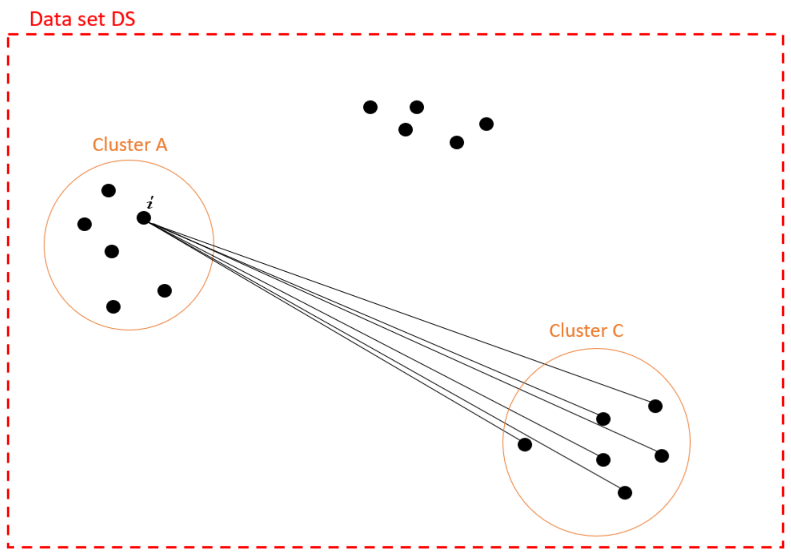


Figure 22: Silhouette analysis second step: calculate distance (dissimilarity) between object i in cluster A and all objects in cluster $C \neq A$, as derived from [75].

After computing the average distance d between i and all objects in clusters $C \neq A$, the smallest distance $b(i)$ is selected [75]:

$b(i) = \min_{C \neq A} d(i, C)$

Denoting the cluster for which this is attained cluster B (i.e. $d(i, B) = b(i)$), the *Silhouette coefficient* $s(i)$ of object i can be defined as [75]:

$$s(i) = \begin{cases} 1 - \frac{a(i)}{b(i)} & \text{if} \quad a(i) < b(i) \\ 0 & \text{if} \quad a(i) = b(i) \\ \frac{b(i)}{a(i)} - 1 & \text{if} \quad a(i) > b(i) \end{cases} \tag{13}$$

or

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{14}$$

It may be seen from Equation (13) and Equation (14) that $-1 \leq s(i) \leq 1$.

The higher the value of $s(i)$ is, the better the assessment for object i to the given cluster is. The average of $s(i)$ over $[1, 2, ..., N]$ is used as a basis for evaluating the quality of a given partition (clustering) of DS. The best clustering is achieved when the average $s(i)$ is maximised [75].

7.5 Refine data set

In this Section, some methods for data set refinement are proposed. Of the proposed methods, outlier analysis is implemented in the developed customer segmentation programme, as described in Section 8.

7.5.1 Outlier analysis

Outliers are broadly considered as observations which do not conform to expected behaviour [9]. A more formal definition of time series outliers was given by M. Hawkins [30] in 1980:

> "An observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism."

However, the concept of what constitutes an outlier and what does not is context-dependent. The user of the programme will have to decide what he or she deems as an outlier in the specific data set that is being analyzed.

Outlier identification and analysis can be helpful in improving the Silhouette score in sets of time series if the set of time series displays overall low cluster compactness. This effect may be observed in Section 9.2.6. Additionally, outliers may be important to detect because of results in Section 9.2.6 demonstrating that the output of the K-Shape and Silhouette algorithms may become unpredictable if enough randomness exists in the data set. Furthermore, outlier analysis may help improve the substantiality of clusters, which is a condition for cluster usefulness as described in Section 7.3.3.

Outlier analysis has to be used with caution. There is a fine line between removing outliers that disturb the algorithms, and removing valuable data from the data set.

An outlier can be either "unwanted data" or an "event of interest". In case events of interest are captured by the outlier analysis, they may be interesting to analyse separately. This is illustrated in Figure 23.

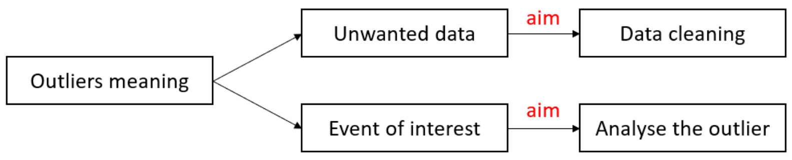


Figure 23: Meaning of outliers in time series data depending on the aim of the analyst, as adapted from [9].

7.5.2 Choice of outlier analysis methods in this report

The customer segmentation model developed in this report is modular, and a module for outlier analysis is included. However, outlier analysis is not considered a main element of the report. Therefore, only simple outlier analysis methods are included. These methods are based on DTW, and on the user of the programme visually inspecting the data set and the clustering results. Additionally, analysis performed for this report have shown that in instances of large amounts of random data ($\approx$ 10 % of the data set), or varying sizes in cluster groups, the method developed is still able to deliver adequate results even without outlier analysis is not performed (Section 9.2.6). In any case, outlier analysis is included in the customer segmentation programme as an option, not a requirement.

The following method is proposed: Separate the data set into inliers and outliers and analyse these separately. The inlier data set will contain the majority of data, and the outlier data set will contain the data which is assumed to impact the partition quality of the inlier data set negatively. This way, no data is deleted in the programme, but sorted into a separate category.

Three methods for detecting and removing outliers from a time series data set are proposed. All thee methods are flexible and require input from a user. The methods are: Removing sparse/undesirable clusters (OD1), removing outliers based on extra-cluster dissimilarity (OD2), and removing outliers based on intra-cluster dissimilarity (OD3). The following Subsections explain these three methods in turn.

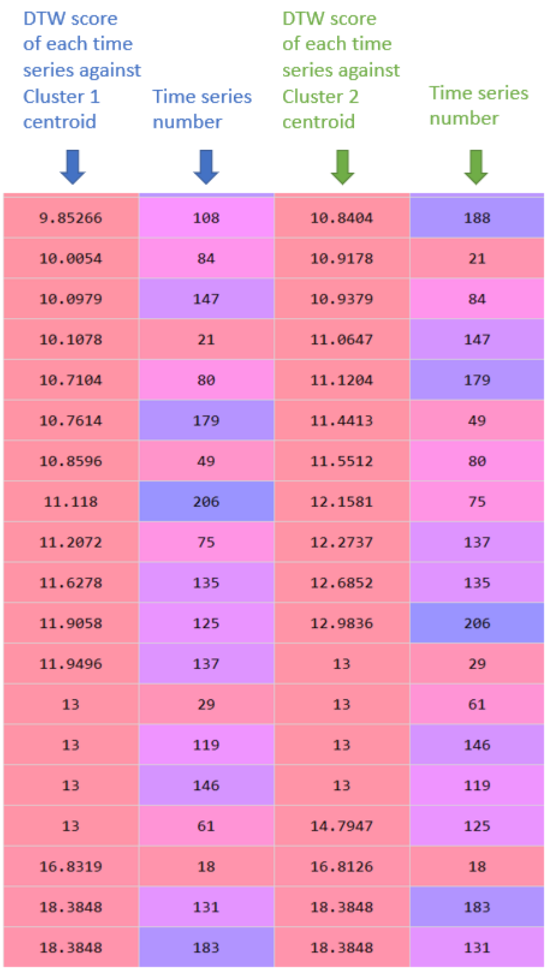
OD1: Remove sparse/undesirable clusters

In this method, the data set is clustered into either the optimal number of clusters given by a Silhouette score, or into the number of clusters which the program user wishes to compute in the data set. The user of the program may then visually inspect the clusters and remove clusters that he/she judges to be an outlier cluster. Alternatively, a threshold which specifies how sparse an inlier cluster may be applied, where clusters below the sparseness threshold are moved from the inlier to the outlier data set.

OD2: Remove outliers based on extra-cluster dissimilarity

In this method, the data set is first clustered into the optimal number of clusters (judging by the Silhouette Score), then each cluster centroid is compared with DTW to each time series in the entire data set. A DTW threshold is then specified. All time series which achieve a "worse" DTW-score that the threshold when compared to *all* clusters will be removed from the inlier data set to the outlier data set. This is illustrated in Figure 24 and Figure 25.

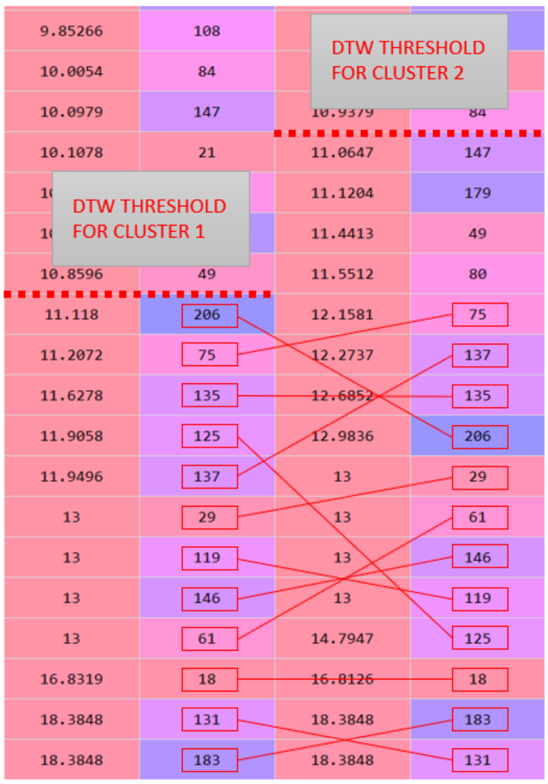
As an example, given a data set of 218 time series, and an optimal number of clusters `K_optimal`=2, the centroid of cluster 1 and the centroid of cluster 2 are in turn compared to all 218 time series. Figure 24 shows a screenshot of an extract of the result when this is applied to an actual AMS time series data set.



DTW score of each time series against Cluster 1 centroid	Time series number	DTW score of each time series against Cluster 2 centroid	Time series number
9.85266	108	10.8404	188
10.0054	84	10.9178	21
10.0979	147	10.9379	84
10.1078	21	11.0647	147
10.7104	80	11.1204	179
10.7614	179	11.4413	49
10.8596	49	11.5512	80
11.118	206	12.1581	75
11.2072	75	12.2737	137
11.6278	135	12.6852	135
11.9058	125	12.9836	206
11.9496	137	13	29
13	29	13	61
13	119	13	146
13	146	13	119
13	61	14.7947	125
16.8319	18	16.8126	18
18.3848	131	18.3848	183
18.3848	183	18.3848	131

Figure 24: Comparing (DTW) cluster centroids to each time series in a data set.

If a DTW threshold of 11 were applied, the following time series would be selected and moved from the inlier data set to the outlier data set. The outliers are marked in red in Figure 25.



9.85266	108	[illegible]	[illegible]
10.0054	84	[illegible]	[illegible]
10.0979	147	10.9379	84
10.1078	21	11.0647	147
[illegible]	[illegible]	11.1204	179
[illegible]	[illegible]	11.4413	49
10.8596	49	11.5512	80
11.118	206	12.1581	75
11.2072	75	12.2737	137
11.6278	135	12.6852	135
11.9058	125	12.9836	206
11.9496	137	13	29
13	29	13	61
13	119	13	146
13	146	13	119
13	61	14.7947	125
16.8319	18	16.8126	18
18.3848	131	18.3848	183
18.3848	183	18.3848	131

Figure 25: Selecting time series which are found to be outliers in all clusters.

OD3: Remove outliers based on intra-cluster dissimilarity

In this method, the data set is clustered into either the optimal number of clusters given by a Silhouette Score, or into the number of clusters which the program user wishes to compute in the data set. Each cluster centroid is then compared using DTW to each time series that "belong" to the cluster. A DTW threshold is applied by the program user to the intra-cluster DTW score, and time series that have a "worse" score than the threshold will be moved from the cluster to the outlier data set.

7.5.3 Possibility to implement other methods of outlier analysis

As stated in Section 7.5.2 the customer programme model is modular, which implies that it is possible to implement more advanced outlier analysis code in the programme. One such advanced outlier analysis method evaluated for use was time series residuals analysis. The residuals of a time series may be computed in Python with the `statsmodels` library's Seasonal-Trend decomposition using LOESS (STL) functions. Information on the `statsmodels` STL functions may be found in resource [80], and information on analysis of residuals may be found in resource [31].

Although not written into the programme code in Appendix IV it is easily possible to modify the code so that a customer segmentation analysis can be performed on the outlier data set independently. This could be done by uploading the outlier `DataFrame` to a Comma-Separated Values (CSV) file with a `pandas` function, and then uploading the outlier CSV-file into the start of the programme as the programme “main file”.

A thorough survey of time series outlier detection techniques may be found in resource [9].

7.5.4 Time series manipulation

Though not included in the customer segmentation programme in this report, it is possible to manipulate the AMS-data in Step 2. of the programme as shown in Figure 28. The two forms of time series manipulation which were considered for the programme but not included were rolling averages, and dividing the AMS-data set into subsets based on mean measurement values.

Using a rolling average on a data set could help to “smooth out” some short-term fluctuations in the data, while still preserving the overall shape of the time series. This would assist the K-Shape algorithm from `tslearn` to compute the clusters and cluster centroids more efficiently. `tslearn` includes three functions for computing rolling averages of time series data. Examples of the application of these three functions is shown in Figure 26. Details on the individual functions in Figure 26 may be found in reference [88].

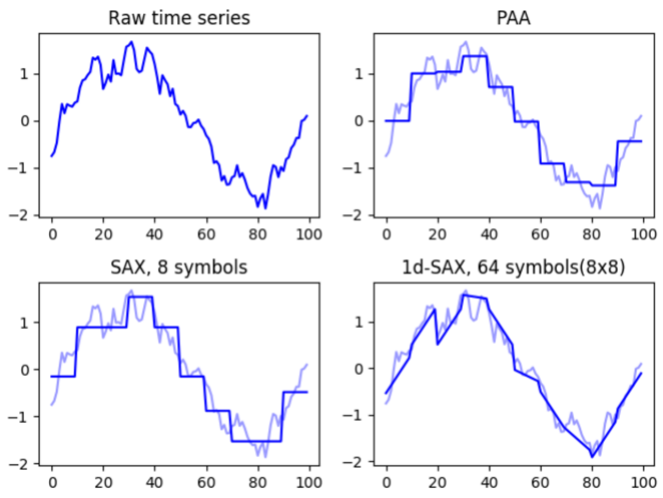


Figure 26: Different functions for computing rolling averages of time series found in the `tslearn` library [88].

Dividing AMS-data into subsets based on the mean of the different time series could prove useful when analysing small-scale and large-scale utility customers separately. As can be seen in Figure 27, which is a plot of the AMS-data set analysed in Section 10.1, it may be interesting to analyse large- and small-scale customers separately. This could be done by calculating the mean of each time series with `numpy.mean` in Python along the first axis of the data set, and then dividing the data set into subsets according to thresholds for the mean.

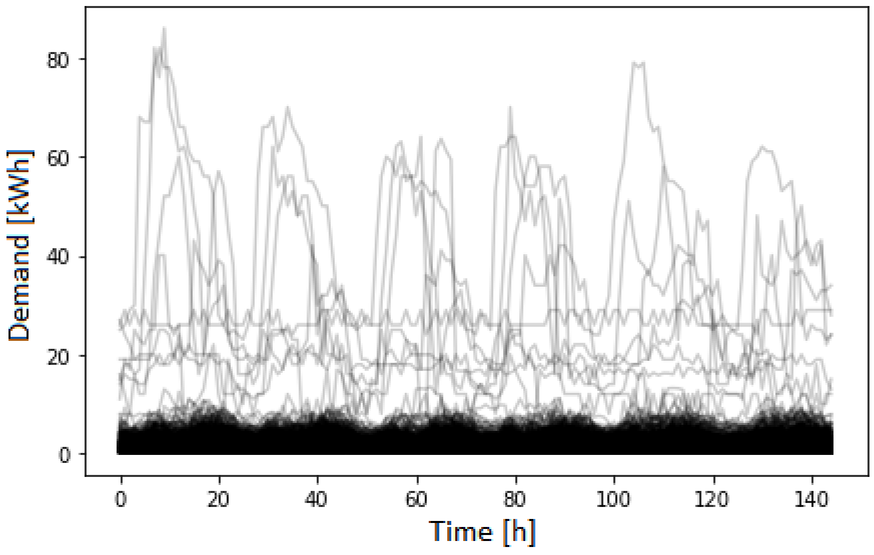


Figure 27: AMS data set spanning one week in 2018 used in this report.

8 The developed customer segmentation programme

This Section explains the developed programme for discovering customer groups of similarly shaped power consumption profiles within a set of AMS-data. The programme is based on a clustering approach (introduced in Section 7.2.2).

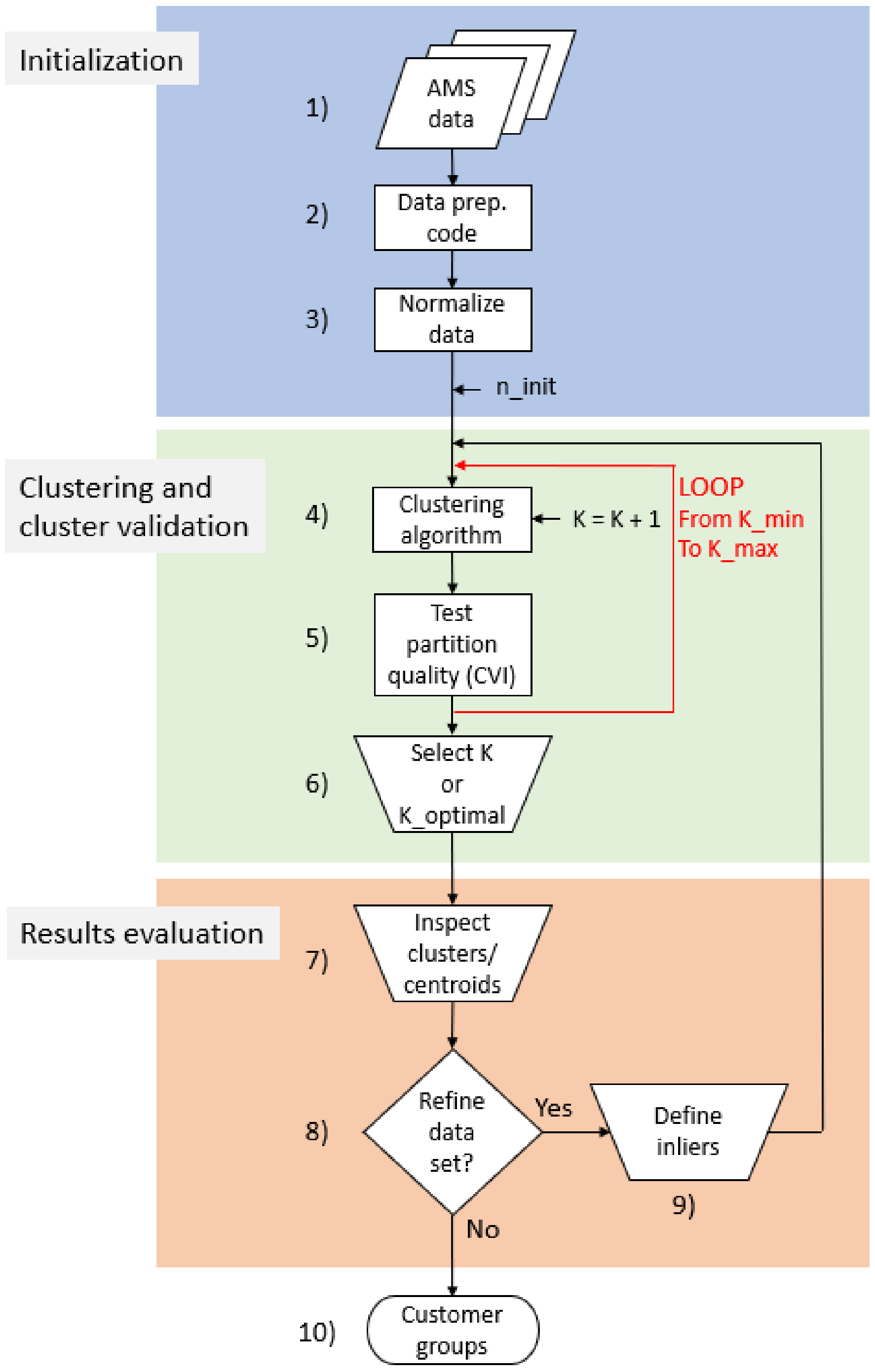


Figure 28: Developed method for producing customer segments given a set of AMS-data.

When the programme is running, it will ask the user to specify two types of parameters. The first parameter, `n_init`, defines how many iterations the `KShape` algorithm (i.e. the K-Shape algorithm from `tslearn`) will be run for. This parameter, varies according to the "stability limits" of the `KShape` algorithm for the specific data set. Further elaborated in Section 7.3.2.

The second type of parameters is an input to the `KShape` algorithm which specifies how many clusters the algorithm will find in the data set. An interval of `K`, between `K_min` and `K_max` is specified by the programme user based on what partitions the user considers to be realistic/practical to test the suitability of for a given AMS time series data set, i.e. the various number of clusters to be tested by the algorithms for the data set. It may not be clear to the programme user how many partitions he/she wishes to find in the data set. Instead the user defines an interval. This aims to ensure a result with a reasonable and practical number of clusters, when in theory each time series could be defined as one cluster.

The most important steps in the developed computer programme is illustrated in Figure 28.

The selected customer segmentation method in Figure 28 consists of the following steps:

1. The algorithm starts with a set of AMS-data which is listed row-wise in a .txt file as input. Some AMS-data formats are explained in Section 4.4, giving the reader an idea of its initial appearance.

2. The AMS-data is converted with code in Python such that it is stored in a format which is readable to the algorithms in the Python `tslearn` library. More on the required format of the data following data preparation is found in Section 9.1.

3. The AMS-data is then normalized to facilitate a shape-based customer segmentation approach. This means that the clustering algorithm recognizes the similarity of two customer consumption profiles irrespective of amplitude variations, offset variations and time shifts. This is explained in more detail in Section 7.1.

   Before entering Step 4) the programme user will be asked to select a value for `n_init`, which will decide the integer number of iterations that the `KShape` algorithm performs for each `K` in Step 4). The value for `n_init` that produce stable Silhouette scores for any given `K` can be found experimentally by varying `n_init`, while keeping `K` constant. Generally, larger/more complex data sets will require a larger `n_init`. More on `n_init` and K-Shape algorithm stability is found in Section 7.3.2.

4. `K` is used to denote the number of clusters that the clustering algorithm in the customer segmentation programme is instructed to find in the AMS-data set. `K` is an integer which is incrementally increased in a for loop from some minimal `K` (`K_min`), to some maximal `K` (`K_max`) which are defined by the user of the programme. `K_min` and `K_max` define the interval of customer segments

to be tested. An interval of `Ks` is tested in order to generate Silhouette scores for multiple `Ks` in Step 5) and aid the user in selecting a `K` suitable for the data set.

5. The partition quality of each `K` used in the for loop is then tested by a CVI. When the for loop is completed, i.e. the CVI (i.e. Silhouette scores) for all `K_min` to `K_max` is calculated. Following this, the programme presents the user with the complete set of CVI scores for all `K`. This aids the user in determining the choice number of customer segments in a specific set of AMS-data in Step 6).

6. `K_optimal` is used to denote the optimal number of clusters `K`, as found by a CVI within the defined interval between the selected `K_min` and `K_max`. Specifically for the Silhouette algorithm, the score closest to +1 is the optimal score, as explained in Section 7.4.1. The `K` for which the optimal Silhouette score is achieved is denoted `K_optimal`. However, the programme user may chose another `K`, for example in the case where `K_optimal` is calculated to be small, e.g. `K_optimal`=2 or large, e.g. `K_optimal`=`K_max`.

7. After selecting a `K`, the programme will plot the generated clusters, as well as the highlighted cluster centroids. The user is then able to visually inspect the clustering, and evaluate the programme results. Some guidelines as to evaluate clustering results may be found in Section 7.3.3.

8. Based on the evaluation of the results in Step 7), the user can chose to refine the data set by performing an outlier analysis as described in section 7.5.2.

9. Based on three simple outlier detection methods (OD1, OD2 and OD3) explained in Section 7.5, the AMS-data is sorted by the user of the programme into an outlier and inlier data set. This way, no data is deleted in the programme.

   After one of the three outlier detection methods have been applied, the programme directs the user back to Step 4). The steps Step 4) to 8) are then repeated with the refined data set, i.e. the inlier data set. This is because after removing outliers the Silhouette scores for different `Ks` changes, and partition quality may improve.

10. When the programme user is of the opinion that (further) refinement of the data set does not bring advantages, and/or considers the cluster partitions satisfactory, the programme is terminated.

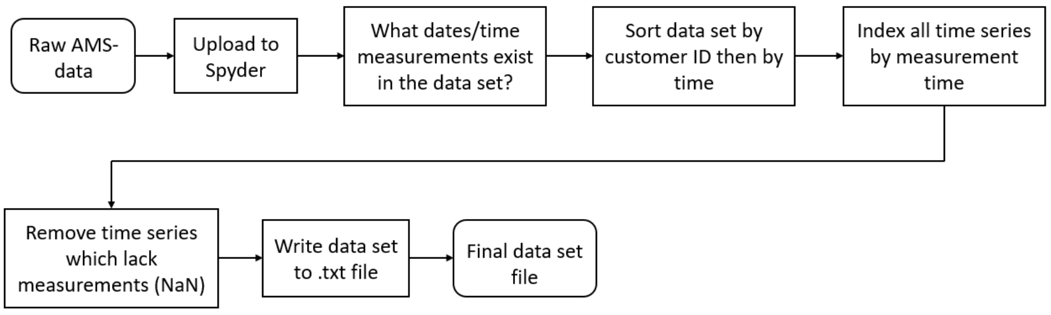
9 Assessing the developed customer segmentation programme using known data

In this Section the customer segmentation model as described in Section 8 is assessed. This is done by first testing the programme on a known data set, to see if the programme gives the results that are known beforehand to be "correct". Then, the programme is tested on an AMS-data set obtained from the DSO Lyse Elnett. Finally, the programme is compared to the currently applied standard DSO customer segmentation model as described in Section 6.2 and Figure 11.

9.1 Data preparation phase

This Subsection describes the data preparation phase of the customer segmentation model, which is step 2) in the illustration of the model in Figure 28, Section 8.

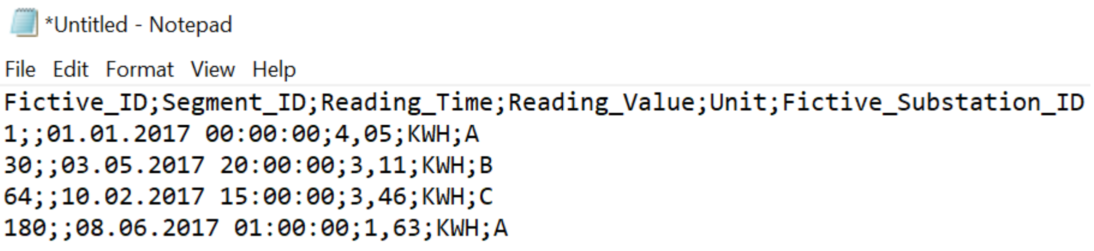
The data preparation phase of the AMS-data analysis does not include any machine learning code, and is mainly focused on indexing and sorting the data set to a format which appropriate for the subsequent steps of the programme. A chart of the data refinement and manipulation process is shown in Figure 29.



Raw AMS-data
Upload to Spyder
What dates/time measurements exist in the data set?
Sort data set by customer ID then by time
Index all time series by measurement time
Remove time series which lack measurements (NaN)
Write data set to .txt file
Final data set file

Figure 29: Chart of the data preparation phase of AMS-data analysis.

The data preparation code is written for a data set which has a format as shown in Figure 30. The code is given in Appendix II, with an explanation of how the code may be adjusted to AMS-data of other formats than the one shown in Figure 30. These other formats are described in Section 4.4.



```
Fictive_ID;Segment_ID;Reading_Time;Reading_Value;Unit;Fictive_Substation_ID
1;;01.01.2017 00:00:00;4,05;KWH;A
30;;03.05.2017 20:00:00;3,11;KWH;B
64;;10.02.2017 15:00:00;3,46;KWH;C
180;;08.06.2017 01:00:00;1,63;KWH;A
```

Figure 30: Data set for which the data preparation code is written.

The result of the data preparation code is a .txt file where all AMS time series are sorted by ID row-wise and time column-wise. This is illustrated in Table 2.

Table 2: The format to which the processed AMS-data set is written to a .txt file.

	`Start_date_time`	`Start_date_time` + 1h	...	`End_date_time`
First ID	3.09	4.92	...	3.45
Second ID	3.83	3.11	...	4.04
...	...	...	...	...
Final ID	0.46	0.44	...	1.4

For a data set to be usable to a `tslearn` function it will need to have the format as shown in Table 2, because this is the format that the `tslearn` Python library is able to understand.

9.2 Performance test of simplified programme without outlier analysis

The steps 1) to 6) of the customer segmentation model as shown in Figure 28 was tested to assess how effective the Silhouette and K-Shape algorithms are in evaluating the optimal number of clusters `K_optimal` in a time series data set, and how the data set is partitioned given `K_optimal`.

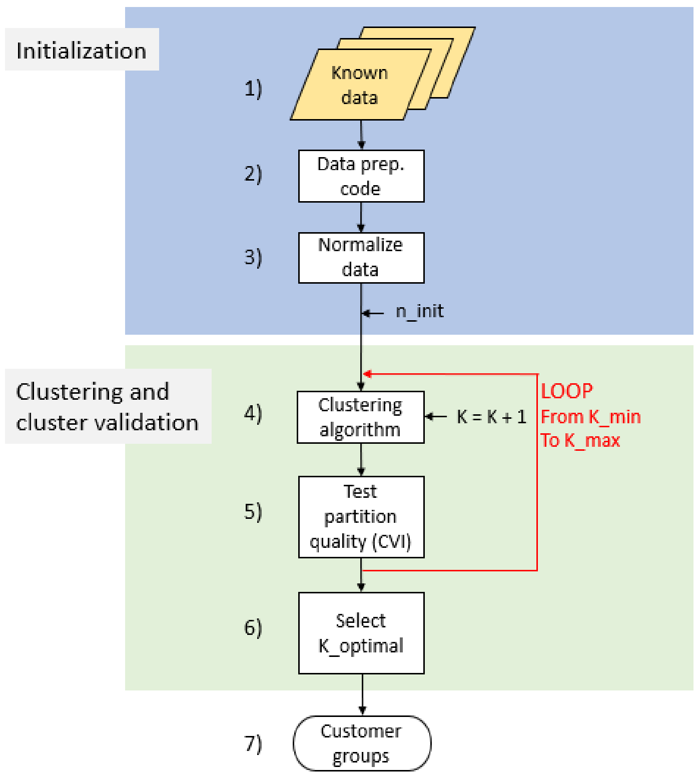


Figure 31: The programme steps 1) to 6) used in this Section for testing known data sets.

Before using the complete customer segmentation programme (Figure 28) to analyse and detect customer groups in actual AMS-data sets, the method was tested with a known data set where the different time series groups were visually distinct and identifiable. It was considered important to test whether the Silhouette and K-Shape algorithms would provide the results that were known beforehand to be correct.

Reproducibility of scientific research is important for the credibility of the work, and to enable peer-review [72]. The reproducibility of results found in Section 9.2 and Section 9.3 aims to ensure the credibility of the findings in this report. As such, Appendix III and Appendix IV contains all code used in every test performed for Section 9.2 and Section 9.3. It should be possible to recreate all data sets used in Section 9.2 and Section 9.3 with the code from Appendix III and Appendix IV, and thus reproduce analysis results with the Silhouette analysis and K-Shape clustering code. The reader should practice some caution when attempting to reproduce results in Section 9.2.6 and Section 9.3 due to the randomness of the data added in these tests. Care has been taken to not draw firm conclusions when statistically significant results were lacking, and therefore analysis results should probably be possible to reproduce even though the data set contains (pseudo) randomness.

9.2.1 The applied known data set

The known data set consists of four known and distinct time series groups, with a minimum of 50 time series per group. The following time series groups were included:

- A square wave time series set with a varying amplitude, with no phase shift and a constant frequency.

  This time series group was chosen to demonstrate the amplitude invariance of the programme.

- A sine wave time series set with a slight phase shift between some of the sine wave time series, no varying amplitude and a constant frequency.

  This time series group was chosen to see what effect two periodic time series groups (the sine and square wave groups) with different frequencies would have on the programme.

- A set of time series made available by `tslearn` characterised by a peak, then drop, and finally stabilising to the starting value. These time series are shifted in time and have slightly varying amplitude and offset.

  This time series group was included to test the accuracy of the K-Shape and silhouette algorithm for non-periodic time series.

- A set of time series made available by `tslearn` characterised by a sloping increase in level from one stable level to another higher level. Some time series in this group also display oscillations after reaching the higher stable measurement value. The time series in this group are shifted in time and have a slightly varying amplitude and offset.

This time series group was also included to test the accuracy of the K-Shape and silhouette algorithm for non-periodic time series.

The different time series groups listed above are shown in Figure 32. This data set forms the basis for the tests performed in Section 9.2.2 to Section 9.3.

As the unit-value on the y-axis of the different known time series groups was undefined, the y-axis value have been assumed (for illustrative purposes) to be kilowatt hours, kWh. The value on the x-axis have been assumed to be time, in hours.

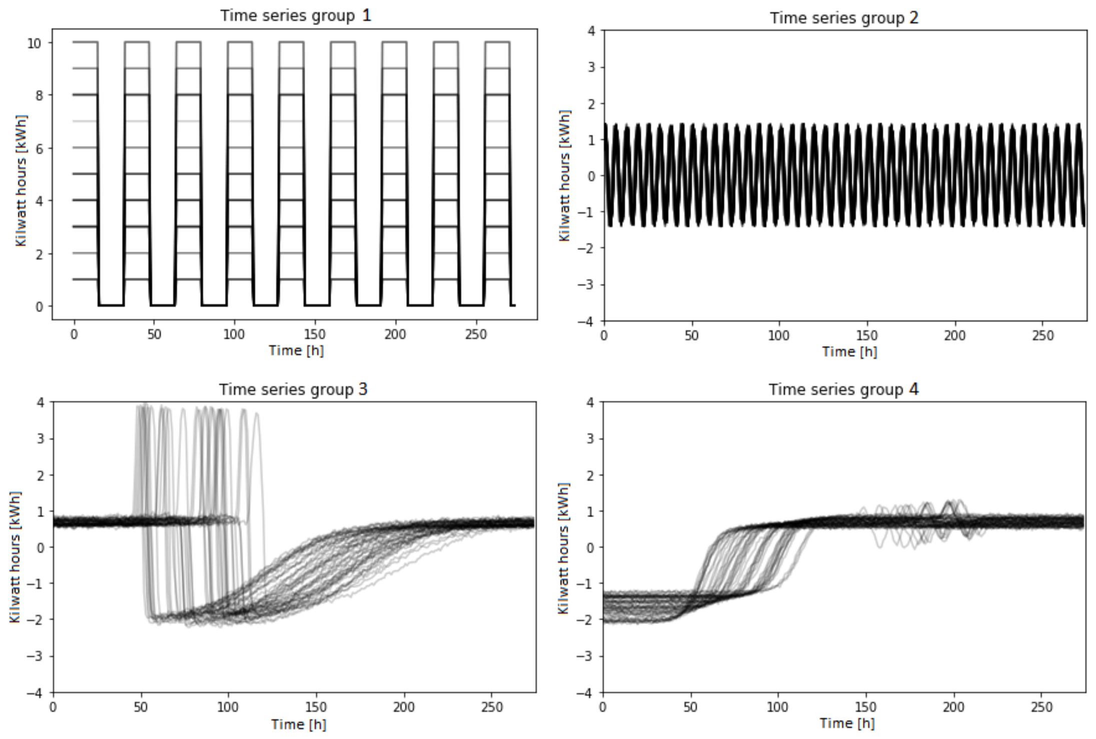


Figure 32: The known time series groups used for testing the developed programme.

9.2.2 Base case

The code given in Appendix III, Section C.1 was tested on the set of time series described in Section 9.2.1. The original appearance of the time series data set before any operations were performed on them are displayed in one plot in Figure 33.

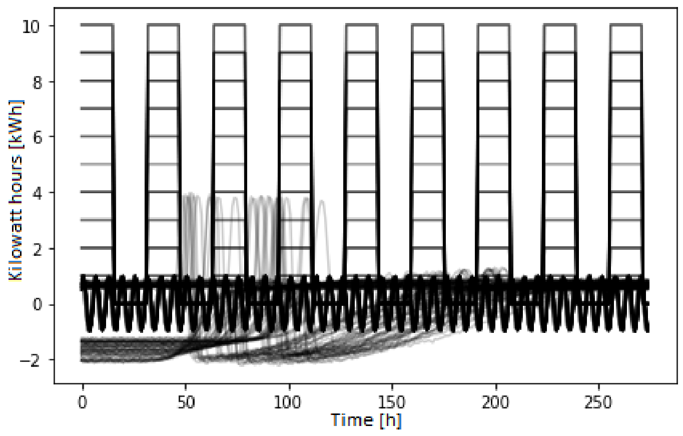


Figure 33: All time series in the Base case data set.

As explained in Section 7.1.3, amplitude and offset invariance are implemented by normalizing all time series in the data set. The appearance of the time series data set after normalization is shown in Figure 34.

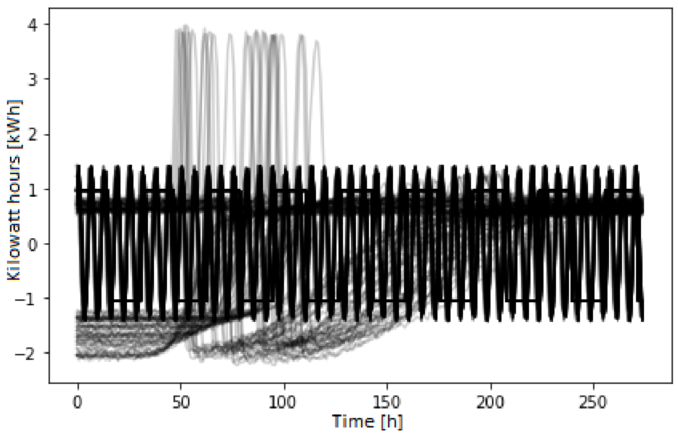


Figure 34: All time series in the Base case data set, normalized.

The Silhouette algorithm tested each partition of the normalized data set between `K`=2 and `K`=9, and gave the results shown in Table 3 and Figure 35:

Table 3: Silhouette scores for all partitions of the Base case data set, between `K`=2 and `K`=9 clusters.

Number of clusters:	Silhouette Score:
K = 2	0.352
K = 3	0.619
K = 4	0.884
K = 5	0.801
K = 6	0.813
K = 7	0.779
K = 8	0.757
K = 9	0.593

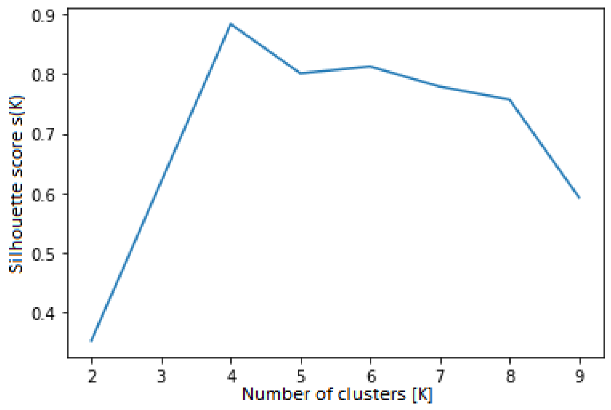


Figure 35: Silhouette score for all partitions between `K`=2 and `K`=9 clusters.

From Table 3 and Figure 35 it is clear that the Silhouette algorithm gives the partition of `K`=4 an optimal score, i.e. `K_optimal`=4. This is the desired result.

The K-Shape algorithm was instructed to find `K_optimal`=4 clusters, in order to view the discovered optimal solution. The resulting clusters are shown in Figure 36. The red time series in each plot is the cluster centroid found in the cluster partition by the K-Shape algorithm.

As explained in Section 7.1.3, the clustering algorithm provides offset, amplitude and time shift invariant time series centroids, i.e. the K-Shape algorithm uses shape-based distance. Since the centroids are phase invariant, a slight phase shift may be observed in Figure 36, particularly in Cluster 2 and 4 between cluster centroid and cluster time series.

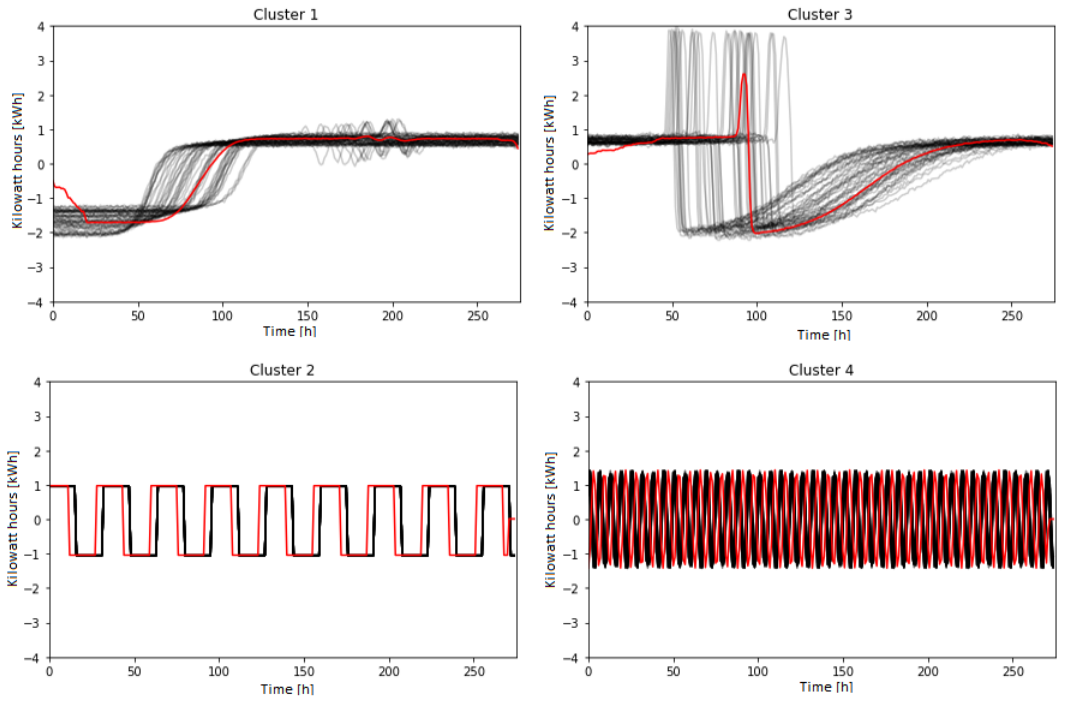


Figure 36: Result of using K-Shape clustering on the Base case data set, with `K_optimal`=4.

It may be concluded from the results that given a known data set with visually distinct groups the Silhouette algorithm is able to calculate an optimal number of clusters `K_optimal`, which corresponds to what may be found when visually inspecting the data set. It may also be concluded that the K-Shape algorithm finds the expected partitions in the time series data set provided `K_optimal` which is calculated by the Silhouette algorithm. The resulting cluster centroids visually correspond well to the cluster partitions.

Algorithm stability for the Base case

With the conditions for `KShape` algorithm stability described in Section 7.3.2, the stability region of the `KShape` algorithm was tested for the Base case for `K` between 2 and 9, by varying `max_iter` between 100 and 1000, and `n_init` between 2 and 200. A segmentation result was considered stable if it received consistent Silhouette scores. The results are illustrated in Figure 37. Green zones in the tables are considered stable areas, while red zones are unstable areas. From Figure 37 it may be seen that the size of `max_iter` does not influence algorithm stability, while `n_init` does. However, increasing `n_init` is far more computationally expensive for the algorithm than increasing `max_iter`. For this reason, `n_init` will be selected at the approximate stability boundary. For the Base case the approximate stability boundary lies at `n_init`=40 for `K` between 2 and 9, as may be seen in Figure 37.

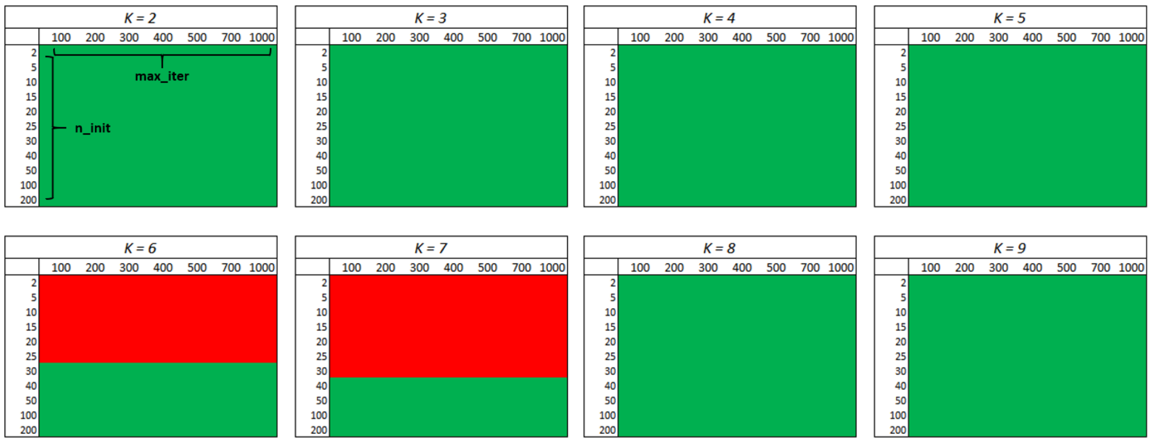


Figure 37: The stability regions for the Base case, for K between 2 and 9.

9.2.3 Effect of changing number of time series in one time series group

The code in Appendix I, Section C.2 was tested with one of the four time series groups described in Section 9.2.1 made to be very sparse. The other time series groups remained unchanged. The sparse time series group, which was chosen to be the sine wave group had only three time series as apposed to 50 which was the case in the Base case. The new data set a), and its difference to the data set used in the Base case b) is seen in Figure 38.

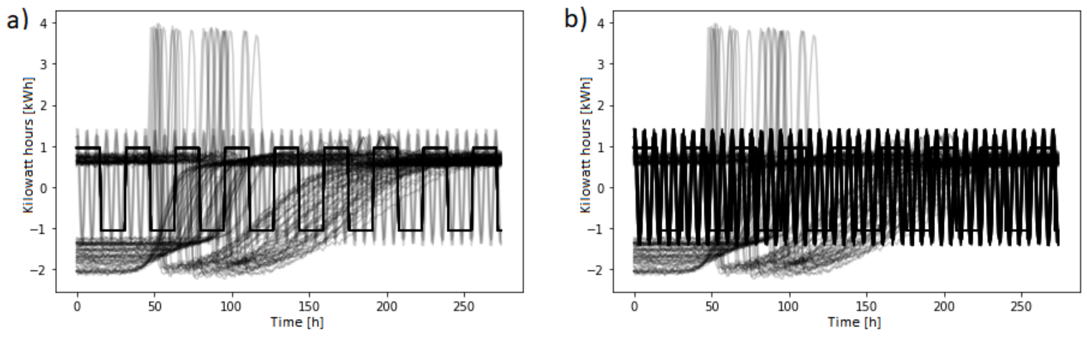


Figure 38: a) is the data set used in this section which has a sparse sine wave group after normalization, and b) is the data set used in the Base case test, normalized, which illustrates the Base case size of the sine wave group.

The Silhouette algorithm tested each partition of the normalized data set between K=2 and K=9, and gave the results shown in Table 4 and Figure 39.

Table 4: Silhouette scores for all partitions of the data set with one sparse sine wave group, between `K`=2 and `K`=9 clusters.

Number of clusters:	Silhouette Score:
K = 2	0.526
K = 3	0.542
K = 4	0.888
K = 5	0.780
K = 6	0.796
K = 7	0.666
K = 8	0.624
K = 9	0.512

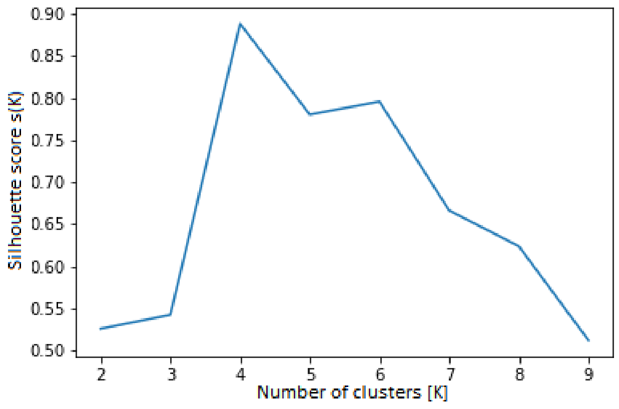


Figure 39: Graph of Silhouette scores for all partitions of the single sparse group data set, between `K`=2 and `K`=9 clusters.

Again, as may be seen from Table 4 and Figure 39, the optimal number of clusters was found to be `K_optimal`=4. The K-Shape algorithm was then instructed to find `K_optimal`=4 clusters in the data set, which produced the clusters and centroids shown in Figure 40.

From these results it may be observed that the Silhouette algorithm is able to detect sparse clusters and differentiate them from larger clusters. It is also interesting to observe that the maximum Silhouette score for the case in Section 9.2.2 is slightly lower than the maximum Silhouette score for the case in this section. This may be because the data set has been reduced by 47 time series samples.

The K-Shape algorithm is also able to correctly identify the sparse cluster and differentiate it from the other larger clusters when given the optimal number of clusters `K_optimal`.

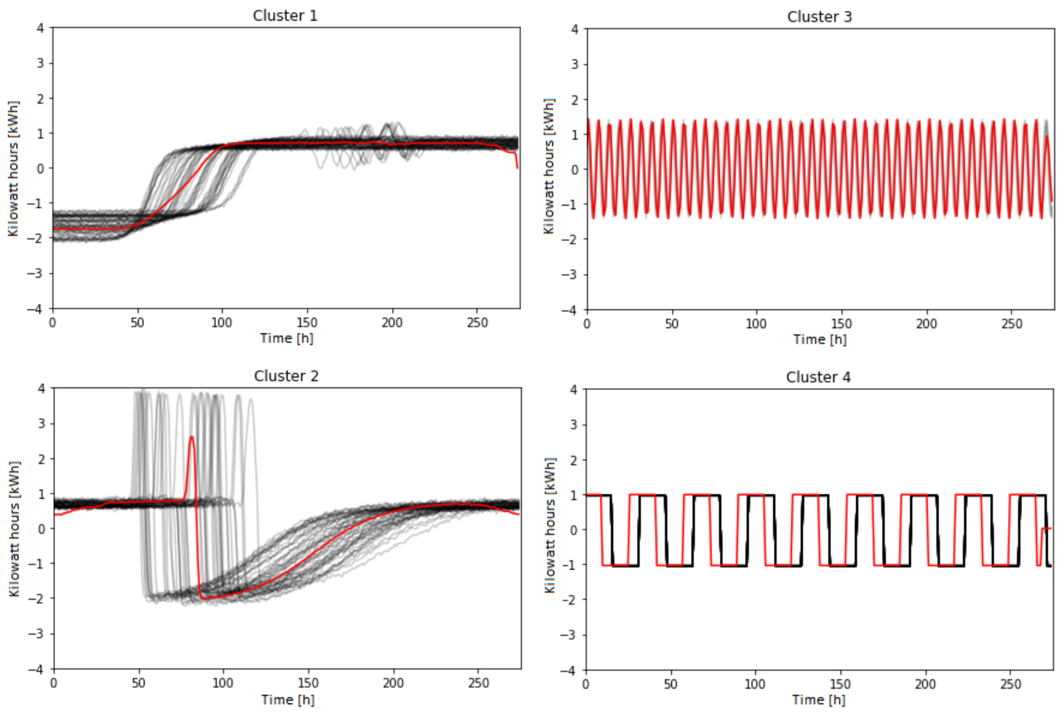


Figure 40: Result of using K-Shape clustering on the data set with one single sparse group, with `K_optimal`=4.

Algorithm stability

The stability of the K-Shape algorithm was tested for values of `max_iter` between 100 and 1000, and for values of `n_init` between 2 and 600 for all `K` between 2 and 9. The results are shown in Figure 41.

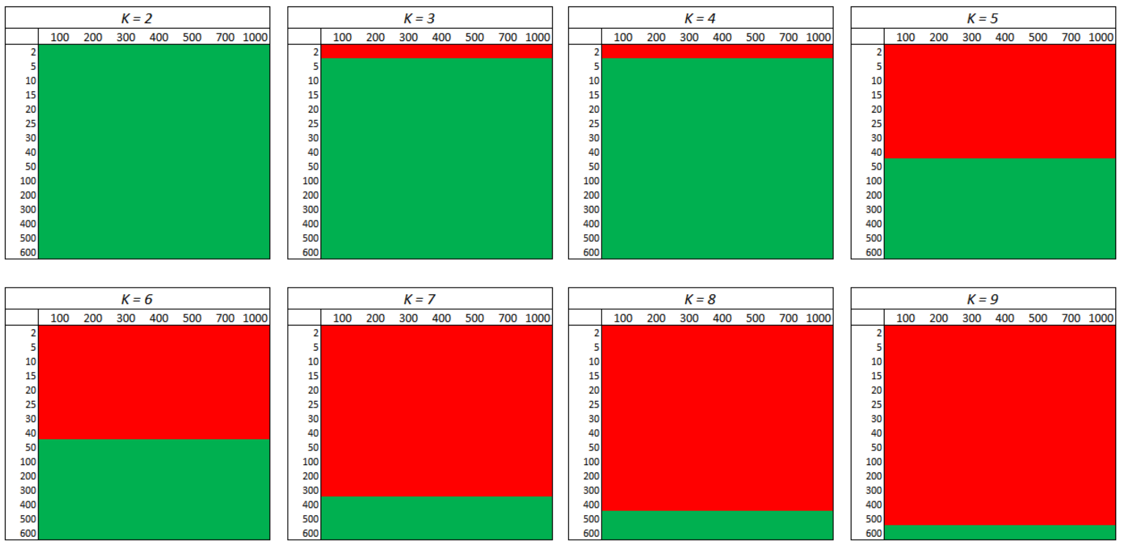


Figure 41: K-Shape algorithm stability for the data set with one sparse time series group.

It may again be observed that the variable `max_iter` does not effect the stability of the K-Shape algorithm. It may also be observed from Figure 41 that the K-Shape algorithm has become significantly less stable when compared with the Base case, Figure 37 (K-Shape algorithm becomes stable for `n_init`≈40 iterations for `K` between 2 and 9). This may indicate that the K-Shape algorithm becomes less stable when attempting to find clusters in data sets with sparse groups. For the data set in this section, the K-Shape algorithm becomes stable for `n_init`≈600 for `K` between 2 and 9.

9.2.4 Testing time invariance of periodic time series

Time/phase invariance may already be observed in the Base case, Figure 36 Cluster 1 and 3. However, it was unknown whether the Silhouette and K-Shape algorithms would cluster two periodic time series of opposite phase into one or two clusters. This property was tested by adding a sine wave of opposite polarity to the sine time series data set, to see if the K-Shape algorithm would differentiate the sine waves. The code for this test is given in Appendix I, Section C.3. The two time series groups imported from `tslearn` as described in Section 9.2.1 were removed from the data set to simplify it. The data set then included:

- The square wave group as described in Section 9.2.1, with slightly fewer data points per time series (150 in this data set, and 275 in the data set in the Base case).
- The sine wave group as described in the Section 9.2.1, with slightly fewer data points per time series (150 in this data set, and 275 in the data set in in the Base case).
- A sine wave group identical to the one described in Section 9.2.1, but with 180 degrees phase shift. This group also had 150 data points per time series.

The data set had the appearance as shown in Figure 42 before and after normalization.

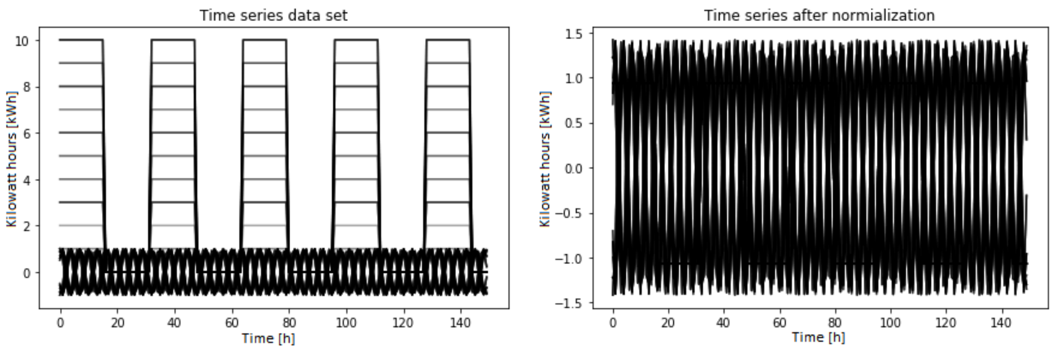


Figure 42: Time series data set for testing time invariance before and after normalization.

The Silhouette algorithm tested each partition of the normalized data set between `K`=2 and `K`=5, and gave the following results:

Table 5: Silhouette scores for all partitions of the data set used for testing time invariance, between `K`=2 and `K`=5 clusters.

Number of clusters:	Silhouette Score:
K = 2	0.848
K = 3	0.575
K = 4	0.717
K = 5	0.624

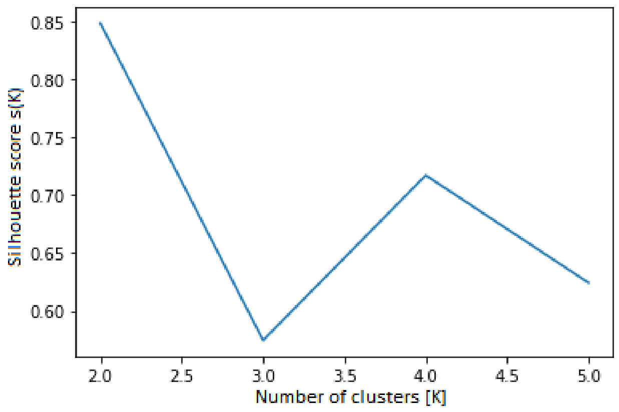


Figure 43: Silhouette scores for all partitions of the time invariance testing data set, between `K`=2 and `K`=5 clusters.

From Table 5 and Figure 43 it is clear that `K_optimal`=2 produces the best partition result. When told to produce `K_optimal`=2 clusters, the K-Shape algorithm gives the results displayed in Figure 44.

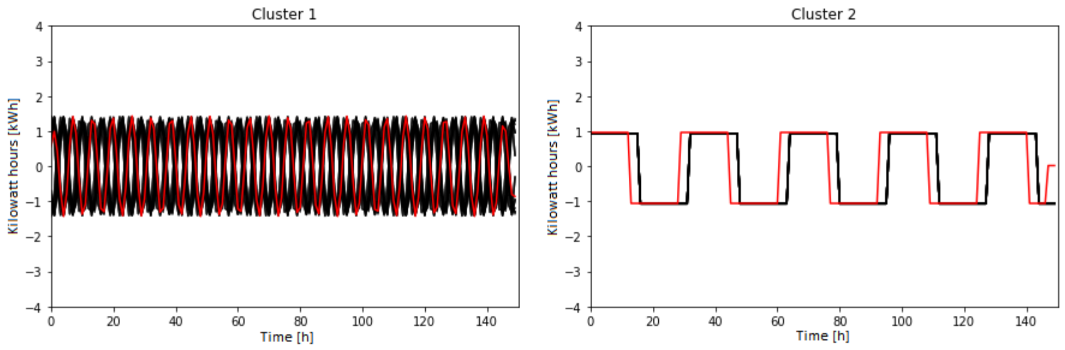


Figure 44: Result of using K-Shape clustering on the data set for testing time invariance, with `K`=2.

These results demonstrate that the method is time invariant, and will group periodic time series into the same cluster, even if they are of opposite phase. The algorithm stability was not tested for this Section, as it was evaluated to have no utility when compared to the Base case. `n_init` was chosen at the stability boundary of the Base case, which was 40 iterations.

9.2.5 Effect of length of data set on programme accuracy

The programme code given in Appendix I Section C.4 was tested to see if it would provide different results given a data set half the length of the data set in Section 9.2.1. In the Section 9.2.1 each time series has 275 measurement points, while the time series in the data set used in this Section and shown in Figure 45 contained only 136 data points per time series.

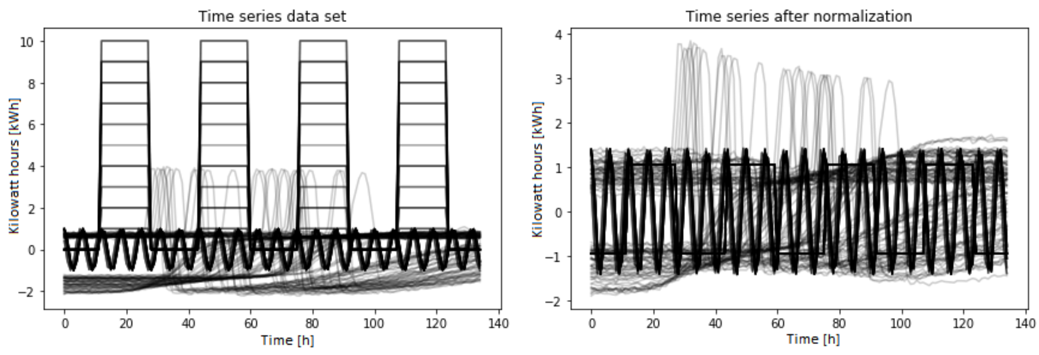


Figure 45: The data set used for testing the programme with shorter time series, before and after normalization.

The Silhouette algorithm tested each partition of the normalized data set between `K`=2 and `K`=9, and gave the results shown in Table 6 and Figure 46:

Table 6: Silhouette scores for all partitions of the shorter time series data set, between `K`=2 and `K`=9 clusters.

Number of clusters:	Silhouette Score:
K = 2	0.238
K = 3	0.509
K = 4	0.812
K = 5	0.771
K = 6	0.724
K = 7	0.725
K = 8	0.780
K = 9	0.665

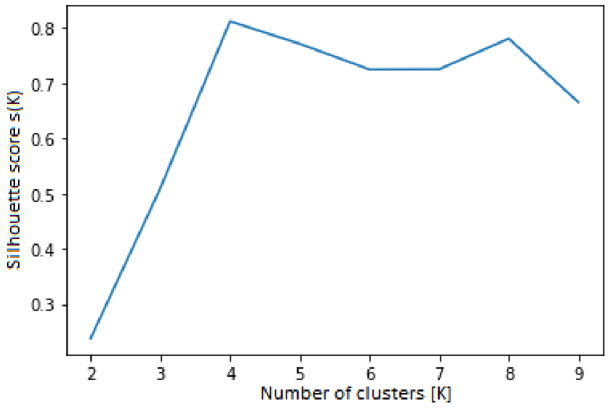


Figure 46: Silhouette scores for the shorter time series data set.

As may be seen from Table 6 and Figure 46, the optimal number of clusters was found to be `K_optimal`=4. The K-Shape algorithm was then instructed to find `K_optimal`=4 clusters in the data set, which produced the clusters and centroids shown in Figure 47.

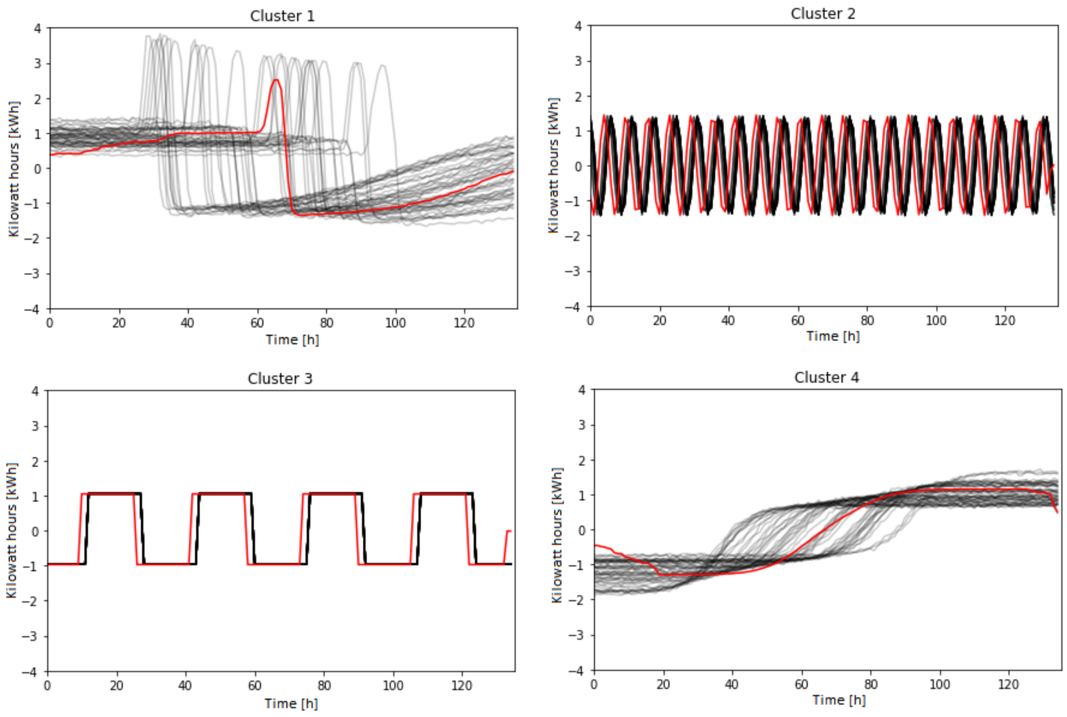
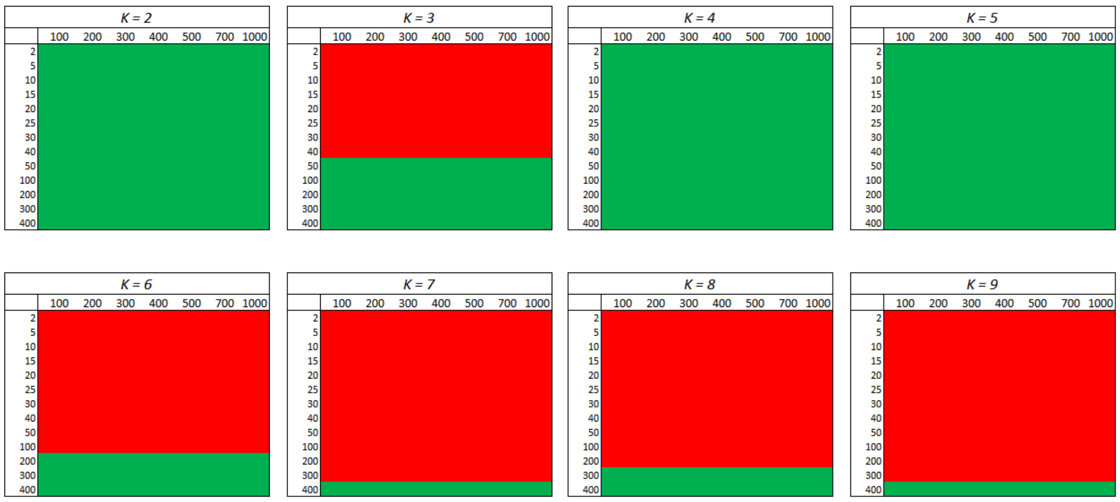


Figure 47: Result of using K-Shape clustering on the data set used for testing the effect of using shorter time series, with `K`=4.

As may be seen in Figure 47 the programme produced the “correct results”, and the cluster centroid is similar in shape to the four different time series groups. It should be noted that the shape of the time series has remained largely intact, even though data has been removed. This result seems to indicate that for the known data set the K-Shape algorithm is stable as long as the predominant shape of the time series remains present. Stability over time, i.e. over varying time periods of the data set, was a condition for cluster usefulness as described in Section 7.3.3.

Algorithm stability

The stability of the K-Shape algorithm was tested for values of `max_iter` between 100 and 1000, and for values of `n_init` between 2 and 400 for all `K` between 2 and 9. It may again be observed that the variable `max_iter` does not affect the stability of the K-Shape algorithm. It may also be observed from Figure 48 that the K-Shape algorithm has become less stable when compared with the Base case, Figure 37. This may indicate that the K-Shape algorithm's accuracy is reduced by having less data in the data set.



	K = 2						
	100	200	300	400	500	700	1000
2	🟩	🟩	🟩	🟩	🟩	🟩	🟩
5	🟩	🟩	🟩	🟩	🟩	🟩	🟩
10	🟩	🟩	🟩	🟩	🟩	🟩	🟩
15	🟩	🟩	🟩	🟩	🟩	🟩	🟩
20	🟩	🟩	🟩	🟩	🟩	🟩	🟩
25	🟩	🟩	🟩	🟩	🟩	🟩	🟩
30	🟩	🟩	🟩	🟩	🟩	🟩	🟩
40	🟩	🟩	🟩	🟩	🟩	🟩	🟩
50	🟩	🟩	🟩	🟩	🟩	🟩	🟩
100	🟩	🟩	🟩	🟩	🟩	🟩	🟩
200	🟩	🟩	🟩	🟩	🟩	🟩	🟩
300	🟩	🟩	🟩	🟩	🟩	🟩	🟩
400	🟩	🟩	🟩	🟩	🟩	🟩	🟩

	K = 3						
	100	200	300	400	500	700	1000
2	🟥	🟥	🟥	🟥	🟥	🟥	🟥
5	🟥	🟥	🟥	🟥	🟥	🟥	🟥
10	🟥	🟥	🟥	🟥	🟥	🟥	🟥
15	🟥	🟥	🟥	🟥	🟥	🟥	🟥
20	🟥	🟥	🟥	🟥	🟥	🟥	🟥
25	🟥	🟥	🟥	🟥	🟥	🟥	🟥
30	🟥	🟥	🟥	🟥	🟥	🟥	🟥
40	🟥	🟥	🟥	🟥	🟥	🟥	🟥
50	🟩	🟩	🟩	🟩	🟩	🟩	🟩
100	🟩	🟩	🟩	🟩	🟩	🟩	🟩
200	🟩	🟩	🟩	🟩	🟩	🟩	🟩
300	🟩	🟩	🟩	🟩	🟩	🟩	🟩
400	🟩	🟩	🟩	🟩	🟩	🟩	🟩

	K = 4						
	100	200	300	400	500	700	1000
2	🟩	🟩	🟩	🟩	🟩	🟩	🟩
5	🟩	🟩	🟩	🟩	🟩	🟩	🟩
10	🟩	🟩	🟩	🟩	🟩	🟩	🟩
15	🟩	🟩	🟩	🟩	🟩	🟩	🟩
20	🟩	🟩	🟩	🟩	🟩	🟩	🟩
25	🟩	🟩	🟩	🟩	🟩	🟩	🟩
30	🟩	🟩	🟩	🟩	🟩	🟩	🟩
40	🟩	🟩	🟩	🟩	🟩	🟩	🟩
50	🟩	🟩	🟩	🟩	🟩	🟩	🟩
100	🟩	🟩	🟩	🟩	🟩	🟩	🟩
200	🟩	🟩	🟩	🟩	🟩	🟩	🟩
300	🟩	🟩	🟩	🟩	🟩	🟩	🟩
400	🟩	🟩	🟩	🟩	🟩	🟩	🟩

	K = 5						
	100	200	300	400	500	700	1000
2	🟩	🟩	🟩	🟩	🟩	🟩	🟩
5	🟩	🟩	🟩	🟩	🟩	🟩	🟩
10	🟩	🟩	🟩	🟩	🟩	🟩	🟩
15	🟩	🟩	🟩	🟩	🟩	🟩	🟩
20	🟩	🟩	🟩	🟩	🟩	🟩	🟩
25	🟩	🟩	🟩	🟩	🟩	🟩	🟩
30	🟩	🟩	🟩	🟩	🟩	🟩	🟩
40	🟩	🟩	🟩	🟩	🟩	🟩	🟩
50	🟩	🟩	🟩	🟩	🟩	🟩	🟩
100	🟩	🟩	🟩	🟩	🟩	🟩	🟩
200	🟩	🟩	🟩	🟩	🟩	🟩	🟩
300	🟩	🟩	🟩	🟩	🟩	🟩	🟩
400	🟩	🟩	🟩	🟩	🟩	🟩	🟩

	K = 6						
	100	200	300	400	500	700	1000
2	🟥	🟥	🟥	🟥	🟥	🟥	🟥
5	🟥	🟥	🟥	🟥	🟥	🟥	🟥
10	🟥	🟥	🟥	🟥	🟥	🟥	🟥
15	🟥	🟥	🟥	🟥	🟥	🟥	🟥
20	🟥	🟥	🟥	🟥	🟥	🟥	🟥
25	🟥	🟥	🟥	🟥	🟥	🟥	🟥
30	🟥	🟥	🟥	🟥	🟥	🟥	🟥
40	🟥	🟥	🟥	🟥	🟥	🟥	🟥
50	🟥	🟥	🟥	🟥	🟥	🟥	🟥
100	🟥	🟥	🟥	🟥	🟥	🟥	🟥
200	🟩	🟩	🟩	🟩	🟩	🟩	🟩
300	🟩	🟩	🟩	🟩	🟩	🟩	🟩
400	🟩	🟩	🟩	🟩	🟩	🟩	🟩

	K = 7						
	100	200	300	400	500	700	1000
2	🟥	🟥	🟥	🟥	🟥	🟥	🟥
5	🟥	🟥	🟥	🟥	🟥	🟥	🟥
10	🟥	🟥	🟥	🟥	🟥	🟥	🟥
15	🟥	🟥	🟥	🟥	🟥	🟥	🟥
20	🟥	🟥	🟥	🟥	🟥	🟥	🟥
25	🟥	🟥	🟥	🟥	🟥	🟥	🟥
30	🟥	🟥	🟥	🟥	🟥	🟥	🟥
40	🟥	🟥	🟥	🟥	🟥	🟥	🟥
50	🟥	🟥	🟥	🟥	🟥	🟥	🟥
100	🟥	🟥	🟥	🟥	🟥	🟥	🟥
200	🟥	🟥	🟥	🟥	🟥	🟥	🟥
300	🟥	🟥	🟥	🟥	🟥	🟥	🟥
400	🟩	🟩	🟩	🟩	🟩	🟩	🟩

	K = 8						
	100	200	300	400	500	700	1000
2	🟥	🟥	🟥	🟥	🟥	🟥	🟥
5	🟥	🟥	🟥	🟥	🟥	🟥	🟥
10	🟥	🟥	🟥	🟥	🟥	🟥	🟥
15	🟥	🟥	🟥	🟥	🟥	🟥	🟥
20	🟥	🟥	🟥	🟥	🟥	🟥	🟥
25	🟥	🟥	🟥	🟥	🟥	🟥	🟥
30	🟥	🟥	🟥	🟥	🟥	🟥	🟥
40	🟥	🟥	🟥	🟥	🟥	🟥	🟥
50	🟥	🟥	🟥	🟥	🟥	🟥	🟥
100	🟥	🟥	🟥	🟥	🟥	🟥	🟥
200	🟥	🟥	🟥	🟥	🟥	🟥	🟥
300	🟩	🟩	🟩	🟩	🟩	🟩	🟩
400	🟩	🟩	🟩	🟩	🟩	🟩	🟩

	K = 9						
	100	200	300	400	500	700	1000
2	🟥	🟥	🟥	🟥	🟥	🟥	🟥
5	🟥	🟥	🟥	🟥	🟥	🟥	🟥
10	🟥	🟥	🟥	🟥	🟥	🟥	🟥
15	🟥	🟥	🟥	🟥	🟥	🟥	🟥
20	🟥	🟥	🟥	🟥	🟥	🟥	🟥
25	🟥	🟥	🟥	🟥	🟥	🟥	🟥
30	🟥	🟥	🟥	🟥	🟥	🟥	🟥
40	🟥	🟥	🟥	🟥	🟥	🟥	🟥
50	🟥	🟥	🟥	🟥	🟥	🟥	🟥
100	🟥	🟥	🟥	🟥	🟥	🟥	🟥
200	🟥	🟥	🟥	🟥	🟥	🟥	🟥
300	🟥	🟥	🟥	🟥	🟥	🟥	🟥
400	🟩	🟩	🟩	🟩	🟩	🟩	🟩

Figure 48: Result of using K-Shape clustering on the data set with shorter time series, with `K_optimal`=4.

9.2.6 Effect of adding randomness to the data set

Up until this point the method has been tested with well-defined time series groups that include a small to no degree of randomness/noise. In this section, Random Walk time series will be added to the known data set to test how severely the method affected by increasing randomness in the data set.

To the data set described in Section 9.2.1, 25 Random Walk time series were added. This means that around 10% of the time series in the data set displayed random behaviour. The Random Walk time series were computed with the formula $ts[t] = ts[t-1] + a$, where a is drawn from a normal distribution of mean $\mu = 0$ and standard deviation $\sigma = 1$ [87]. This was done with the code given in Appendix II, Section C.5. Ten such time series were generated and tested in ten different run of the programme as illustrated in Figure 31 for this Section.

Figure 49 shows a data set used in one of the ten runs. The Random Walk time series are clearly visible, particularly before the data set is normalized.

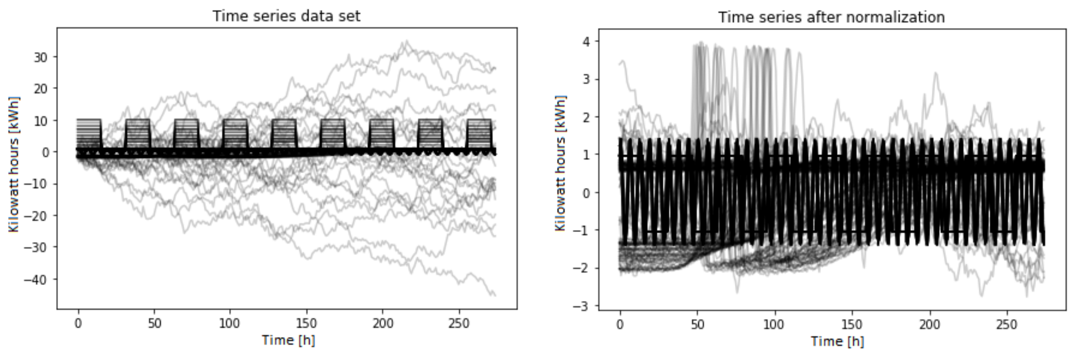


Figure 49: Time series data set from the Base case with 25 Random Walks included, before and after normalization.

The accuracy of the Silhouette algorithm was tested with the ten data sets in turn. The following results were obtained:

Table 7: Silhouette scores for all partitions of the data set with 25 Random Walks, between K=2 and K=9 clusters.

Number of clusters:	Average Silhouette score:
K = 2	0.287
K = 3	0.521
K = 4	0.668
K = 5	0.706
K = 6	0.708
K = 7	0.592
K = 8	0.578
K = 9	0.534

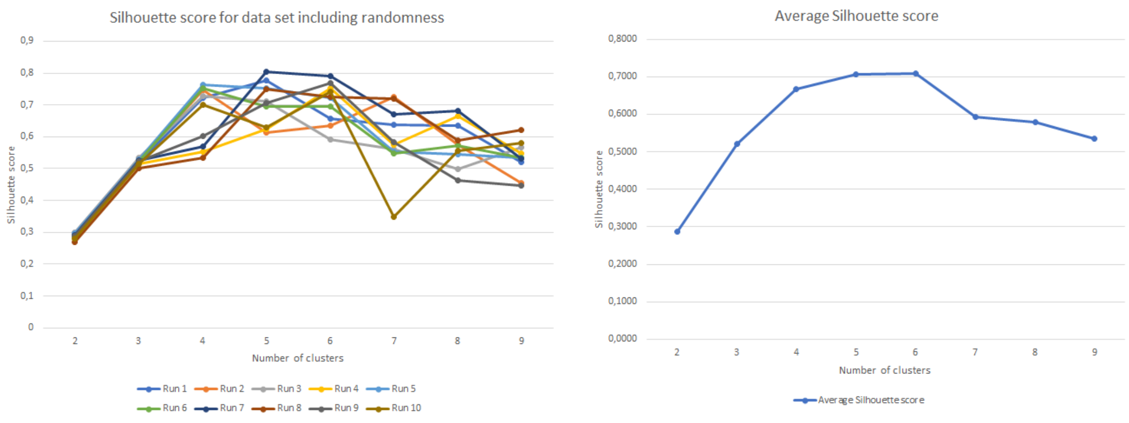


Figure 50: Silhouette scores for ten separate runs of partitions between K=2 and K=9 of the time series data set with 25 Random Walk time series, including a graph of the average Silhouette score across the ten runs.

The results in Table 7 and Figure 50 show that adding randomness to the data set may make the Silhouette and K-Shape algorithms behave unpredictably, i.e. the ten runs of the programme provided varying Silhouette scores. Averaging the Silhouette scores of the ten runs gave some indication of the range in which `K_optimal` may be found. Even though no clear conclusions can be drawn from the results given with ten runs in Table 7 and Figure 50, it appears likely that the Silhouette algorithm will determine an optimal partition of `K_optimal`=4, `K_optimal`=5 or `K_optimal`=6 when the data set described in the Base case includes 25 Random Walks.

The following figures provide illustrations of the clusters and cluster centroids found in data sets where `K_optimal`=4, `K_optimal`=5 or `K_optimal`=6.

In a data set where `K_optimal`=4, the partitions and cluster centroids were as shown in Figure 51.

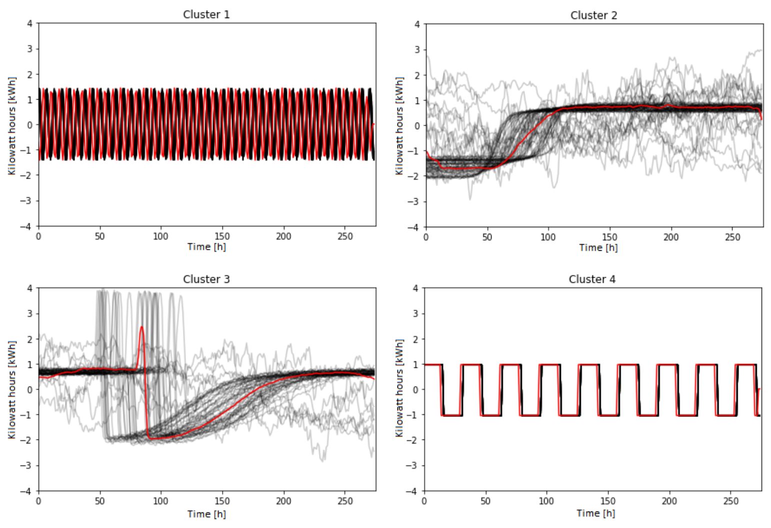


Figure 51: Base case data set including 25 random walks K-Shape output given an optimal Silhouette score for `K_optimal`=4 partitions.

In a data set where `K_optimal`=5, the partitions and cluster centroids were as shown in Figure 52.

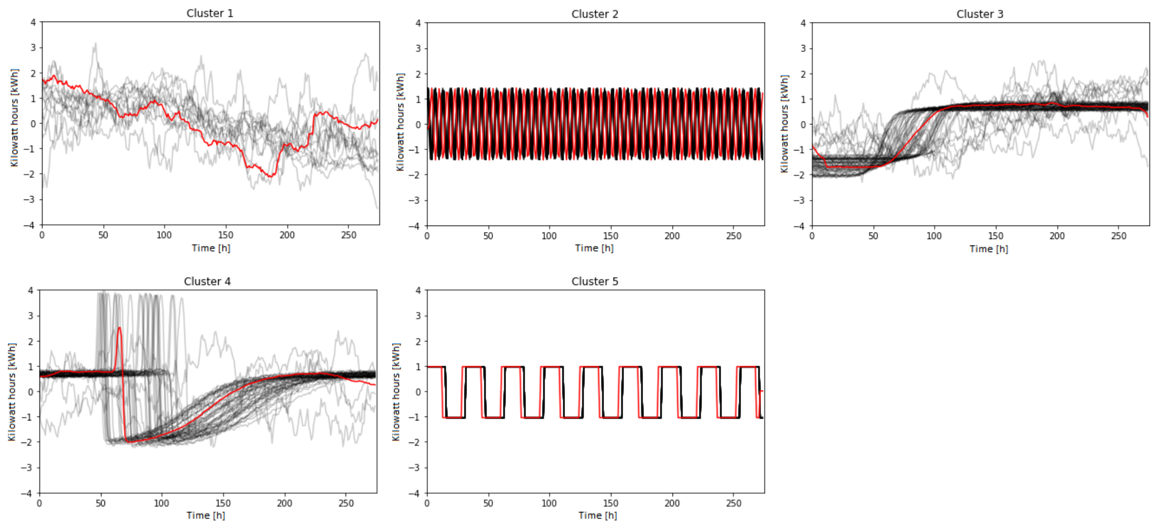


Figure 52: Base case data set including 25 random walks K-Shape output given an optimal Silhouette score for `K_optimal`=5 partitions.

In a data set where `K_optimal`=6, the partitions and cluster centroids were as shown in Figure 53.

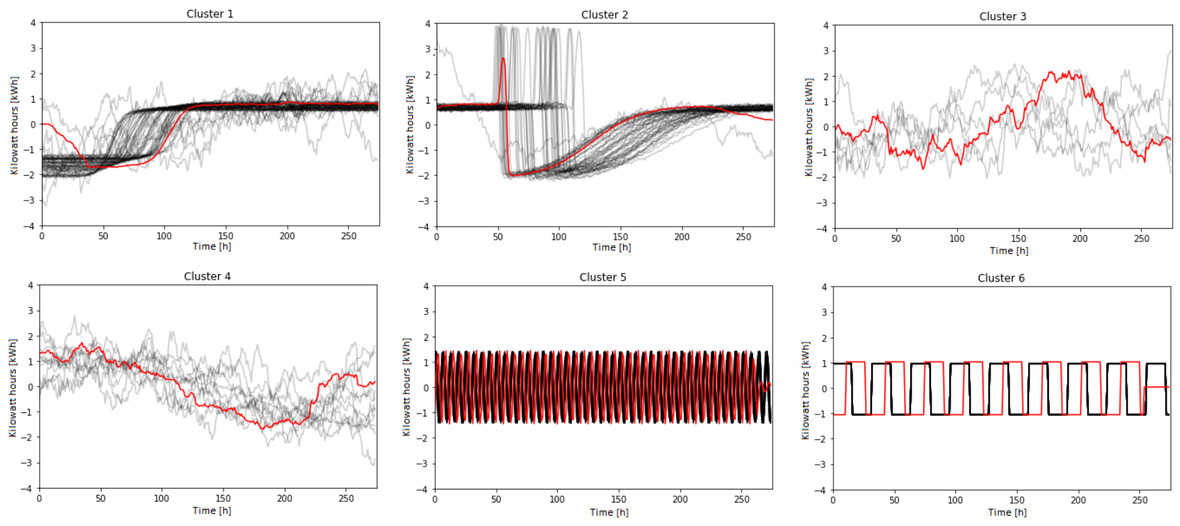


Figure 53: Base case data set including 25 random walks K-Shape output given an optimal Silhouette score for `K_optimal`=6 partitions.

It may be observed from Table 7 and Figure 50 that `K_optimal` generally achieves a lower Silhouette score than in Table 3 from the Base case. This indicates that cluster *compactness* in the data set as defined in Section 7.3.3 has been reduced by adding the Random Walks.

As is seen from Figure 51, Figure 52 and Figure 53, cluster centroids of the time series groups described in Section 9.2.1 have largely preserved their shape even though the data set now contains 25 Random Walks.

The algorithm stability for one specific data set at shown in this Section is tested in Section 9.3. In this Section `n_init` was selected to be 200.

From Figure 52 and Figure 53 it can be seen that even though the K-Shape algorithm is told to calculate more than four clusters, it does not separate all Random Walks from the existing time series groups. Due to this, cation should be taken when assigning `K_optimal` without due consideration for a data set which includes a significant amount of randomness, or where the optimal `K_optimal` is not visually obvious. Furthermore, this result indicates that *outlier analysis* may be useful (please refer to Section 7.5 for additional detail).

9.2.7 General conclusions from assessment of the developed model

From tests performed in Section 9.2.2 to Section 9.2.6, some general conclusions on the developed programme can be drawn:

- Given a data set of multiple, well-defined time series groups, the developed programme:
  - ☐ is able to correctly assess the number of distinct time series groups in the data set.
  - ☐ is able to correctly cluster the time series groups given the calculated optimal number of time series groups in the data set.
  - ☐ produces shape-based time series clusters and cluster centroids, i.e. the clusters and cluster centroids are amplitude, offset and phase/time invariant.
  - ☐ is not prone to significant inaccuracy given variations in group sparseness when calculations are performed within the K-Shape algorithm's stability zone.
  - ☐ is not prone to significant inaccuracy given variations in length of the time series when calculations are performed within the K-Shape algorithm's stability zone.
  - ☐ produces clusters and cluster centroids which appear to be stable over time if the predominant shape of the time series remains.
- Given a data set of multiple, well-defined time series groups, the `KShape` algorithm:
  - ☐ displays reduced stability given increased sparseness of a time series group in a data set (requires a higher value for `n_init`).
  - ☐ displays reduced stability given a set of shorter time series (requires a higher value for `n_init`).
  - ☐ stability does not seem to be effected by the `max_iter` variable for intervals in which `max_iter` was varied.

- Given sufficient randomness in a data set of multiple, well-defined time series groups, the programme will calculate varying optimal partitions depending on the nature of the added random data.
- In a data set of multiple, well-defined time series groups with added randomness ($\approx$ 10 % random data), the programme is able to produce cluster centroids with approximately the same shape as it would without the random data added.

9.3 Testing outlier analysis

In this Section the programme model, as shown in Figure 54 and given in Appendix IV, is tested with a known data set as a basis to determine whether outlier analysis may improve the performance of the customer segmentation programme.

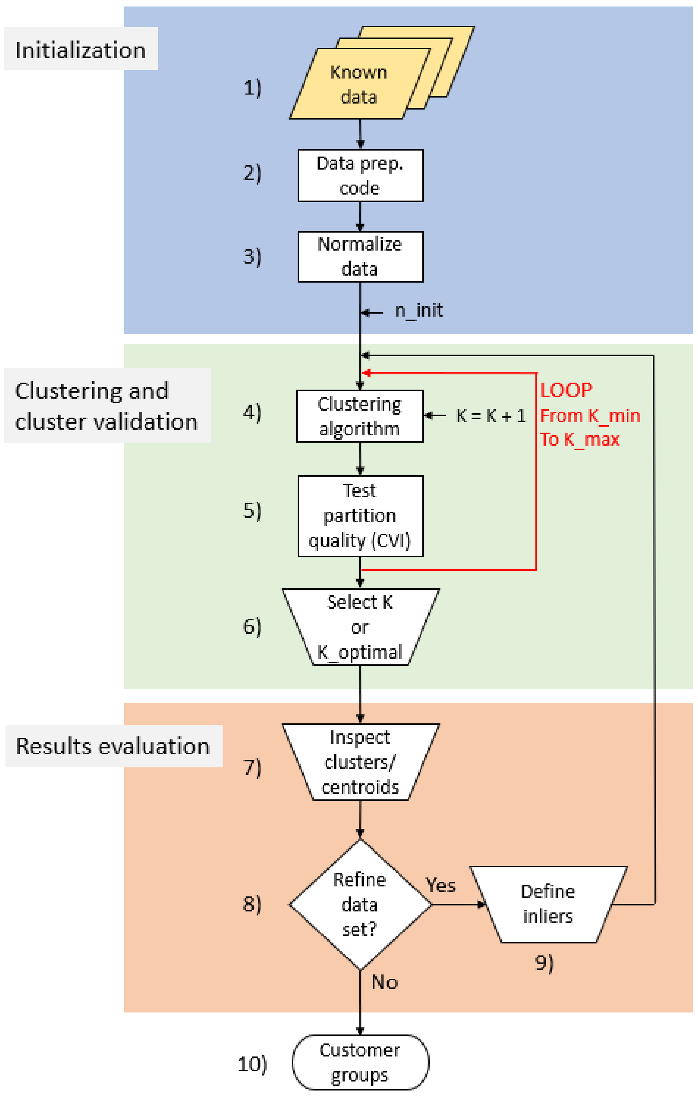


Figure 54: The programme used for testing outlier analysis.

As described in Section 7.5 three methods for outlier analysis are selected for the customer segmentation programme; OD1, OD2 and OD3.

9.3.1 Data set used in the outlier analysis

The data set used in this Section includes the "known data set" from Section 9.2.1, and 25 Random Walks, the same data set as used in Section 9.2.6. As may be observed in Section 9.2.6, the Silhouette and K-Shape algorithms regarded the Random Walks in some cases sufficiently similar to the existing time series groups, to group them into the same clusters. However, the Random Walks time series visually appear very different from the time series groups used in the Base case. It was therefore considered interesting to see if the different outlier analysis techniques described in Section 7.5 would be able to sort out the Random Walks given user input. This specific data set before and after normalization is shown in Figure 55.

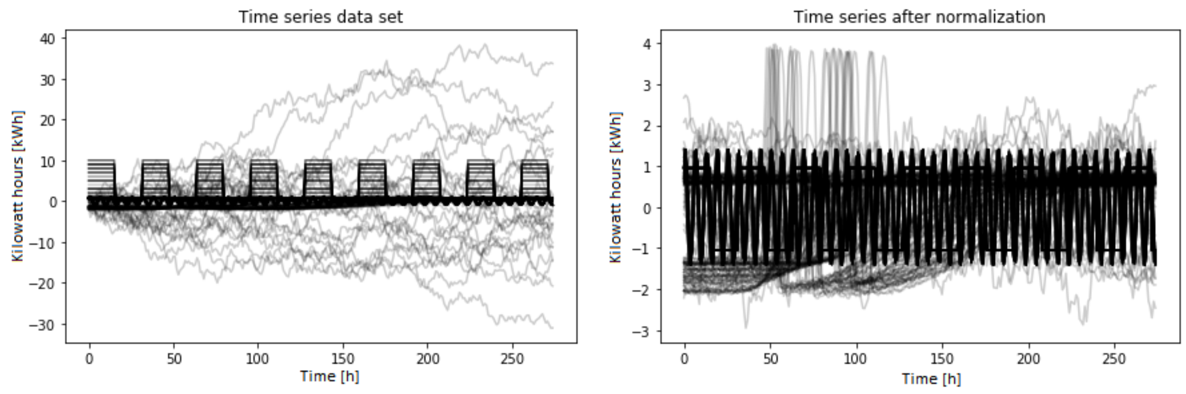


Figure 55: The data set used to test outlier analysis, before and after normalization.

After testing the K-Shape algorithm stability limits as explained in Section 9.3.3, the following Silhouette scores for partitions between K=2 and K=9 were obtained, when the K-Shape algorithm was used in its stable zone. The Silhouette scores are shown in Table 8.

Table 8: Silhouette scores for all partitions of the data set with 25 Random Walks, used for testing the effect of outlier analysis on developed programme, between K=2 and K=9 clusters.

Number of clusters:	Silhouette score:
K = 2	0.297
K = 3	0.534
K = 4	0.716
K = 5	0.748
K = 6	0.631
K = 7	0.759
K = 8	0.629
K = 9	0.621

As may be seen from Table 8 `K_optimal`=7 with a Silhouette score of 0.759.

9.3.2 Outlier analysis

Outliers were sorted out of the data set in an iterative process between steps 4) to 9) in Figure 28. The results are shown in Figure 56.

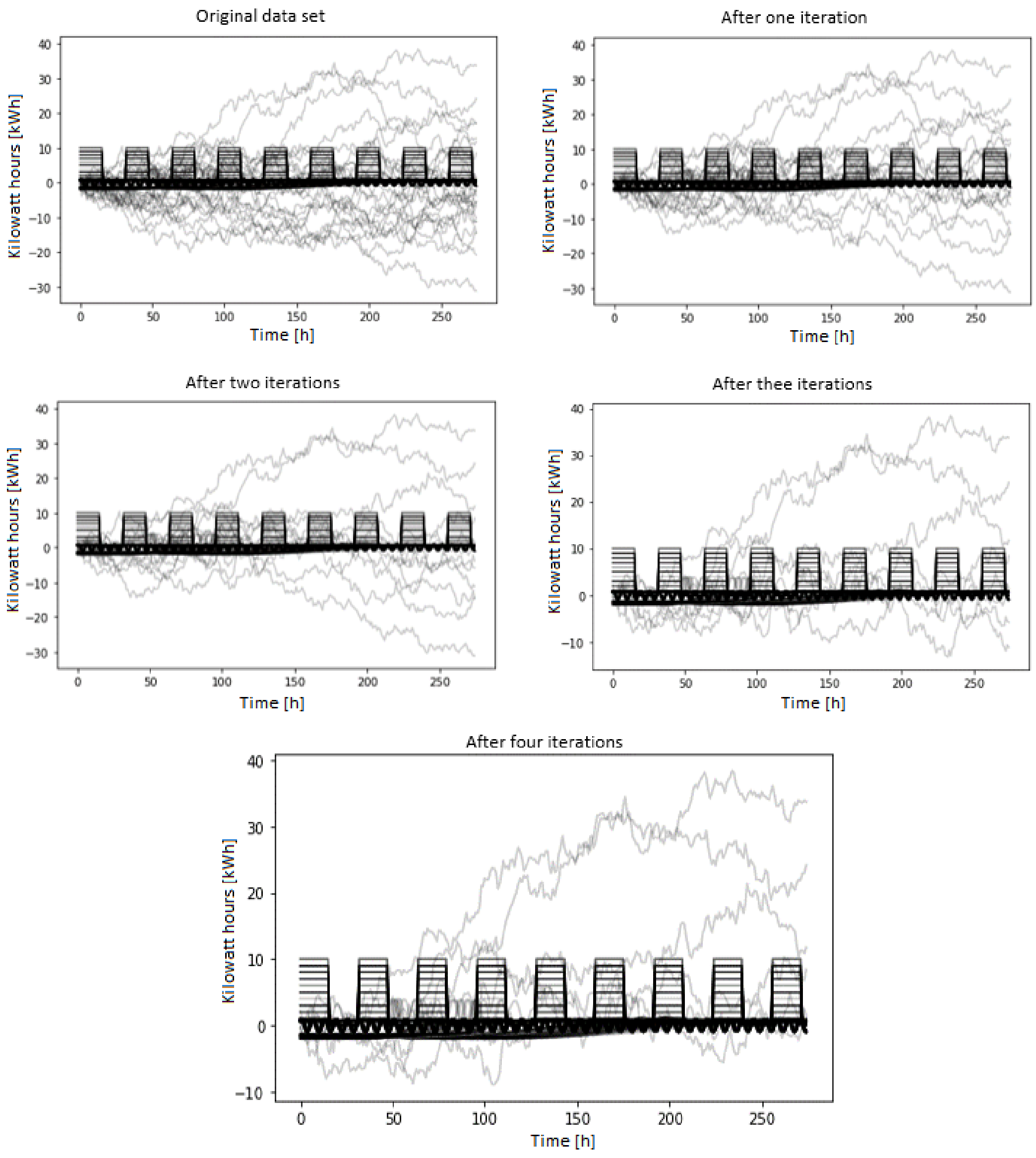


Figure 56: Inlier data set throughout an outlier analysis of a data set.

The outlier selection methods which were found to provide the best results, and were used for this outlier analysis in this case were OD1 and OD3. The individual time series groups were not homogeneous enough for OD2 to provide any benefits, i.e. each time series group was more "similar" to the outliers than to each other. OD3 also proved only moderately successful, as within the groups imported from `tslearn` (Cluster 1 and 3 in the Base case, Figure 36) the DTW difference between the cluster centroids and the outliers (Random Walks) was towards the end of the iteration process less than the DTW difference between the cluster centroids and some of the time series, which in the Base case were allocated to the clusters.

It would be possible to remove all Random Walks from the data set given many iterations, but four iterations proved enough to illustrate the effect of outlier analysis on cluster partition quality.

Over the four outlier analysis iterations, the inlier and outlier data set changed as shown in Figure 56 and Figure 57.

It can be seen in Figure 56 and Figure 57 that outliers are more easily filtered out in the beginning of the outlier analysis, as these are the time series which fit less well within the pre-determined time series groups from the Base case.

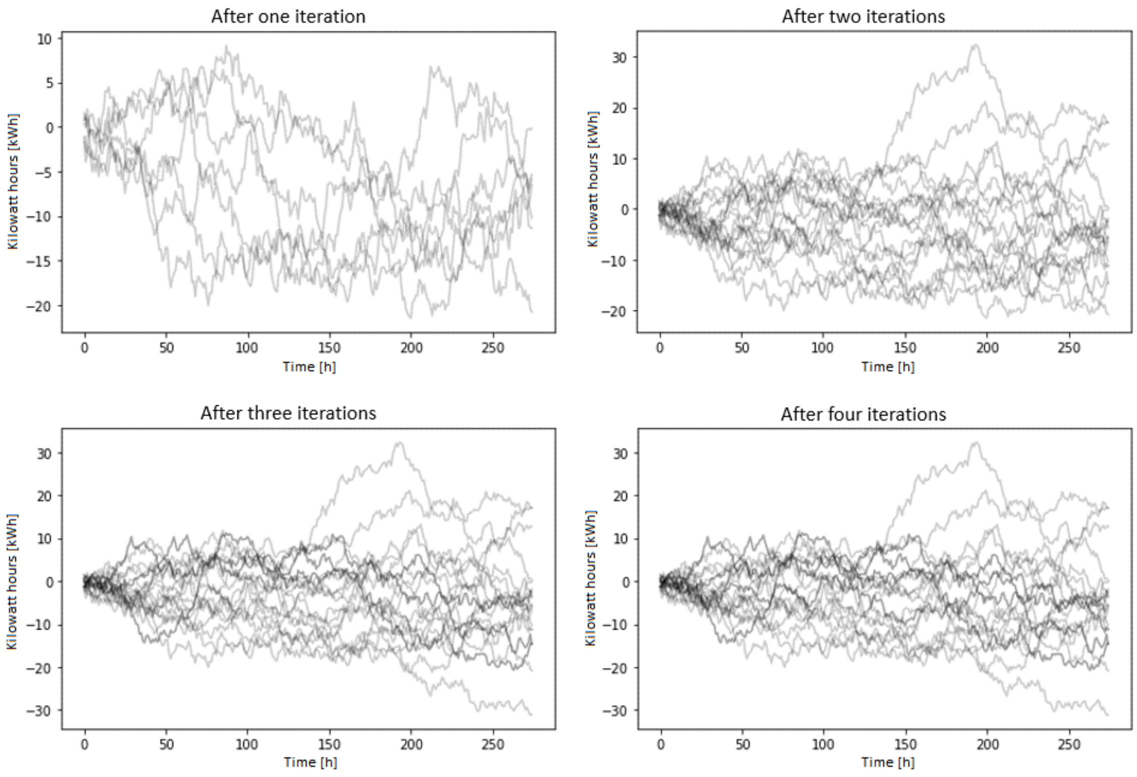


Figure 57: Outlier data set throughout an outlier analysis of a data set.

The Silhouette scores of each partition `K` of the inlier data set before any, and after four iterations with outlier analysis are plotted in Figure 58 against the Silhouette scores for partitions `K` of the Base case data set.

Figure 58 shows that the Silhouette score for the data set after outlier analysis has improved for `K` between `K`=2 and `6`, and appears to resemble the Silhouette scores of the Base case to a higher degree. `K_optimal` has also become more distinct after outlier analysis.

After four iterations, six of the 25 Random Walks still remained in the data set, which accounts for around 24% of the Random Walks originally in the data set. It is expected that the remaining Random Walks are the reason why the Silhouette score after four iterations do not perfectly match the scores from the Base case.

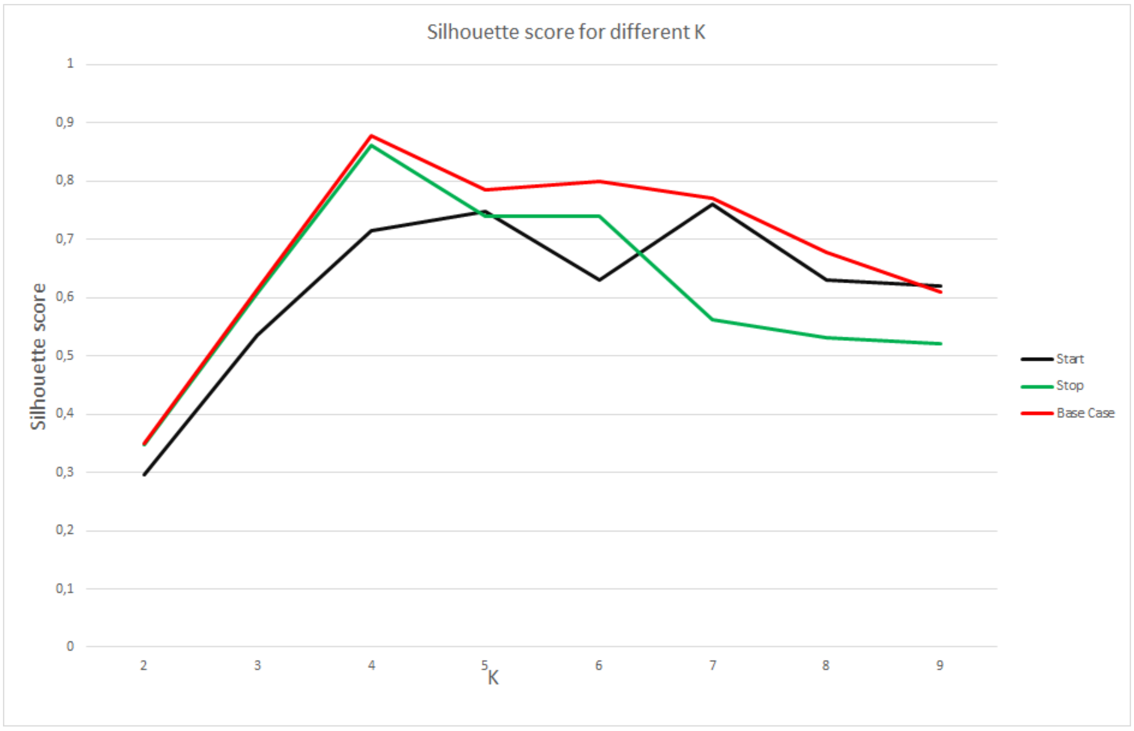


Figure 58: Comparison of the Silhouette scores of the Base case (red), Base case + 25 Random Walks (black), and Base case + 25 Random Walks with four iterations of outlier analysis (green).

9.3.3 Algorithm stability

The stability of the `tslearn` `KShape` algorithm was tested for the data set described in Section 9.3.1. To test the algorithm stability the same methodology as in Section 9.2.2 was applied. `max_iter` was varied between 100 and 1000, and `n_init` was varied between 2 and 300 until stable Silhouette scores for the different partitions `K`=2 to `K`=9 was achieved. The stability regions for the data set in Section 9.3.1 is illustrated in Figure 59. `n_init` is once again shown to be the critical variable for algorithm stability. The algorithm is shown to produce stable results for all `K` between 2 and 9 for `n_init`≈200 and above. It is clear from comparing the results in Figure 59 to the results in Section 9.2.2 that the K-Shape algorithm becomes less stable when random time series are added. There may also be a slight destabilising effect of simply adding more time series to the data set.

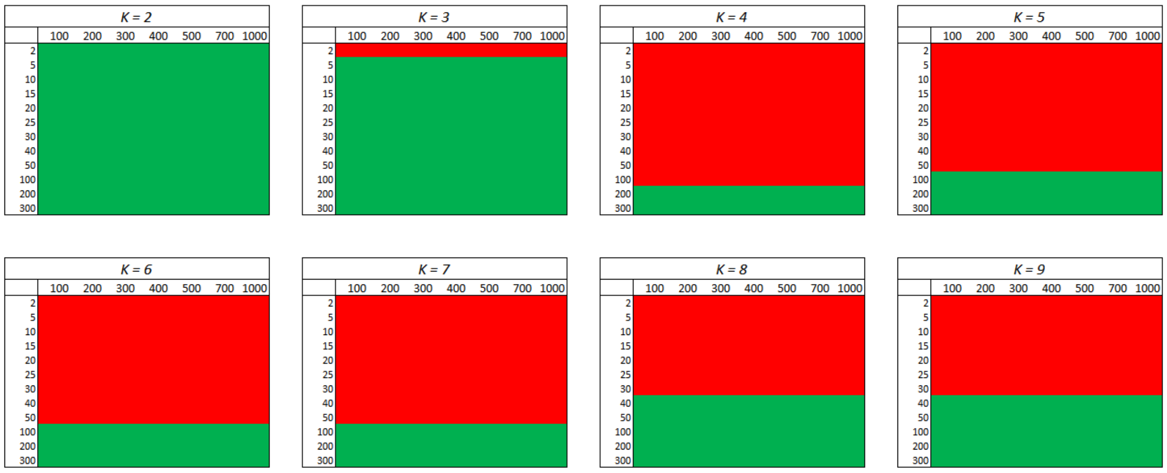


Figure 59: K-Shape algorithm stability before outlier analysis.

After outlier analysis it can be seen, by comparing Figure 59 and Figure 60, that removing outliers has made the `KShape` algorithm stable for more values of `K`, and less iterations of `n_init`. From this it appears that the code becomes more efficient with outlier analysis. However, the highest `n_init` stability boundary remains ≈200 and above iterations. This is likely because not all Random Walks were removed in the outlier analysis (six of the 25 Random Walks remained).

By comparing Figure 58, Figure 37 and Figure 60 it may also be observed that the K-Shape algorithm becomes unstable for the data set after outlier analysis for the same values of `K` that the Silhouette score diverges from the Silhouette score of the Base case.

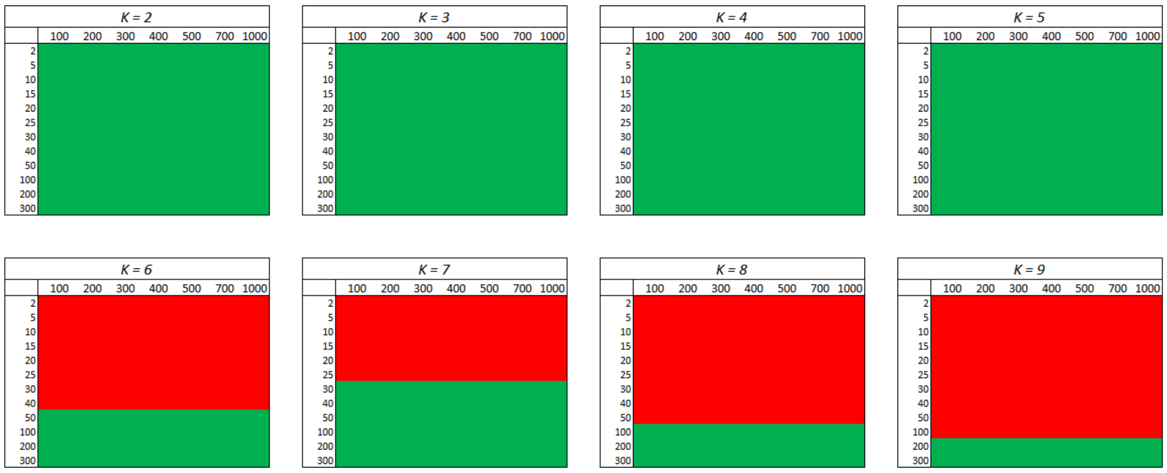


Figure 60: K-Shape algorithm stability after outlier analysis.

9.3.4 General observations on outlier analysis methods

The general observations on OD1-OD3, as discovered in this Section are:

- **OD1**:
  - ☐ works well if the K-Shape algorithm is able to "sort out" outliers into their own clusters, perhaps by increasing `K`.
  - ☐ does not work well if outliers are "similar" to inliers according to the Shape-based distance, even though they may visually appear to be different.
- **OD2**:
  - ☐ works well if the DTW difference between the cluster centroid and the "inliers" of a cluster is smaller than the DTW difference between the cluster centroid and the outliers.
  - ☐ does not work well for the opposite case, where the DTW difference between cluster centroid and outlier is smaller than the DTW difference between cluster centroid and inlier.
- **OD3**:
  - ☐ works well for largely homogeneous data sets where outliers generally have a large DTW difference from all inliers.
  - ☐ does not work well when the DTW difference between clusters is larger than the DTW difference between clusters and outliers.

The strengths and weaknesses of using outlier analysis as proposed in the customer segmentation programme are listed in Table 9.

Table 9: The strengths and weaknesses of the outlier analysis methods used in the customer segmentation programme (OD1, OD2 and OD3).

Strengths:	**Weaknesses:**
Outlier analysis makes the K-Shape algorithm more stable, improving the runtime of the algorithm.	The user of the programme has to visually or in other ways be able to distinguish between outliers and inliers.
Gives the user of the customer segmentation programme the option to remove time series from the data set that the he/she wishes to not include in the analysis.	The outlier selection is neither considered smart nor automatic.
	Using outlier analysis for AMS-data sets would mean actually removing some AMS-data from the customer segmentation programme, which means that valuable and/or interesting data may be removed from the analysis.

Based on conclusions given in Table 9, outlier analysis is, as mentioned in Section 7.5 included in the customer segmentation programme as an option, *not* a necessity.

10 Assessing the developed customer segmentation programme using AMS-data

In this Section the developed customer segmentation programme, as shown in Figure 28 in Section 8 and provided in Appendix IV, is assessed using two different AMS-data sets. Firstly, an AMS-data set without the current standard DSO customer segment information included, to observe the performance of the developed programme applied to AMS-data. Secondly, the performance of the programme is compared with the current standard DSO customer segmentation method.

10.1 Performance of developed programme on AMS-data

The data analysed in this Section is actual AMS time series data obtained from Lyse Elnett. This data is considered power-sensitive information, and can therefore not be shared in this report. For this reason, the exact reproducibility of the analysis can not be verified by a third party.

The AMS-data set consisted of 378 customer consumption measurements in kWh, taken on an hourly basis. Different time-spans to be analysed by the developed programme were considered, from daily to monthly. In this respect several factors were evaluated:

- The possibility to maximise the number of customers in the data set. Not all customers had complete measurements for each day.
- The possibility to capture periodic patterns in the data set.
- The stability of the `KShape` algorithm for the data set. When a large value for `n_init` is needed, the runtime of the algorithm increases proportionally (see Section 7.3.1).
- The length (number of measurements per time series) of the data set. A large data set would be more computationally demanding.

Following an assessment, a weekly measurement period between the 13/11/2017 and the 20/11/2017 was chosen. The main reason for selecting this week was because in this week there existed measurements from all 378 customers. Secondly, a week was considered an adequate time-span for capturing some of the more pronounced periodic consumption patterns. A week was also considered to be able to provide sufficient stability for the `KShape` algorithm. Finally, a week was expected to provide an acceptable runtime for the programme.

The analysis was performed with the programme as described in Section 8, Figure 28. The data set before and after normalization had the appearance as shown in Figure 61.

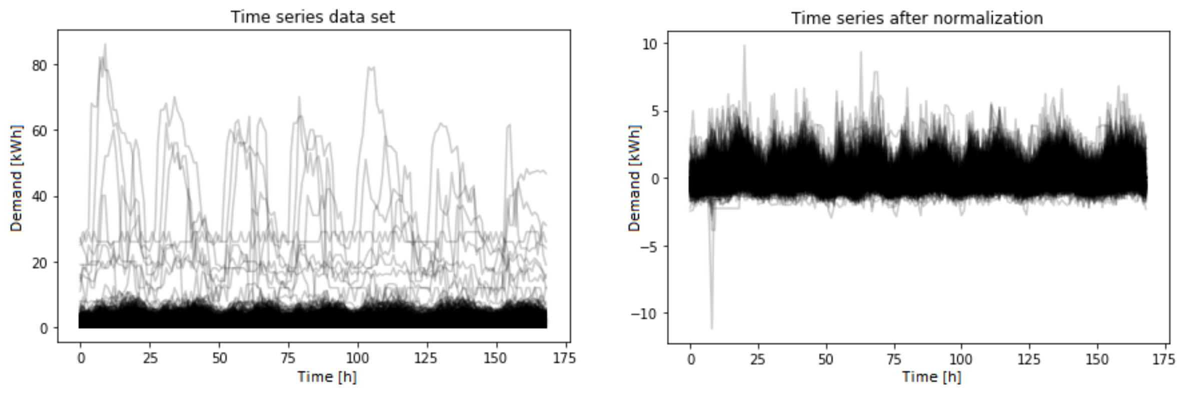


Figure 61: AMS-data set before and after normalization.

The data set did not include any information on which or how many of the 38 standard DSO customer groups shown in Figure 11 existed in the AMS-data set, i.e. this information was unknown. The Silhouette algorithm was instructed to test each partition of the normalized data set between `K`=2 and `K`=15, and produced the results shown in Table 10 and Figure 62. When finding the stability zone of the `KShape` algorithm for the AMS-data set, each `K` is tested for an increasing `n_init` until the stability zone is reached. Given that the stability region of the AMS-data set is ≈10000, determining the stability for a `K_max`>15 would result in runtime which was considered incompatible with the work for this report. Therefore, `K_max` was chosen to be 15.

For various combinations of `K` and `n_init`, the `tslearn KShape` algorithm found that the optimal clustering produced at least one empty cluster, which had not been experienced with any other data sets tested by the programme. This produced an error message in the programme code, and is denoted Not a Number (NaN) in Table 10. In Figure 62 these error messages are shown with a red dot, while results with a calculated value are shown with green dots. These irregularities are discussed below.

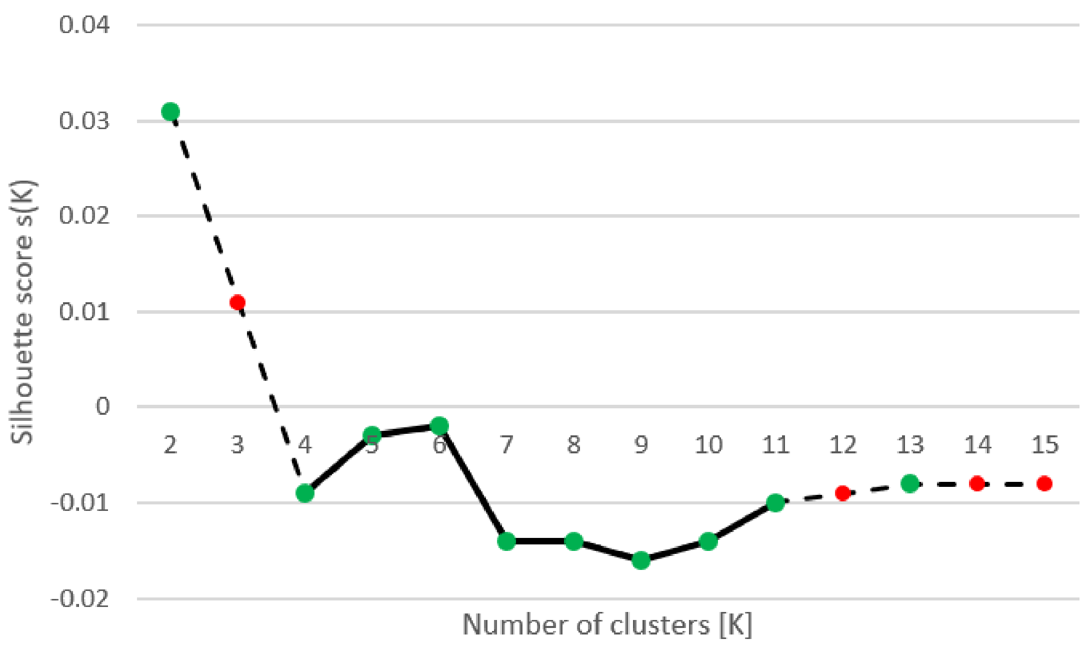


Figure 62: Silhouette scores for the AMS-data set.

Table 10: Silhouette scores for all partitions of the AMS-data set, between K=2 and K=15 clusters.

Number of clusters:	Average Silhouette score:
K = 2	0.031
K = 3	NaN
K = 4	-0.009
K = 5	-0.003
K = 6	-0.002
K = 7	-0.014
K = 8	-0.014
K = 9	-0.016
K = 10	-0.014
K = 11	-0.010
K = 12	NaN
K = 13	-0.008
K = 14	NaN
K = 15	NaN

As may be seen from Table 10 and Figure 62, the optimal Silhouette score was achieved for K=2. When instructed to produce K=2 for an n_init within the KShape algorithm stability zone, the clusters and centroids shown in Figure 63 were produced.

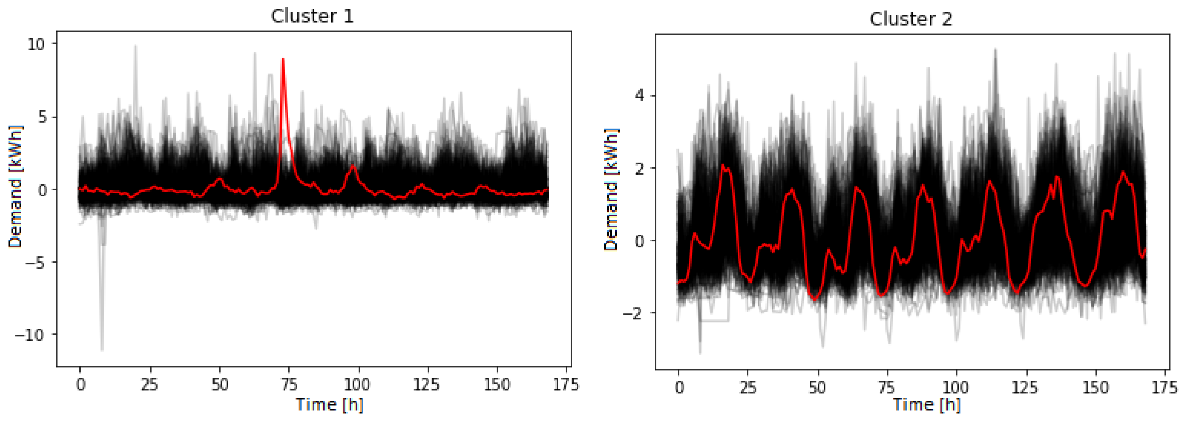


Figure 63: Result of using K-Shape clustering on the AMS-data set, with `K_optimal`=2.

The K-Shape algorithm seems to initially segment the data set based on a perceived "period" in the consumption pattern. Cluster 2 in Figure 63 displays a clear 24-hour repetitive pattern characterised by a small peak in the morning, a larger peak in the afternoon and a clear dip during night time. Cluster 1 does not display any clear patterns, but is characterised by a weekly peak in consumption. Note that the K-Shape algorithm is time invariant, and the different peaks seem to occur at different times of the week.

In order to understand the composition of the data set more completely, an outlier analysis was performed. Additionally, the error messages produced by the `KShape` algorithm seemed to indicate that some "irregular" data was present in the data set. During the analysis it was discovered that the AMS-data set contained several "flat" time series. Using OD2 and a DTW threshold of 12, the outliers shown in Figure 64 were separated out.

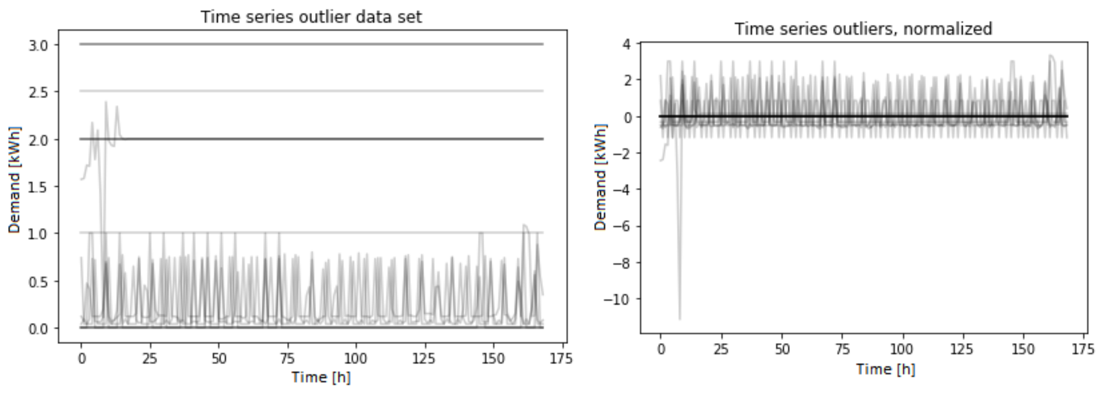


Figure 64: Detected outliers using OD2 with a DTW threshold of 12, for the AMS-data set.

Of the 18 selected outliers, 12 (67%) of them were flat time series. Figure 65 shows a histogram of the extra-cluster DTW distribution (DTW threshold included) between the centroids of Cluster 1 and Cluster 2, and the entire data set. OD2 is explained in further detail in Section 7.5.2. This DTW threshold (the red line in Figure 65) was found to select the highest number of flat time series, while selecting the least number of other time series.

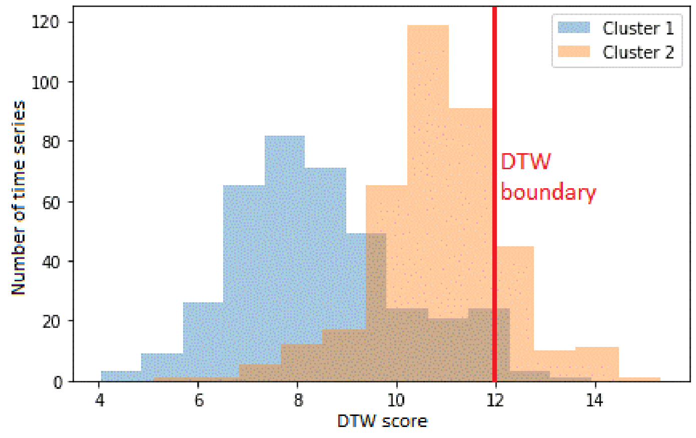


Figure 65: Histogram of extra-cluster DTW score for the two generated clusters.

After isolating the outliers shown in Figure 64, the inlier data set included 360 time series and had the appearance as shown in Figure 66.

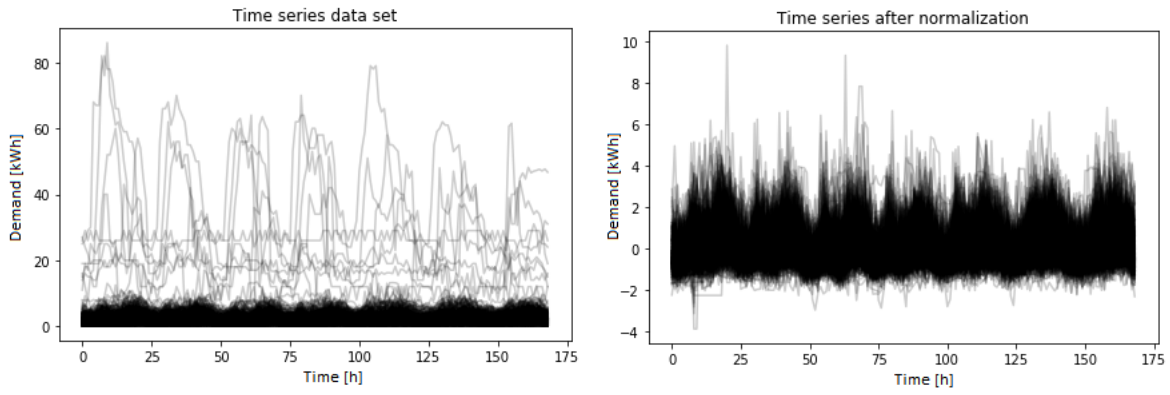


Figure 66: Inlier AMS-data set, before and after normalization.

The Silhouette algorithm tested each partition of the normalized inlier data set from Figure 66, between K=2 and K=15, and gave the results shown in Table 11 and Figure 67.

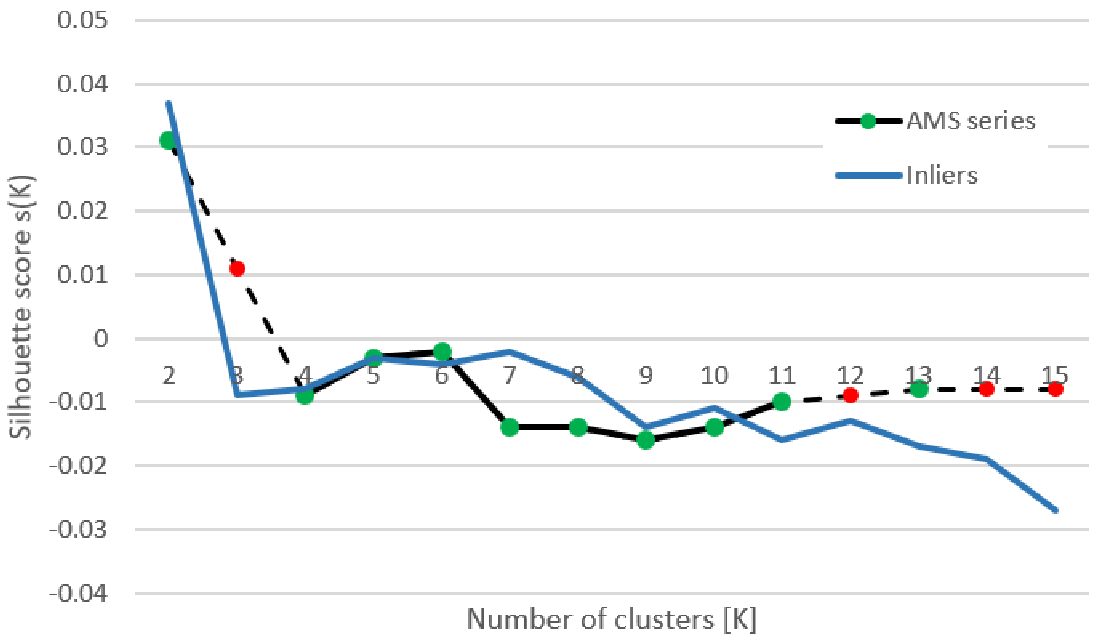


Figure 67: Silhouette scores for the inlier AMS-data set.

Table 11: Silhouette scores for all partitions of the inlier AMS-data set, between K=2 and K=15 clusters.

Number of clusters:	Average Silhouette score:
K = 2	0.037
K = 3	-0.009
K = 4	-0.008
K = 5	-0.003
K = 6	-0.004
K = 7	-0.002
K = 8	-0.006
K = 9	-0.014
K = 10	-0.011
K = 11	-0.016
K = 12	-0.013
K = 13	-0.017
K = 14	-0.019
K = 15	-0.027

After the outliers were removed, the problem of the KShape algorithm producing empty clusters disappeared. Additionally, the KShape algorithm became stable for lower values of n_init.

As mentioned in Section 7.4, the Silhouette score should be used as a guideline for selecting a K, as K should not necessarily always be selected to be K_optimal. For this reason, it was considered interesting to test a K≠K_optimal. In Table 11 the second highest Silhouette score is achieved for K=7. Therefore K=7 was selected as input to the clustering algorithm (Step 6) in Figure 28).

When instructed to produce K=7 for an n_init within the KShape algorithm stability zone, the clusters and centroids shown in Figure 68 were produced.

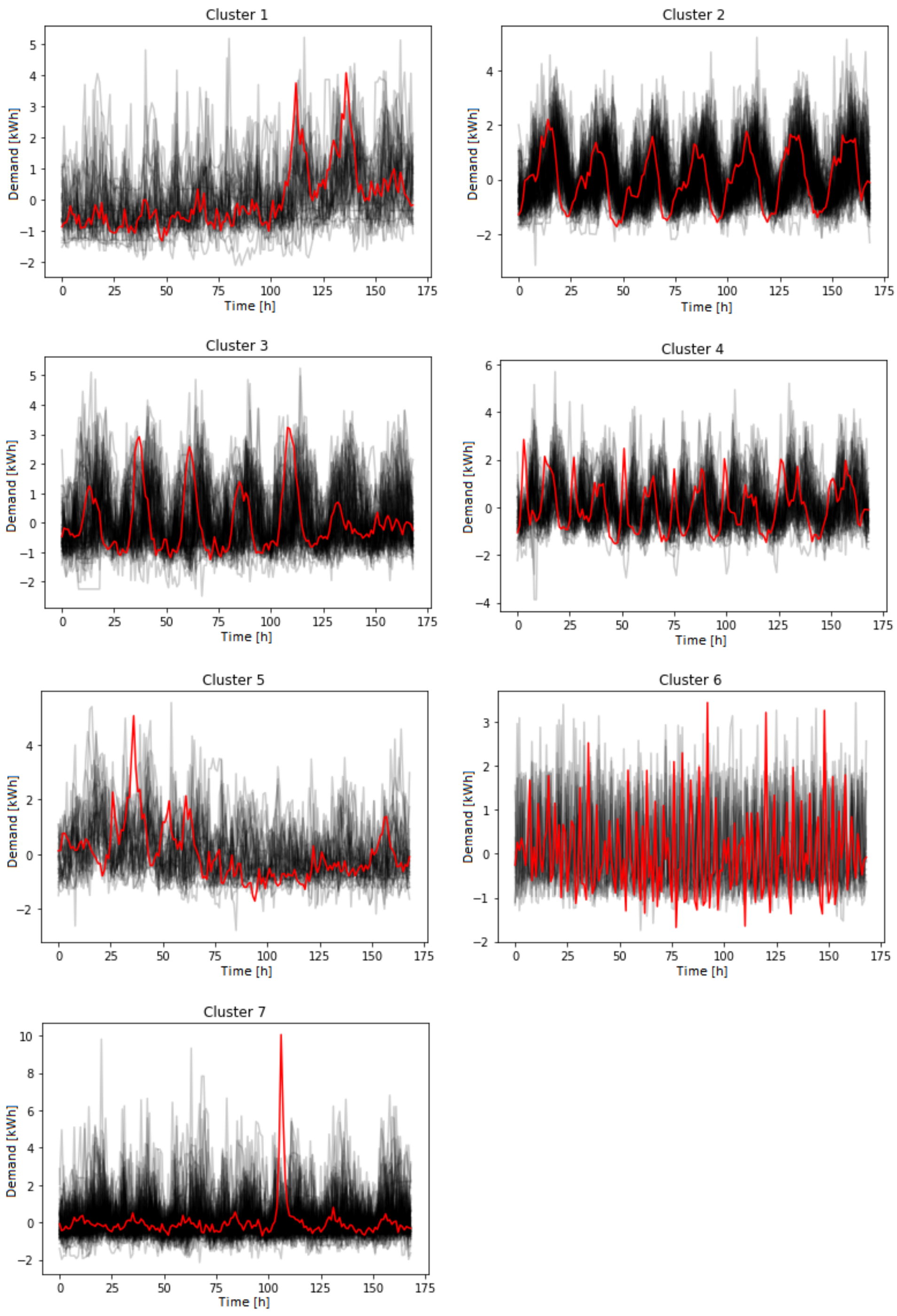


Figure 68: Result of using K-Shape clustering on the inlier AMS-data set, with K_optimal=7.

The clusters in Figure 68 are visually distinct from each other, and some with clear periodic patterns. Cluster 7 in Figure 68 may be recognized as having the characteristic weekly spike seen in Cluster 1, Figure 63. Equally, Cluster 2 in Figure 68 has the recognizable characteristics of a consumption spike in the morning and afternoon, and a dip during night time, when compared to Cluster 2 in Figure 68. These two clusters became the dominant shapes when the `KShape` algorithm was instructed to produce `K`=2 clusters (Figure 63), because they are the most dense clusters in Figure 68. This may also be seen in Table 12.

The size of each generated cluster, as shown in Table 12 demonstrates the *substantiality* of the clustering performed by the programme. Cluster substantiality is a condition for cluster usefulness described in Section 7.3.3. None of the clusters include less than 10 time series, and are probably large enough for some insight to be gained by analysing them. Such an analysis could be done given that information on standard DSO customer segments also were provided for this AMS-data set. This type of analysis is mentioned in Section 12.2 as a potential further work for this report.

Table 12: The density of the clusters produced from the inlier data set given `K`=7.

Cluster:	**Number of time series:**
1	31
2	87
3	63
4	49
5	29
6	14
7	87

Algorithm stability for the AMS-data set

As was concluded in Section 9, the variable `max_iter` did not affect the stability of the `KShape` algorithm. Due to this, `max_iter` was in this Section kept constant at `tslearn` default (`max_iter`=100), while `n_init` was varied.

For the initial/original AMS-data set, the stability boundaries for different `K` shown in Figure 69 were found. As may be seen in Figure 69, the algorithm becomes stable for `K` between 2 and 15 for `n_init`≈10000.

For the inlier data set, the stability boundaries for different `K` shown in Figure 70 were found. It may be seen by comparing Figure 69 and Figure 70 that the algorithm has become more stable for all values of `K` tested with maximum `n_init`≈7000, following the outlier analysis.

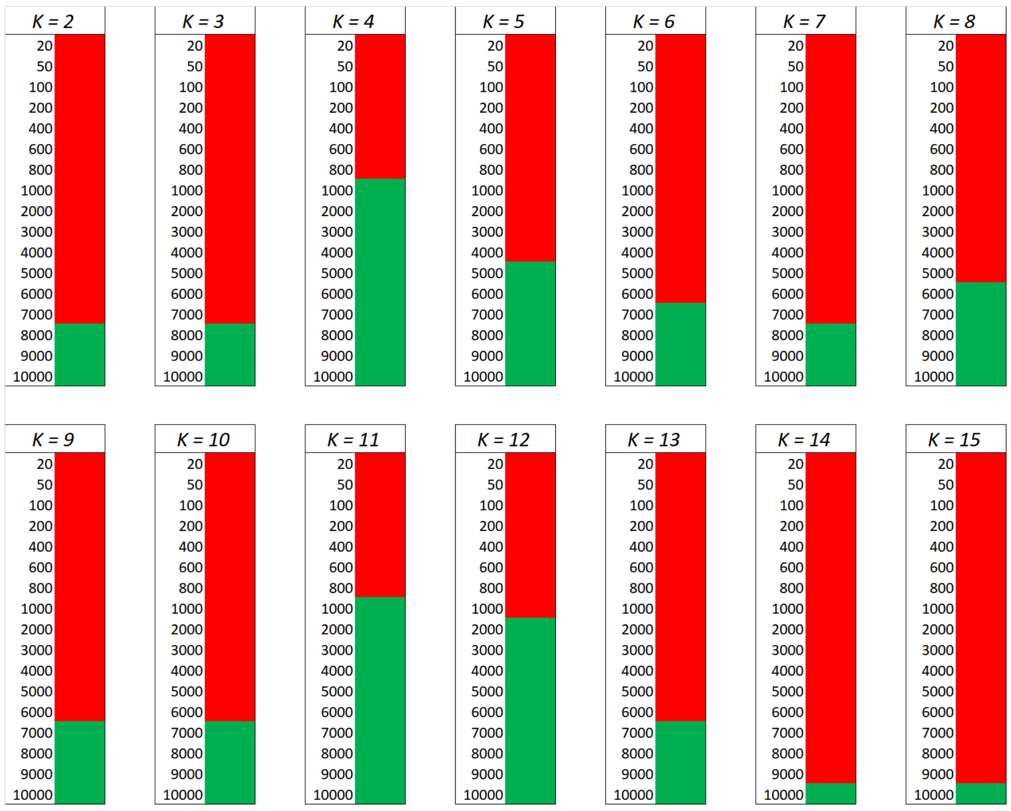


Figure 69: Stability of the KShape algorithm for the original AMS-data set, with maximum n_init≈10000.

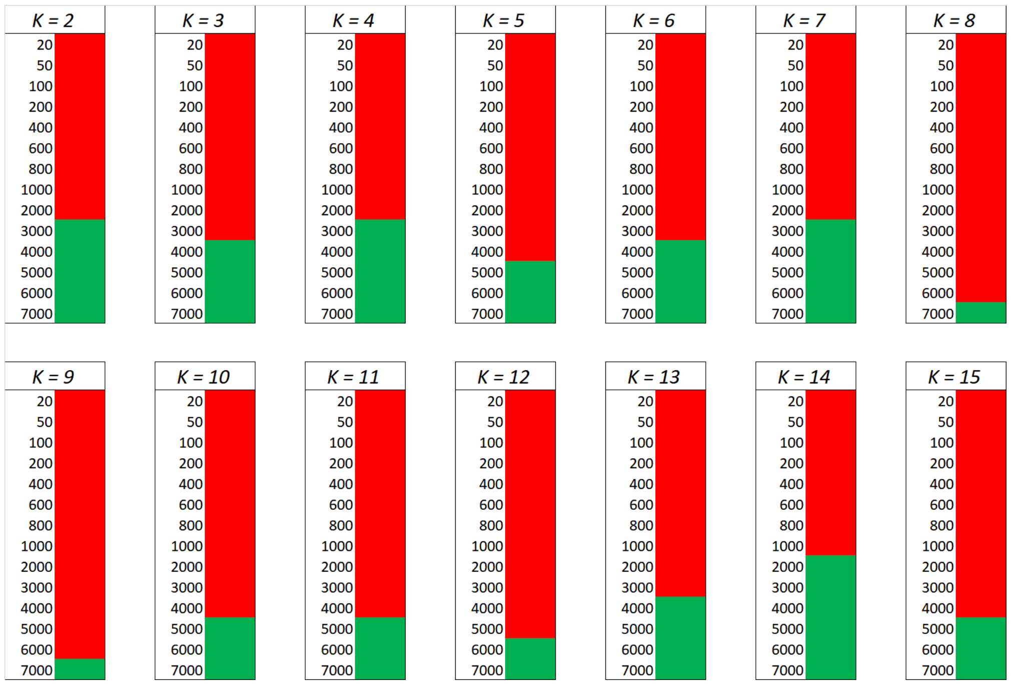


Figure 70: Stability of the KShape algorithm for the inlier AMS-data set, with maximum n_init≈7000.

10.1.1 Conclusions from analysis with AMS-data

The following conclusions can be drawn from the analysis of the AMS-data set:

- The AMS-data set achieved overall much lower Silhouette scores for each `K`, when compared to analysis on the known data set. This indicates that it was much harder to achieve similar cluster compactness with the given AMS-data.
- The highest cluster compactness, i.e. highest Silhouette score, was achieved for `K`=2 in both the initial/original and inlier AMS-data sets. Separating the time series with repetitive 24-hour patterns from those without a clear 24-hour repetitive pattern seems to yield the highest cluster compactness. However, selecting `K`=2 may not lead to the preferred result when segmenting customers from AMS-data.
- When inspecting the `K` for which the second-highest Silhouette score is achieved in the inlier data set (`K`=7), the programme produces substantial clusters. This holds true for `K`=2 as well.
- Flat time series were shown to disturb the `KShape` algorithm, which resulted in it producing empty clusters (NaN).
- After the flat time series were removed from the data set, the stability of the `KShape` algorithm for all `K` improved.
- Outlier analysis was shown to be useful when analysing AMS-data.

10.2 Comparison of developed segmentation programme to standard DSO customer segmentation

To perform a comparison of the developed customer segmentation programme to the standard DSO method described in Section 6.1, Lyse Elnett provided an additional AMS-data set towards the completion of the report, which included information on the customer segment that each time series belonged to. This data set is not the data set analysed in Section 10.1, and differs from the data set in Section 10.1 in that it includes less (103, not 378) and other customer IDs, in addition to the inclusion of customer segment information. As in Section 10.1, the provided AMS-data is power-sensitive information and can not be included in this report.

As stated in Section 5.4.4, the goal of customer segmentation analysis in this report is to divide the customers into groups of similar consumption patterns. Due to this, a *shape-based* clustering approach was selected. The approach was motivated by the potential for DSOs to, for example, use typical consumption profiles to assess whether a transformer has the necessary capacity for affiliation of new customers, or for design of customer-specific tariff systems. However, given that the approach is shape-based, it does not consider amplitude and offset of the customers, i.e. whether the customer has a large or small electric consumption need. Furthermore, the shape-based approach does not differentiate between customers, e.g. whether they are household or industrial customers.

The fundamental principle of the standard DSO customer segmentation method is based on differentiating between types of customers. This approach is likely to be more amplitude and offset sensitive than the shape-based approach.

The comparison was carried out by considering the customer segments proposed by Lyse Elnett to be clusters, calculating the CVI (i.e. Silhouette score) of this partition, and comparing it to the CVIs of the partitions created by the developed customer segmentation programme.

10.2.1 Current standard DSO customer segmentation method

As explained in Table 14, when a new customer connects to the distribution grid, they will be assigned by the DSO to one of the segment categories 1-36 listed in Table 14.

As mentioned above, the AMS-data set consisted of 103 customer measurement time series, which included the customer segments shown in Table 13. As seen from Table 13, the largest segment in the data set is the “Household” segment, with 76 customers. “Turnover and operation of real estate” is the second largest segment, with 13 customers. There are three categories with only one customer. These are the “Extraction of crude oil and natural gas”, “Construction and structural business” and the “Other transport and storage” categories.

Table 13: Customer segments available in the Lyse Elnett AMS-data set.

Code:	**Customer consumption segment:**	**Number of customers:**
3	Extraction of crude oil and natural gas	1
18	Construction and structural business	1
19	Wholesale trade, repair of motor vehicles	3
21	Other transport and storage	1
23	Accommodation and serving business	4
26	Turnover and operation of real estate	13
27	Professional, scientific and technical services	2
31	Health and social services	2
35	Household	76

Table 14: Current standard method of customer segmentation for Norwegian DSOs.

Code:	**Customer segment:**
1	Agriculture, forestry, fishing
1A	Greenhouse
2	Mining
3	Extraction of crude oil and natural gas
4	Services in connection with extraction of crude oil and natural gas
5	Production of pulp, paper and cardboard
6	Production of chemical raw materials
7	Production of iron and steel
8	Production of ferroalloys
9	Production of primary aluminium
10	Production of other non-ferrous metals
11	Food and nutrition industry
12	Refineries
13	Other industries
14	Production and distribution of electricity
15	Production and distribution gas through gas distribution network
16	District heating
17	Water supply, sewerage, sanitation
18	Construction and structural business
19	Wholesale trade, repair of motor vehicles
20	Railway, tramway and suburban train
21	Other transport and storage
22	Postal and distribution activities
23	Accommodation and serving business
24	Information and communication
25	Financial services, insurance and pension funds
26	Turnover and operation of real estate
27	Professional, scientific and technical services
28	Business services
29	Public administration and defence
29A	Street and road lights
30	Education
31	Health and social services
32	Artistic activities, libraries etc., sports and leisure
33	Activities in member organizations
34	Other services
35	Households
36	Cottages and holiday homes

As done is section 10.1, a a time period of a week was chosen for the analysis. The chosen section of the AMS-data set included customer segment information between the 14/10/2018 to the 21/10/2018. This week appeared to be the week with the most complete data. The selected AMS-data set before and after normalization, is shown in Figure 71.

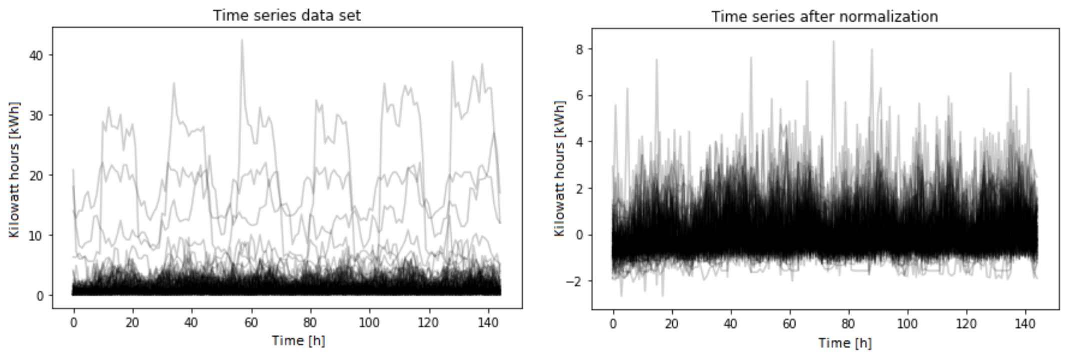


Figure 71: AMS-data set with customer segment information, before and after normalization between 14/10/2018 and the 21/10/2018.

Plotting the different customer segments according to Table 13 produces the partitions shown in Figure 72.

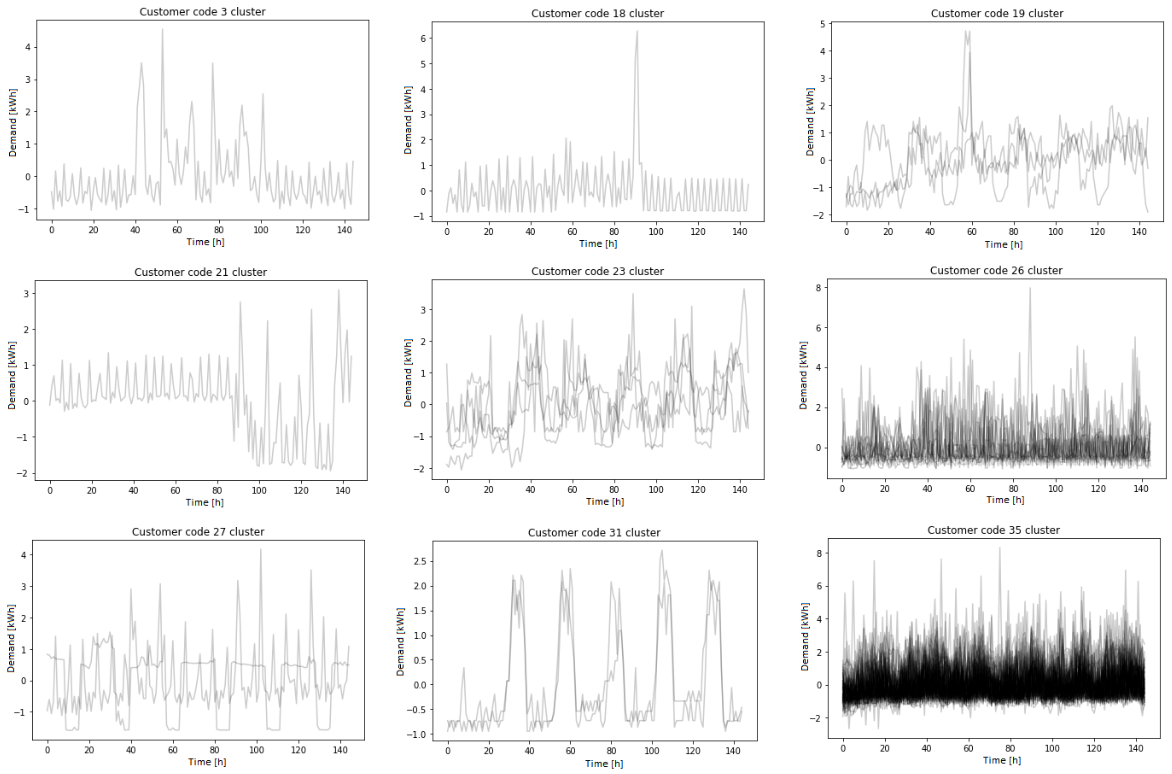


Figure 72: The customer segments according to the standard DSO segmentation method present in the AMS-data set, normalized.

If regarded as clusters, this segmentation of the AMS-data set achieves a Silhouette score of **-0.032**. This Silhouette score is telling of a poor partition. Particularly the partition of customer group 29 and 27 from visual inspection seem to be “ill-formed”.

10.2.2 Developed customer segmentation programme applied to the data set

The developed customer segmentation programme was tested with the whole data set as shown in Figure 72. For the analysis the developed programme as illustrated in Figure 28 was used, though outlier analysis was not employed.

According to the standard DSO customer segmentation method, there exists 9 customer groups in the data set shown in Figure 72. The Silhouette scores for partitions between `K`=2 and `K`=9 were calculated for this data set, within the stability zone of the K-Shape algorithm.

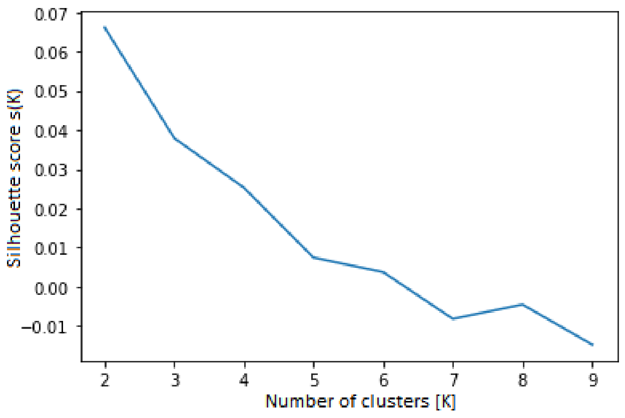


Figure 73: Silhouette scores for all partitions of the data set which included customer segment information, between `K`=2 and `K`=9 clusters.

Table 15: Silhouette scores for all partitions of the data set which included customer segment information, between `K`=2 and `K`=9 clusters.

Number of clusters:	Silhouette Score:
K = 2	0.066
K = 3	0.038
K = 4	0.025
K = 5	0.007
K = 6	0.004
K = 7	-0.008
K = 8	-0.005
K = 9	-0.015

It may be seen from Figure 73 and Table 15 that all partitions between `K`=2 and `K`=9 found by the K-Shape algorithm achieve a better score than the partition proposed by Lyse Elnett in Section 10.2, which had has a Silhouette score of -0.032.

The three partitions which achieved the best Silhouette scores were the partitions for `K`=2, `K`=3, and `K`=4. The K-Shape algorithm was then instructed to produce these three partitions, with `n_init` between 3000 and 4000 (`KShape` stability zone). The results are shown in Figure 74, Figure 75 and Figure 76.

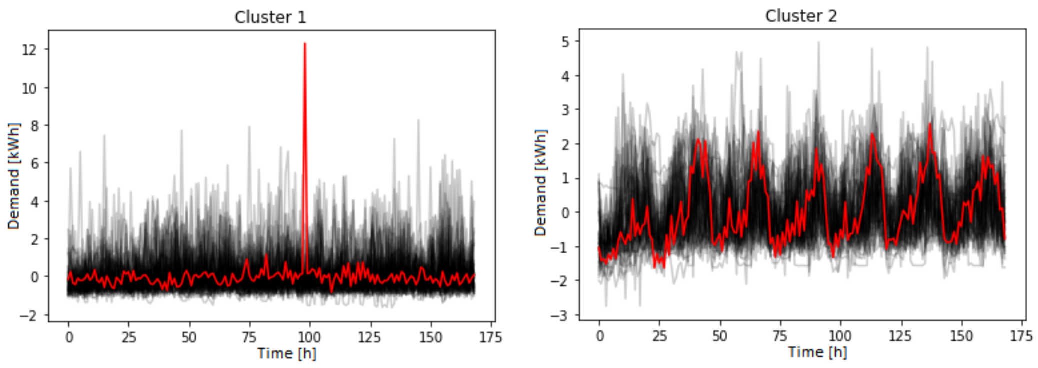


Figure 74: The customer segments created by the K-Shape algorithm from the original data set, with `K`=2.

It may be seen in Figure 74 that the K-Shape algorithm initially segments the data set based on the perceived "periodicity" of the consumption pattern. Cluster 2 in Figure 74 displays a consumption pattern which repeats itself around once every 24 hours (with some variation), while Cluster 1 displays a consumption pattern with a periodicity of less than 24 hours. Additionally, from Cluster 1 in Figure 74 the K-Shape algorithm has found that many customers have a single large consumption spike in the course of a week, and relatively even consumption for the rest of the week (when normalized).

When instructed to make `K`=3 clusters, the K-Shape algorithm in Figure 75, Cluster 3, creates a segment of consumption data with a periodicity which resembles the one in Cluster 1 of Figure 74 but without the one large weekly consumption spike.

When instructed to make `K`=4 Clusters, the same effect in Figure 76 as in Figure 75 may be seen. The K-Shape algorithm creates one more cluster with a periodicity of less than 24 hours, and without the one large weekly consumption spike.

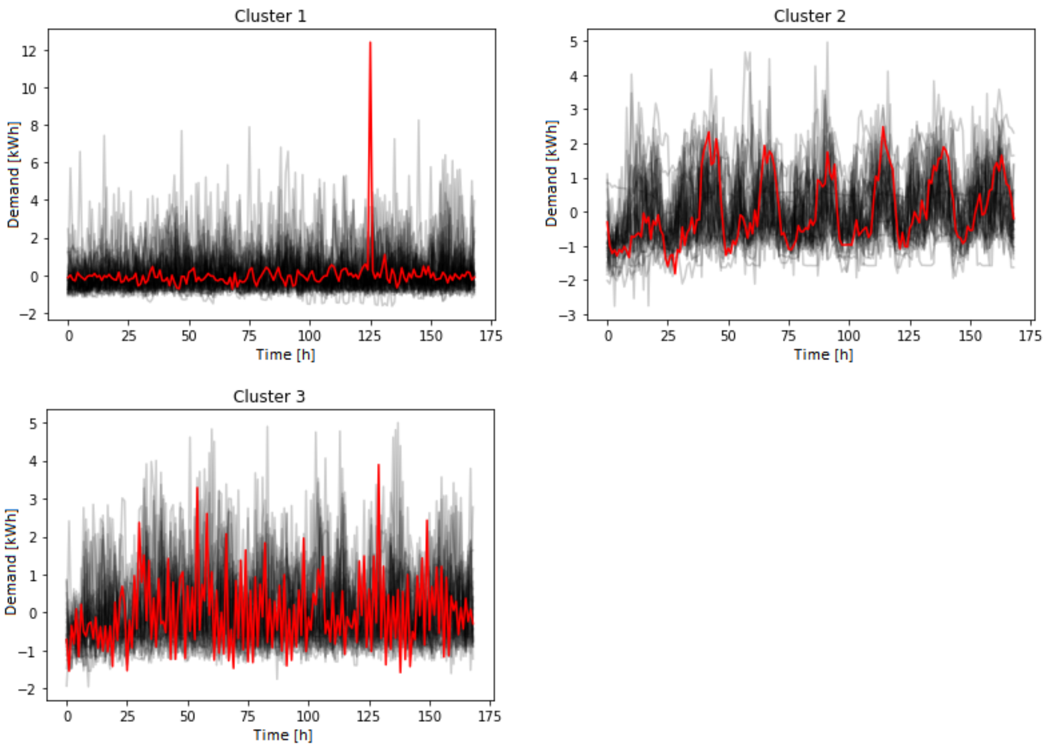


Figure 75: The customer segments created by the K-Shape algorithm from the original data set, with K=3.

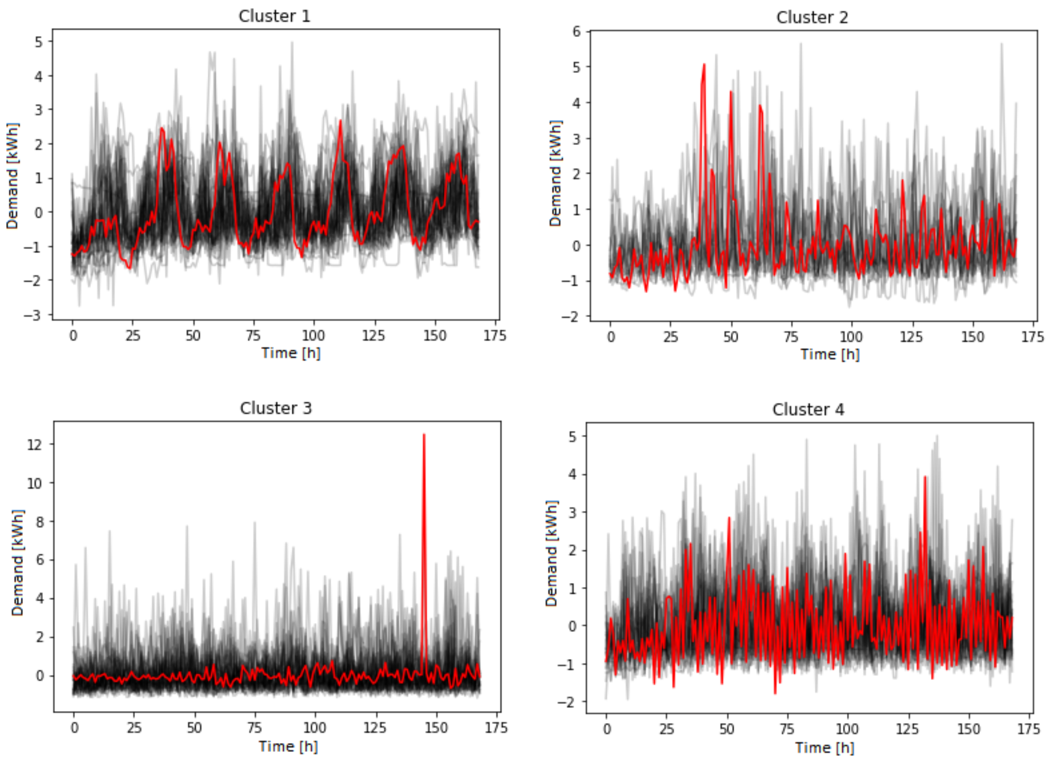


Figure 76: The customer segments created by the K-Shape algorithm from the original data set, with K=4.

If `K`=9, the programme produces the clusters as shown in Figure 77. Cluster compactness is not easily assessed through visual inspection with shape-based clustering, due to the time invariance of the clusters and centroids. However, this partition achieves a Silhouette score of -0.015, which is a higher score than the standard DSO segmentation method achieves (-0.032).

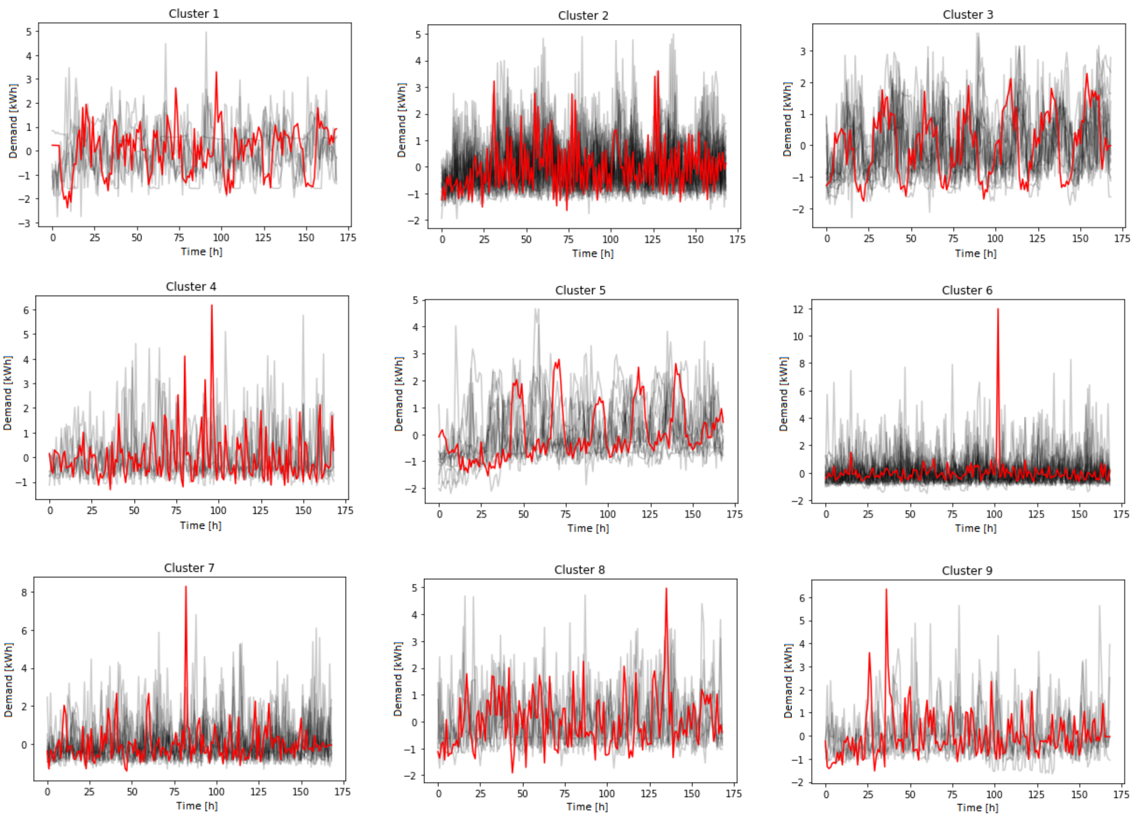


Figure 77: The customer segments created by the K-Shape algorithm from the original data set, with `K`=9.

10.2.3 Removing customer groups with single time series from AMS-data set

As may be seen in Table 13 and Figure 72, three of the "clusters" included only one consumption profile (customers in code 3, 18 and 21). For these customer segments it is not possible to evaluate to what extent the single costumer is representative of other customers in the same segment. Therefore, an additional assessment of the data set with these three customer groups removed is performed in this Section.

The data set with customer group 3, 18 and 21 removed is shown in Figure 78.

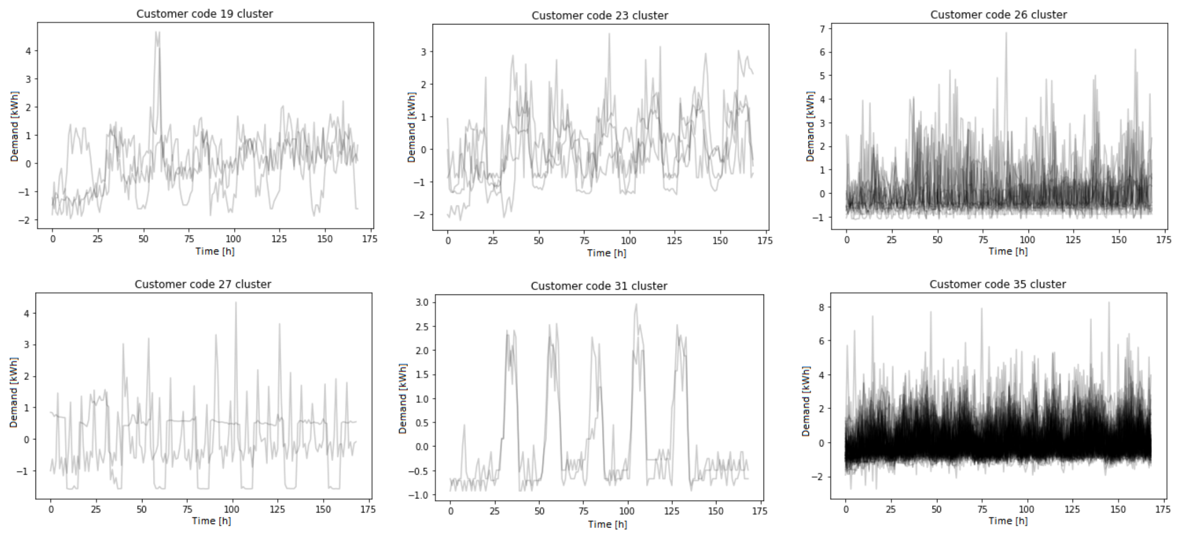


Figure 78: The customer segment groups with more than one customer per group, normalized.

If regarded as clusters, this segmentation of the AMS-data set achieves a Silhouette score of **-0.006**. This is an improvement on the overall partition quality, when compared to the Silhouette score in Table 16.

The Silhouette scores for partitions between K=2 and K=9 were found for the data set, within the stability zone of the K-Shape algorithm.

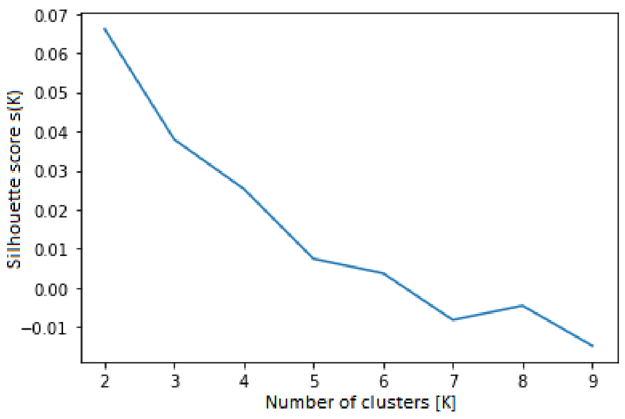


Figure 79: Silhouette scores for all partitions of the data set with single time series groups removed.

Table 16: Silhouette scores for all partitions of the data set with single time series groups removed, between `K`=2 and `K`=9 clusters.

Number of clusters:	Silhouette Score:
K = 2	0.072
K = 3	0.034
K = 4	0.013
K = 5	0.001
K = 6	0.002
K = 7	-0.005
K = 8	-0.007
K = 9	-0.015

When instructed to find `K`=2 clusters, the programme produces the clusters and centroids shown in Figure 80. This result greatly resembles the result found in Section 10.2.1, Figure 74. When instructed to find `K`=3 clusters, the programme produces the clusters and centroids shown in Figure 81. This result also greatly resembles the result found in Section 10.2.1, Figure 75. Finally, when instructed to find `K`=4 clusters, the programme produces the clusters and centroids shown in Figure 81.

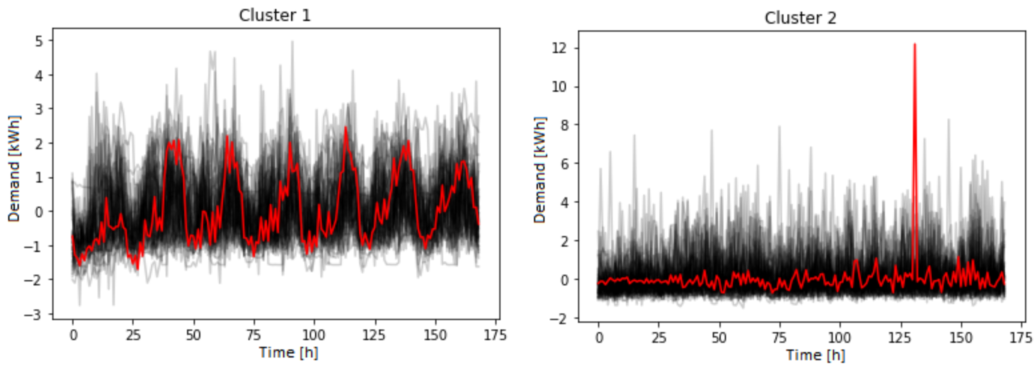


Figure 80: The customer segments created by the K-Shape algorithm from the data set with single time series groups removed, with `K`=2.

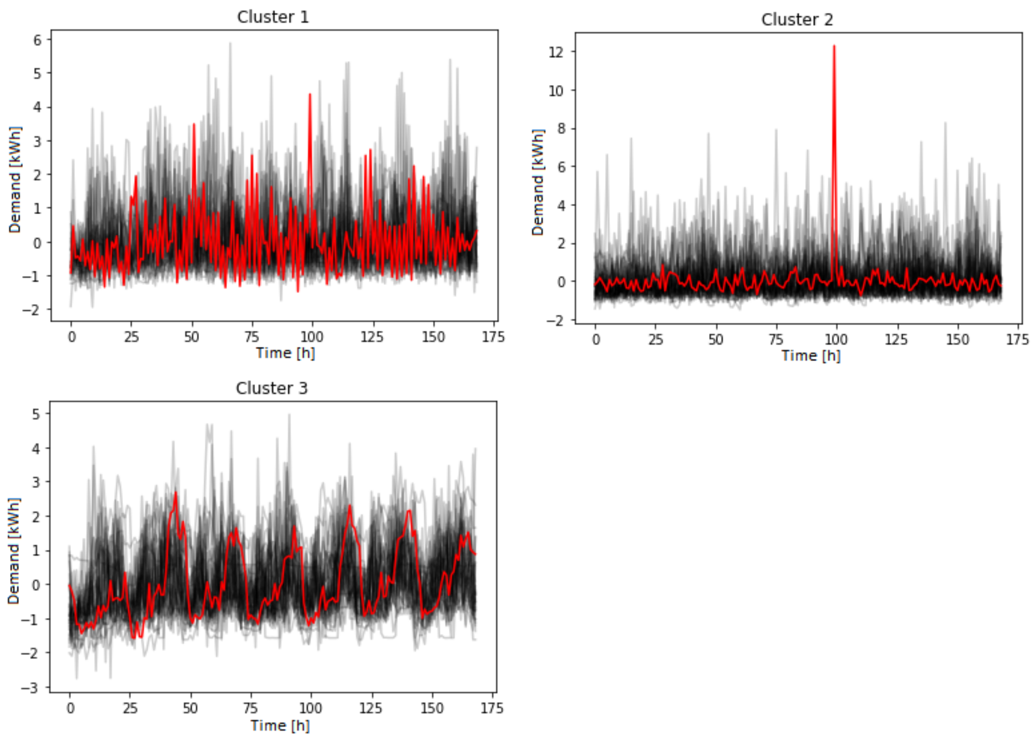


Figure 81: The customer segments created by the K-Shape algorithm from the data set with single time series groups removed, with K=3.

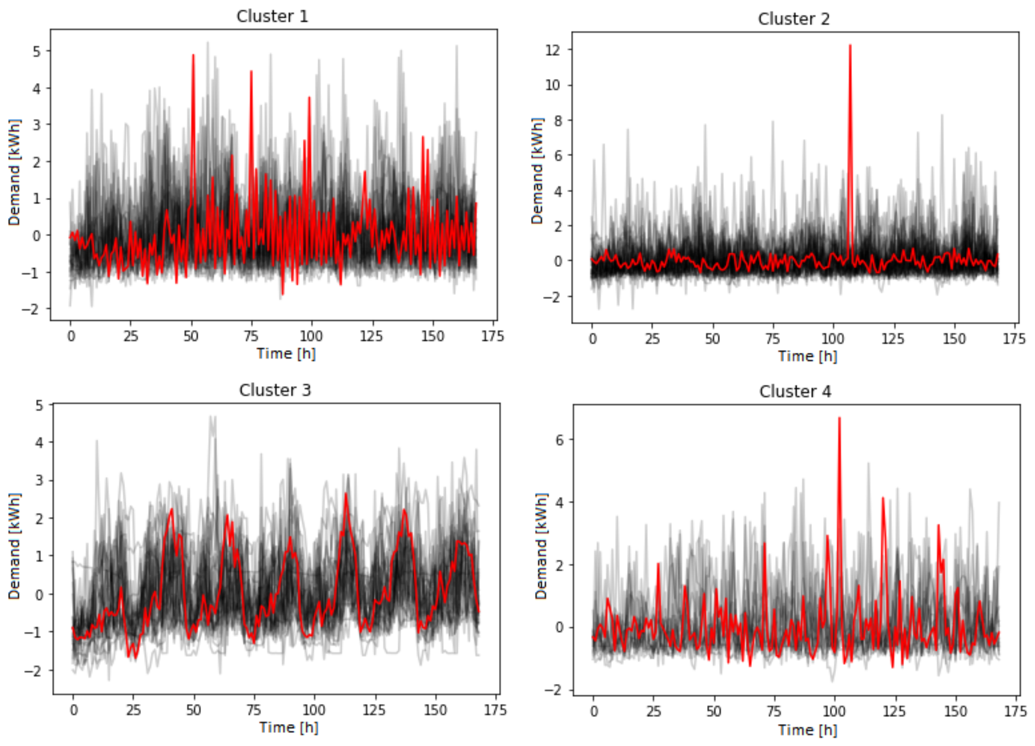


Figure 82: The customer segments created by the K-Shape algorithm from the data set with single time series groups removed, with K=4.

This final result resembles the result found in Section 10.2.1, Figure 76 as well.

As found in this Section, clusters and cluster centroids were not much changed by removing the single time series groups when inspected visually. However, as may be observed when comparing Table 15 and Table 16 the Silhouette scores for the different partitions (values of `K`) were generally slightly improved.

10.2.4 Conclusions from the comparison

As was found in both Section 10.2.1 and Section 10.2.3, the developed customer segmentation programme produced segments with a higher Silhouette score than the standard DSO customer segmentation method does. In other words, the developed customer segmentation programme produces clusters with an improved *compactness* (described in Section 7.3.3).

As stated in Section 10.2, the developed customer segmentation method and the standard DSO customer segmentation method may have different foundations. Although the actual foundation for the standard DSO segmentation method is unknown, the standard DSO method is based on grouping together, i.e. clustering, different customer "types" (such as "Financial services, insurance and pension funds" and "Public administration and defence"). For this reason, the standard DSO method may be more sensitive to for example amplitude and offset variations. On the other hand, the developed programme is based on shape-based clustering, with the aim of grouping together customers with of similarly shaped consumption patterns.

As both segmentation methods appear to be based on different principles, both may provide benefits for various uses. The developed customer segmentation programme can be regarded as a supplementary segmentation method which provides additional information.

11 Conclusions

The Norwegian power system will in the coming years undergo changes due to new European Union and Norwegian legislation as well as increasing electrification of the transportation sector, decentralisation of power production and the general trend of digitalization.

One result of the digitalization trend, which also has been formalised through legislation, is the implementation of AMS-meters at all consumer locations in the Norwegian distribution grid. An AMS-meter is able to capture a wide variety of data types in real time. This has the potential to benefit DSOs in several ways. However, as was confirmed during work with this report, DSOs have up to now only to a limited extend exploited the potential benefits of using AMS-data.

Machine learning has become an important area of research, which has resulted in increased accessibility of machine learning algorithms and methods. Now there exists a variety of computer algorithm libraries for machine learning, in various coding languages. This opens the possibility for utilization of freely accessible computer algorithms for machine learning for efficient analysis of for example AMS-data at insignificant cost. One such library, which can be applied to time series data is `tslearn`, a machine learning library for Python.

In relation to this work the founder and creator of `tslearn`, Romain Tavenard, was contacted. Through a dialogue it was concluded that when using machine learning algorithms such as clustering algorithms it is important to understand what type of distance metrics the algorithm is based on. Only the machine learning library `tslearn` was found to operate with distance metrics suitable for time series.

In the course of this report, a synthesis computer programme was developed for shape-based (i.e. amplitude, offset and time invariant) DSO customer segmentation based on AMS time series data. On a known/defined data set the developed customer segmentation programme was shown to produce the results that were known to be correct beforehand. Outlier analysis methods, which the user can chose to apply, were also added to the programme. These outlier analysis methods were shown to be able to sort out predefined outliers provider user input and guidance. The outlier analysis methods have been included into the programme as an option that the user can select, i.e. not a necessity. The outlier analysis part is only developed to a certain point and can be further improved.

An assessment was made on the basis of actual AMS-data sets provided by Lyse Elnett. When tested on the first AMS-data set, the programme showed a tendency to prefer segmenting based on similar periodicity of the time series. Additionally, AMS-data was more challenging for the algorithm to cluster than the known data set, as the AMS-data set contained less distinct time series groups and to some extent more irregular data. Outlier analysis was shown to improve the programme performance by removing irregular (i.e. flat) time series.

When testing the developed programme for the known data set, the runtime of the clustering algorithm was as expected, i.e. acceptable. However, when analysing the

AMS-data set, the clustering algorithm in the programme became significantly less stable. As a consequence, the runtime of the programme increased dramatically due to the increased number of iterations needed to reach algorithm stability (i.e. $\approx$40 iterations on the known data set, to $\approx$10000 for the AMS-data set). This is possibly because the AMS-data set was more homogeneous with more similarly shaped and less discernible time series groups. The large number of required iterations proved challenging for the available hardware (a standard laptop), with a runtime of $\approx$10 hours for 10000 iteration. Another reason for the extended runtime may be the coding language that the customer segmentation programme is written in, i.e. Python. A lower-level language would likely significantly improve the programme runtime.

The developed customer segmentation programme was shown to produce a better partition compactness of the second AMS-data set than the standard DSO method when measured with a CVI. It should be remembered that the basis for the standard DSO segmentation method is not necessarily founded on segmenting for similar consumption shapes, but for similar customer types (i.e. "Information and communication", "Financial services, insurance and pension funds", and "Public administration and defence"). For this reason, the developed customer segmentation programme may be used as a complementary method, which provides additional insight to the standard customer segmentation approach.

As stated in the report scope, the objective of this study is to illustrate that opportunities and benefits related to the use of AMS-data can be achieved by applying existing, openly available advanced data analysis tools, and be utilized by DSO engineers with a power systems background and an information technology interest.

In this work, the computer-based programme has been developed based on openly available advanced data analysis tools, and is relatively user friendly. The developed programme aims to segment DSO customers with a shape-based clustering approach, with the use of AMS time series consumption data. The programme is considered usable for a DSO engineer with a power systems background and an information technology interest. Experience with the programme indicates that runtime may be an issue when analysing large AMS-data sets.

Some potential improvements to the developed programme were also identified. One of these was the potential for the programme to take into account the standard DSO segmentation, in order to include amplitude and offset information in the analysis. Another area of improvement was the language in which the programme was written. Python is a relatively high-level coding language, and the efficiency of the programme and thereby calculation time may be improved if a lower-level language such as C++ were utilized.

12 Programme improvements, further work and notes on the report

12.1 Potential modifications to improve the programme

12.1.1 Combination of developed and standard segmentation model

As indicated in Section 10.2.4, the developed programme could be modified to include information on customer segments as currently defined by Norwegian DSOs, given in Table 14. This way it may be possible for the programme to be more sensitive to for example offset and amplitude. For example customers in the "Production of primary aluminium" likely have a larger electrical consumption need than for example the "Education" category.

The programme could be improved with the ability to analyse the produced clusters and inspect the distribution of the customers according to the standard DSO segmentation method which exist within each cluster.

12.1.2 Improve runtime by changing code language

The largest issue when performing the analysis in Section 10.1 and Section 10.2 was the speed of the Python code. Whenever `n_init`>1000 the `tslearn KShape` algorithm would take hours to complete the computation for any `K`. Given that the necessary hardware to run such computationally hard code in a shorter time was unavailable, this was an obstacle to the completion of the analysis.

One remedy to this problem may be to write some of the code in C++. The `tslearn` source code for the `KShape` function is available on GitHub at reference [89], so anyone attempting to make the customer segmentation programme faster can look into writing some of the `KShape` algorithm code in another coding language such as C++ instead of Python. C++ code is used as an example as it is widely accepted to be a much faster code to compile than Python code, as the C++ language is more low-level [70].

12.1.3 Improve runtime by manipulating the time series data

To decrease the complexity of the data set, a rolling average as discussed in Section 7.5.4 could be applied to an AMS-data set. This could potentially decrease the runtime of the `KShape` algorithm. A rolling could also potentially decrease "noise" in a data set and so make the "shape" of the time series more clear.

12.1.4 Testing other clustering methods

In this report the `tslearn KShape` algorithm was chosen as the clustering algorithm for demonstrating the customer segmentation method. However, `tslearn` includes several other clustering algorithms for time series. Among others, a K-Means algorithm. Since the programme model is modular, the `tslearn KShape` algorithm may simply be changed for another clustering algorithm. Note however the challenges of selecting algorithms for time series, as mentioned in Section 7, where it is explained that distance metrics in the algorithms should be adapted to time series data.

12.1.5 Improving outlier analysis

The outlier analysis used in this report was found in Section 9.3 to have several weaknesses which could potentially make it an undesirable took to use in some situations. As explained in Section 7.5.3 more advanced forms of outlier analysis for time series exist, and the customer segmentation method may be improved by implementing another form of outlier analysis.

12.2 Further work

- Decrease the runtime of the customer segmentation method code by for example writing it wholly or partially in another coding language, as explained in Section 12.1.2.
- Investigate other forms of outlier analysis, more details of this may be found in Section 7.5.3.
- Comparing the customer segmentation method with a larger data set of AMS-data that includes information on customer segments, as done in Section 10.2.
- Investigate other formats of AMS-data from AMS-meters produced by for example Kamstrup and Kaifa (introduced in Section 4.2), and modify the code in Appendix II to fit any AMS-data from AMS-meters installed in Norway.
- Investigate solutions to P1, P2 or P3 from Section 5.4.
- Particularly for P3, Long Short Term Memory (LSTM) neural networks with Python, Keras and TensorFlow could be researched.
- Test whether other clustering algorithms perform better with AMS-data than K-Shape. Highlighted in section 12.1.4.
- Adapt other CVIs to time series data by changing the distance metric to one which may be used for time series. The challenges of CVIs for time series is explained in Section 7.4
- Examine whether the `KShape` algorithm is stable over time (a condition for cluster usefulness, described in Section 7.3.3) for an AMS-data set.
- Investigate distribution of standard DSO customer segments in the clusters produced by the developed customer segmentation programme.
- Develop a method for visually inspecting shape-based clusters despite time-shifts by optimally aligning the cluster time series and centroids in a figure. This could potentially be done with a distance metric for time series such as Dynamic Time Warping, or shape-based distance (elaborated in Section 7.1).

12.3 Notes on the writing of this report

12.3.1 Reproducibility of results

As stated in section 9.2, the reproducibility of results in this report has been a topic of focus. As a consequence, all data sets (which were not considered power-sensitive information) used in this report may be generated from code in Appendix II. All analysis on these data sets may also be reproduced with code from Appendix II.

Ensuring reproducibility of results with analysis on power-sensitive information was not possible, given that power-sensitive information is classified and could not be included in this report. However, Appendix IV includes all code for the analysis performed in Section 10.1 and Section 10.2, so similar results may perhaps be reproduced on other AMS-data sets of similar sizes and compositions.

12.3.2 Biases in scientific research

Numerous biases related to scientific research have been described in the literature, which may cause concerns related to the reliability and integrity of the scientific enterprise. Of the most common bias patterns found in the scientific literature, this report may be most vulnerable to the “Small-study effect” [27].

The Small-study effect is characterised by [27]:

> Studies that are smaller (of lower precision) might report effect sizes of larger magnitude. This phenomenon could be due to selective reporting of results or to genuine heterogeneity in study design that results in larger effects being detected by smaller studies.

The data set in Section 9 included four known time series groups, designed to demonstrate the amplitude, offset and time invariance of the programme. However, one bias of the report may be that the known data set was particularly “easy” to cluster for the K-Shape algorithm, and that less promising results for the programme could have been obtained given a data set with other types of time series groups. There could be indications of this effect when analyzing the AMS-data in Section 10.

The data set used in Section 10.2 for comparing the developed customer segmentation programme to the standard DSO customer segmentation method was small, with AMS-data from only 103 customers. It is possible that given a larger, or a differently composed AMS-data set, that standards DSO segmentation method could produce partitions with an equivalent or superior Silhouette score compared to the developed programme.

12.3.3 Available computing hardware

Towards the end of the writing of the report, during the analysis of the AMS-data sets in Section 10.1 and Section 10.2 the runtime of the programme became very long, because such large values of `n_init` were needed. To find the stable region of the `KShape` algorithm for the AMS-data sets in Section 10.1 and Section 10.2 before the due date of the report, six laptop computers borrowed from friends and family were used to run the developed programme over two weeks. Two of these computers were situated in Trondheim, while four were controlled remotely from Stavanger with the use of TeamViewer. Therefore, for large AMS-data sets, more advanced computing hardware than common laptops could be used to cut down on programme runtime.

References

[1] A Abur. *Power System State Estimation.* CRC Press, Boca Raton, 2004.

[2] *Advanced Distribution Management Systems (ADMS)*. Gartner Inc. URL: https://www.gartner.com/it-glossary/advanced-distribution-management-systems-adms (visited on 03/17/2020).

[3] Ann Myhrer Østenby et al. *ANALYSE OG FRAMSKRIVNING AV KRAFT-PRODUKSJON I NORDEN TIL 2040*. 2019. URL: http://publikasjoner.nve.no/rapport/2019/rapport2019_43.pdf (visited on 03/04/2020).

[4] Cheryl martin et al. *The Future of Electricity*. World Economic Forum, 2017.

[5] Junjing Yang et al. *K-Shape clustering algorithm for building energy usage patterns analysis and forecasting model accuracy*. National University of Singapore, 2017.

[6] Odd Erik Gundersen et al. *Day-ahead Forecasting of Losses in the Distribution Network*. TrønderEnergi Kraft AS, 2020.

[7] Shoichi Kitamura et al. *Disconnection Detection Method for Power Distribution Lines Using Smart Meters*. IEEE, 2015.

[8] Sigurd Kvistad et al. *Drift og utvikling av kraftnettet – utforming av DSO-rollen*. Energi Norge, 2018. URL: https://www.energinorge.no/contentassets/2858551aafa94bb798d89a8edf15a42b/drift-og-utvilkling-av-kraftnettet---rapport-05-12-2018.pdf (visited on 03/13/2020).

[9] Usue Mori et al. *A review on outlier/anomaly detection in time series data*. Cornell University, 2020.

[10] *Algorithm*. Oxford University. URL: https://www.lexico.com/en/definition/algorithm (visited on 05/21/2020).

[11] F. Martinez Alvarez, A. Troncoso, J. C. Riquelme, and J. S. Aguilar Ruiz. *LBF: A Labeled-Based Forecasting Algorithm and Its Application to Electricity Price Time Series*. IEEE International Conference on Data Mining, 2008.

[12] F. Martinez Alvarez, A. Troncoso, J. C. Riquelme, and J. S. Aguilar Ruiz. *Energy Time Series Forecasting Based on Pattern Sequence Similarity*. IEEE Transactions on Knowledge and Data Engineering, 2011.

[13] *Application Programming Interface (API)*. Gartner Inc. URL: https://www.gartner.com/en/information-technology/glossary/application-programming-interface (visited on 05/21/2020).

[14] *AVTALE OM HANDTERING OG BESKYTTELSE AV KRAFTSENSITIV INFORMASJON*. NVE.

[15] Caroline Banton. *Efficiency Definition*. Investopedia, 2020. URL: https://www.investopedia.com/terms/e/efficiency.asp (visited on 03/12/2020).

[16] Gouri R. Barai, Sridhar Krishnan, and Bala Venkatesh. *Smart Metering and Functionalities of Smart Meters in Smart Grid - A Review*. IEEE, 2015.

[17] Colin Alexander Boyd. *Lecture 6: Pseudorandom Numbers and Stream Ciphers*. NTNU, 2019.

[18] Anja Butter. *Deep-learned Top Tagging using Lorentz Invariance and Nothing Else*. ITP, Universitat Heidelberg, 2017. URL: https://indico.in2p3.fr/event/16525/contributions/58417/attachments/45836/57087/DeepTop.pdf (visited on 03/04/2020).

[19] *Cache*. Gartner Inc. URL: https://www.gartner.com/en/information-technology/glossary/cache (visited on 05/21/2020).

[20] O. M. Collins and N. Vasudev. *The Effect of Redundancy on Measurement*. IEEE Transactions on Information Theory, 2001.

[21] *COMMISSION REGULATION (EU) .../... establishing a guideline on electricity balancing*. European comission, 2017. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32017R2195&from=EN (visited on 03/04/2020).

[22] *De nye strømmålerne er utstyrt med en HAN-port som gir deg klare fordeler!* Ewave. URL: https://www.ewave.no/blog/de-nye-strommalerne-er-utstyrt-med-en-han-port-som-gir-deg-klare-fordeler/ (visited on 03/10/2020).

[23] *Demand Response (dr)*. Gartner Inc. URL: https://www.gartner.com/en/information-technology/glossary/demand-response-dr (visited on 05/21/2020).

[24] *Drift og utvikling av kraftnettet – utforming av DSO-rollen*. Energi Norge, 2018. URL: file:///C:/Users/Kari/Downloads/Elnett_21__Forbrukerfleksibilitet__avansert_ikke%5C%20kraftsensitiv.pdf (visited on 03/04/2020).

[25] *Energy Management Systems (EMSs)*. Gartner Inc. URL: https://www.gartner.com/it-glossary/energy-management-systems-ems/ (visited on 03/17/2020).

[26] *EUCLIDEAN DISTANCE*. National Institute of Standards and Technology, 2017. URL: https://www.itl.nist.gov/div898/software/dataplot/refman2/auxillar/eucldist.htm (visited on 04/13/2020).

[27] Daniele Fanelli, Rodrigo Costas, and John P. A. Loannidis. *Meta-assessment of bias in science*. Proceedings od the Snational Academy of Sciences in the United States of America, 2017.

[28] *Forskrift om systemansvaret i kraftsystemet*. OED, 2002. URL: https://lovdata.no/dokument/SF/forskrift/2002-05-07-448 (visited on 03/04/2020).

[29] Marc Frincu, Charalampos Chelmis, Muhammad Usman Noor, and Viktor Prasanna. *Accurate and Efficient Selection of the Best Consumption Prediction Method in Smart Grids*. IEEE, 2014.

[30] M. Hawkins. *Identification of outlier*. Springer Netherlands, 1980.

[31] Jordan Hochenbaum, Owen S. Vallis, and Arun Kejariwal. *Automatic Anomaly Detection in the Cloud Via Statistical Learning*. Cornell University, 2017.

[32] Vegard Holmefjord and Anders Kringstad et al. *Et elektrisk Norge – fra fossilt til strøm*. Statnett, 2019. URL: https://www.statnett.no/globalassets/for-aktorer-i-kraftsystemet/planer-og-analyser/et-elektrisk-norge--fra-fossilt-til-strom.pdf (visited on 03/04/2020).

[33] *Hvordan få en elektrisk fremtid uten at sikringen går?* Enova, 2018.

[34] *Information and communication technology.* Oxford university. URL: https://www.oxfordreference.com/view/10.1093/oi/authority.20110803100002983 (visited on 05/21/2020).

[35] Karoline Ingebrigtsen, Volker Hoffmann, and Arne J. Berre. *Energy Analytics – Opportunities for Energy Monitoring and Prediction with Smart Meters.* SINTEF, 2017. URL: https://sintef.brage.unit.no/sintef-xmlui/bitstream/handle/11250/2483460/paper1.pdf?sequence=2 (visited on 03/12/2020).

[36] *Innsending av Norges klimamål til FN.* Regjeringen, 2015. URL: https://www.regjeringen.no/no/aktuelt/innsending-av-norges-klimamal-til-fn/id2403782/ (visited on 03/04/2020).

[37] *INSTALLASJONSVEILEDNING FOR ENERGY SERVICE DEVICES.* Aidon, 2015. URL: https://www.skageraknett.no/getfile.php/1324033-1563186993/Nett/Filer/Proff/Prekvalifiserte/Installasjonsveiledning%5C%20AIDON.pdf (visited on 03/09/2020).

[38] *Internet Of Things (iot).* Gartner Inc. URL: https://www.gartner.com/en/information-technology/glossary/internet-of-things (visited on 06/07/2019).

[39] Weiqing Jiang, Vijay Vittal, and Gerald T. Heydt. *A Distributed State Estimator Utilizing Synchronized Phasor Measurements.* IEEE, 2007.

[40] Alboukadel Kassambara. *CLARA in R : Clustering Large Applications.* Data Novia. URL: https://www.datanovia.com/en/lessons/clara-in-r-clustering-large-applications/ (visited on 05/05/2020).

[41] *KRAFTMARKEDET.* OED, 2019. URL: https://energifaktanorge.no/norsk-energiforsyning/kraftmarkedet/ (visited on 05/15/2020).

[42] *Kraftsystemet.* OED. URL: https://www.energinorge.no/fagomrader/stromnett/kraftsystemet/ (visited on 03/17/2019).

[43] *Leveringsplikt.* NVE, 2019. URL: https://www.nve.no/reguleringsmyndigheten/stromkunde/leveringsplikt/ (visited on 03/04/2020).

[44] *Liste over konsesjonærer.* NVE, 2019. URL: https://www.nve.no/reguleringsmyndigheten/omsetningskonsesjon/liste-over-konsesjonaerer/?ref=mainmenu# (visited on 03/04/2020).

[45] *Logistic regression.* Machine Learning Glossary. URL: https://ml-cheatsheet.readthedocs.io/en/latest/logistic_regression.html?highlight=linear%5C%20regression (visited on 05/21/2020).

[46] Lovdata. *Lov om produksjon, omforming, overføring, omsetning, fordeling og bruk av energi m.m. (energiloven).* OED, 2019. URL: https://lovdata.no/dokument/NL/lov/1990-06-29-50?q=energilov (visited on 03/03/2020).

[47] Cheryl Martin, Francesco Starace, and Jean Pascal Tricoire. *The Future of Electricity, New Technologies Transforming the Grid Edge.* World Economic Forum, 2017.

[48] Hugo Mayo, Hashan Punchihewa, Julie Emile, and Jack Morrison. *Early History of Machine Learning.* URL: https://www.doc.ic.ac.uk/~jce317/history-machine-learning.html (visited on 05/05/2020).

[49] A. P. Sakis Meliopoulos. *Power System Modeling, Analysis and Control.* School of Electrical and Computer Engineering Georgia Institute of Technology, 2006.

[50] Ramin Moghaddass and Jianhui Wang. *A Hierarchical Framework for Smart Grid Anomaly Detection Using Large-Scale Smart Meter Data.* IEEE, 2018.

[51] Ulf Møller. *Elektrifisering av transport.* Energi Norge, 2016. URL: https://www.energinorge.no/fagomrader/energibruk-og-klima/elektrifisering/elektrifisering-av-transport/ (visited on 03/04/2020).

[52] Antonio Moreno-Mufnoz, Daniel Oterino, and Angel Carmona Juan J G. de la Rosa. *Automated Meter Reading Systems in Outage Management.* IEEE, 2007.

[53] *Nå må strømleverandører og nettselskap innhente fødselsnummer for alle strømkunder.* NVE, 2015. URL: http://publikasjoner.nve.no/faktaark/2015/faktaark2015_10.pdf (visited on 03/04/2020).

[54] J. Naas-Bibow. *Norske og europeiske rammer for utøvelse DSO-rollen.* Thommessen AS, 2018. URL: https://www.energinorge.no/contentassets/98d80fc7d5644904b6af3a37fb925d53 11878092_2_2018.06.06-norske-og-europeiske-rammer-for-utovelse-dso-rollen.-presentasjon-medlemsmote-6.6.2018.pdf (visited on 03/03/2020).

[55] *Natural Monopoly.* OECD Glossary of statistical terms. URL: https://stats.oecd.org/glossary/detail.asp?ID=3267 (visited on 05/21/2020).

[56] *Netleie.* NVE, 2019. URL: https://www.nve.no/stromkunde/nettleie/ (visited on 03/04/2020).

[57] *Nettselskap Forsyningsområde.* NVE, 2019. URL: https://gis3.nve.no/ferdigkart/omraedekonsesjon_a0.pdf (visited on 03/12/2020).

[58] *Neural network.* Machine Learning Glossary. URL: https://ml-cheatsheet.readthedocs.io/en/latest/nn_concepts.html?highlight=neural%5C%20network (visited on 05/21/2020).

[59] Andrew Yan-Tak Ng. *Week 1 Lecture Notes: What is Machine Learning?* Coursera, Stanford University.

[60] *Nytt fra NVE 2019.* NVE, 2019. URL: https://www.nve.no/om-nve/presserom/taler-og-presentasjoner/nytt-fra-nve-2019/ (visited on 03/04/2020).

[61] OED. *Forskrift om måling, avregning, fakturering av nettjenester og elektrisk energi, nettselskapets nøytralitet mv.* OED, 1999. URL: https://lovdata.no/dokument/SF/forskrift/1999-03-11-301 (visited on 03/03/2020).

[62] *Om NVE.* NVE, 2015. URL: https://www.nve.no/om-nve/?ref=mainmenu (visited on 03/03/2020).

[63] *Om Olje- og energidepartementet.* OED, 2020. URL: https://www.regjeringen.no/no/dep/oed/org/id774/ (visited on 06/07/2019).

[64] *Om reguleringen av strømnettselskapenes inntekter.* NVE. URL: https://www.nve.no/media/8368/om-reguleringen-av-str%5C%C3%5C%B8mnettselskapenes-inntekter.pdf (visited on 03/10/2020).

[65] *Oppstartsmøte for A5 Forretningsmodeller - Elnett21.* Lyse Elnett AS, 2019.

[66] University of Oslo. *Acts relating to the energy and water resources sector in Norway (unofficial translation)*. OED, 2004. URL: https://app.uio.no/ub/ujur/oversatte-lover/data/lov-19900629-050-eng.pdf (visited on 03/03/2020).

[67] *Outage Management System (OMS)*. Gartner Inc. URL: https://www.gartner.com/it-glossary/outage-management-system-oms/ (visited on 03/17/2020).

[68] John Paparrizos and Luis Gravano. *k-Shape: Efficient and Accurate Clustering of Time Series*. Columbia University, 2015.

[69] *Produktark ESD måler*. Aidon, 2012. URL: https://static1.squarespace.com/static/54ec7207e4b0b896afa8664e/t/5a82ae3fc830259134d7c1e9/1518513730027/Produktark+ESD+%5C%C3%5C%A5ler.pdf (visited on 03/09/2020).

[70] *Python vs. C++: Let's Compare*. BitDegree, 2020. URL: https://www.bitdegree.org/tutorials/python-vs-c-plus-plus/ (visited on 06/01/2019).

[71] *Reguleringsmyndigheten for energi (RME)*. NVE. URL: https://www.nve.no/reguleringsmyndigheten/ (visited on 03/03/2020).

[72] *Reproducibility and Replicability in Science*. National Academies of Sciences, Engineering, Medicine, Policy, and Global Affairs, 2019.

[73] *Reservemarkeder*. Statnett. URL: https://www.statnett.no/for-aktorer-i-kraftbransjen/systemansvaret/kraftmarkedet/reservemarkeder/ (visited on 05/15/2020).

[74] Margaret Rouse. *script*. WhatIs.com, 2005. URL: https://whatis.techtarget.com/definition/script (visited on 05/25/2019).

[75] Peter J. Rousseeuw. *Silhouettes: a graphical aid to the interpretation and validation of cluster analysis*. Journal of Computational and Applied Mathematics, 1987.

[76] T.L. Saaty. *The Analytic Hierarchy Process*. McGraw-Hill, 1980.

[77] Hiroaki Sakoe and Seibi Chiba. *Dynamic Programming Algorithm Optimization for Spoken Word Recognition*. IEEE, 1978.

[78] Kamalanath Samarakoon, Jianzhong Wu, Janaka Ekanayake, and Nick Jenkins. *Use of Delayed Smart Meter Measurements for Distribution State Estimation*. IEEE, 2011.

[79] *Scikit-learn*. NVE, 2020. URL: https://scikit-learn.org/stable/ (visited on 06/04/2019).

[80] *Seasonal-Trend decomposition using LOESS (STL)*. statsmodels v0.12.0.dev0. URL: https://www.statsmodels.org/dev/examples/notebooks/generated/stl_decomposition.html (visited on 05/07/2020).

[81] Chanakya Shah. *Linear Regression*. 2018. URL: http://chanakya.ca/2018/05/13/linear-regression/ (visited on 03/09/2020).

[82] Harry Singh, Shangyou Hao, and Alex Papalexopoulos. *Transmission congestion management in competitive electricity markets*. IEEE, 1998.

[83] Frank Skapalen, Helge Ulsberg, Roger Steen, Rikke C. Arnulf, and Truls Sønsteby. *Veiledning til forskrift om beredskap kraftforsyningen.* NVE, 2011. URL: http://publikasjoner.nve.no/veileder/2011/veileder2011_01.pdf (visited on 03/17/2020).

[84] *Slik fungerer kraftsystemet.* Statnett, 2018. URL: https://www.statnett.no/om-statnett/bli-bedre-kjent-med-statnett/slik-fungerer-kraftsystemet/ (visited on 03/17/2020).

[85] Ida Mattsson Sperre, Henrik Johan Myhrer Janne Hagen, Jon-Martin P. Storm, and Helge Ulsberg. *Oppsummeringsdokument: Endringer i beredskapsforskriften - Krav til IKT-sikkerhet m.m.* NVE, 2018. URL: http://publikasjoner.nve.no/rapport/2018/rapport2018_92.pdf (visited on 03/04/2020).

[86] *Strømmåler og HAN-port.* Lyse Elnett, 2019. URL: https://www.lysenett.no/kunde/faktura-og-forbruk/strommaler-og-han-port/ (visited on 03/12/2020).

[87] Romain Tavenard. *tslearn.generators.random_walks.* 2017. URL: https://tslearn.readthedocs.io/en/stable/gen_modules/generators/tslearn.generators.random_walks.html (visited on 05/08/2020).

[88] Romain Tavenard. *tslearn Documentation, Release 0.3.0.* 2020. URL: https://readthedocs.org/projects/tslearn/downloads/pdf/latest/ (visited on 03/03/2020).

[89] Romain Tavenard. *KShape source code.* tslearn. URL: https://github.com/tslearn-team/tslearn/blob/9d20ed0/tslearn/clustering.py#L877-L1166 (visited on 06/01/2019).

[90] *The Smart Grid.* U.S. Department of Energy. URL: https://www.smartgrid.gov/the_smart_grid/smart_grid.html (visited on 03/17/2020).

[91] Roland Olsson The-Hien Dang-Ha and Hao Wang. *Clustering Methods for Electricity Consumers: An Empirical Study in Hvaler - Norway.* University of Oslo, NTNU and Ostfold University College, 2016.

[92] *Time series.* Oxford University. URL: https://www.lexico.com/definition/time_series (visited on 05/21/2020).

[93] Arne Venjum. *Smarte målere (AMS), Status og planer for installasjon per 1. halvår 2016.* RME, 2016.

[94] Dominik Waeresch, Robert Brandalik, Wolfram H. Wellssow, Joern Jordan, Rolf Bischler, and Nelia Schneider. *Linear State Estimation in Low Voltage Grids based on Smart Meter Data.* IEEE, 2015.

[95] Ali Al-Wakeel, Jianzhong Wu, and Nick Jenkins. *Applied Energy - State estimation of medium voltage distribution networks using smart meter measurements.* Elsevier, 2016.

[96] Kari Walstad. *Secure internet-based communication between SCADA and demand response aggregator in the context of the Elnett21-project.* NTNU, 2019.

[97] *What is metrology?* International Bureau of Weights and Measures (BIPM). URL: https://www.bipm.org/en/worldwide-metrology/ (visited on 05/24/2020).

[98] Zheng Zhang, Romain Tavenard, Adeline Bailly, Xiaotong Tang, and Ping Tang et al. *Dynamic Time Warping Under Limited Warping Path Length.* Information Sciences Elsevier, 2018.

A Appendix I: Key IT systems for DSO operation

This is a verbatim reproduction from the specialization project report titled "Secure internet-based communication between SCADA and demand response aggregator in context of the Elnett21-project" submitted to the Department of Electric Power Engineering in December 2019, found in reference [96].

The DSOs Lyse Elnett and Hafslund, and the technologies firm ABB were interviewed to provide much of the information content of this chapter, as it was not readily available in the literature. Therefore, all statements made in this chapter which are not referenced to a literature source, were provided by either Lyse Elnett, Hafslund or ABB.

Traditionally, the most important IT systems used by the Norwegian DSOs have been Energy Management System (EMS), Distribution Management System (DMS), Supervisory Control And Data Acquisition (SCADA), Outage Management System (OMS), and Network Information System (NIS). In Norway, the systems mentioned above largely operate in a flat hierarchy. This means that no one system is considered a subsystem of another, except for OMS which is considered a subsystem of EMS and DMS. This way of categorizing the different systems differs from convention abroad, where for example DMS, OMS and SCADA may be considered subsystems of EMS [49].

EMS, DMS, SCADA and OMS may be supplied by several different vendors from which a DSO may choose between. Due to this, a DSO could for example have an DMS supplied by ABB and a SCADA system supplied by Siemens. When two software systems that are designed by different vendors interact interoperability problems may arise, and data quality lost. Ensuring proper information security at the interface between two software systems may also prove challenging. Consequently, Advanced Distribution Management System (ADMS) may become a preferred solution for the future. In an ADMS all control and information systems are integrated together into one system (the ADMS), and supplied by one vendor. NIS will however still be supplied by a second vendor.

A.1 Supervisory Control And Data Acquisition (SCADA)

Supervisory Control And Data Acquisition systems are responsible for acquiring data which describes the state of the utility grid for the grid operators. SCADA systems are also responsible for the dispatch of control commands issued from the grid operators to the physical components of the utility grid.

A.2 Energy Management System (EMS)

The global research and advisory firm Gartner Inc. [25] proposes the following definition for Energy Management Systems:

> Energy management systems (EMSs) make up the "central nervous system" of the power transmission network and are the critical governing component of the power grid's operational reliability. EMSs perform

state estimation, contingency analysis and other advanced applications. Operator training applications offer offline study modes and real-time simulations.

EMS is used by DSOs in the parts of the Norwegian utility grid which have voltage levels of between 50 and 132 kV, termed the regional grid (regionalnett). Components of the EMS include State Estimator (SE), Short Circuit Analysis (SCA) and Dispatch Load Flow (DLF). EMS uses data from SCADA in advanced applications to perform real-time analysis on the grid, and for assistance in real-time decision making of the operators. In Norway EMS is not considered a subsystem of SCADA, or vice versa. However, it is common in other countries to define SCADA as a subsystem of EMS [49].

Due to historical reasons, EMS and SCADA are oftentimes fully integrated. This means that the two systems share the same server and user interface, and will from an operator's point of view seem like the same system. In the 1960ies when SCADA systems were first applied to the Norwegian utility grids, the DSOs only had need of computing functionality for the regional grid with higher voltage levels (50 to 132 kV). This was because an outage or fault in the grid with voltage levels between 50 and 132 kV had large social consequences, compared to lower voltage levels which had mostly economical consequences. As a result EMS has a higher emergency regulation classification than DMS. Hence, EMS was installed alongside the SCADA system, and from the same vendor. DMS and OMS would be installed at a later time when the need for such systems arose.

In Norway neither EMS, DMS or OMS has the ability to automatically change the state of breakers in the grid. Only system operators may change breaker connections, which happens through SCADA. EMS may change voltage phasor values in SCADA if it thinks the SCADA values may be incorrect/outdated, after it has computed the State Estimator.

A.3 Distribution Management System (DMS)

Distribution Management System, as utilized by Norwegian DSOs, provides the same functionality as EMS but operates at different voltage levels. While EMS is used for the regional grid, DMS is used for the grid with voltage levels between 400/230 V and 22 kV, termed the distribution grid. Since DMS was often installed prior to EMS and SCADA it is not as closely integrated to SCADA as EMS is, meaning that it will commonly not share its server and user interface with SCADA and EMS.

DMS may not send commands to, or change values in SCADA the way that EMS can (with its computed State Estimator values). Additionally, it is not possible for DMS to manipulate breakers in the grid automatically. DMS acquires its distribution grid information from NIS, in addition to SCADA.

A.4 Outage Management System (OMS)

Gartner Inc. [67] proposes the following definition for an Outabe Management System:

> An outage management system (OMS) is a utility network management software application that models network topology for safe, efficient field operations related to outage restoration. OMSs tightly integrate with call centers to provide timely, accurate, customer-specific outage information, as well as supervisory control and data acquisition (SCADA) systems for real-time-confirmed switching and breaker operations. These systems track, group and display outages to safely and efficiently manage service restoration activities.

Outage Management System exists as a function in EMS and DMS and is used for planning disconnections in the utility grid, when the grid where the disconnections are to take place is so complex that the consequences of a disconnection are not easily seen. For this reason, OMS in medium to small DSOs may only be activated in DMS, because the grid with lower voltage levels has a higher complexity than the grid with higher voltage levels. Large DSOs like Hafslund have OMS activated in their EMS as well.

A.5 Network Information System (NIS)

A Network Information system (NIS) is an IT system which DSOs use to geographically map all utility grid information. NIS provides information on substations, transformer stations and other installations (when they were built, changes that have been made since then, specific data on installation components), zoning plans, customer locations, landowners, specific data on cables, etc. DMS collects data for the distribution grid directly from NIS.

A.6 Advanced Distribution Management System (ADMS)

Gartner Inc. [2] proposes the following definition of an Advanced Distribution Management System:

> An advanced distribution management system (ADMS) is the software platform that supports the full suite of distribution management and optimization. An ADMS includes functions that automate outage restoration and optimize the performance of the distribution grid. ADMS functions being developed for electric utilities include fault location, isolation and restoration; volt/volt-ampere reactive optimization; conservation through voltage reduction; peak demand management; and support for microgrids and electric vehicles.

Advanced Distribution Management System is a highly advanced system that provides the combined functionality of EMS, DMS, OMS, and SCADA. Because it is provided by a single vendor, the interoperability issues that arise when using IT systems from different vendors should not be a problem. There are additionally security advantages to using an ADMS, because of the seamless data interaction between the different components of the ADMS. The disadvantage of changing from the "patchwork" collection of control and information systems that Norwegian DSOs typically have utilized, to employing an ADMS, is that all existing systems will have to be replaced. In other words, it is not possible to keep existing systems and integrate them into an ADMS.

Because of their network size and complexity, Hafslund will in the future change from a "patchwork" solution to using an ADMS. This may be a trend for the future as more DSOs in Norway merge and become larger.

A.7 Emergency regulation related to DSO IT systems

The Norwegian Water Resources and Energy Directorate, NVE, adopts the following classification of plants/electrical installations as mentioned in Section §5-3 of the guidance report on emergency regulations for power supply [83]. Control systems are divided into three classes according to their significance for the country's power supply.

Class 1: Installatons of minor importance.

Class 2: Installations that are important for maintaining the power supply at county level or for operation of regional networks.

Class 3: Installations that have an impact on the power supply in an area, region or for the operation of the central transmission network or for large populations, important infrastructure or other special considerations.

Although the above class definitions are given for electrical installations, they are used in NVE's Guide to regulations on emergency preparedness in the power supply [83] to classify control system criticality. This is because a control system is directly connected to, and can affect the operation of the installations. The SCADA system, EMS system and ADMS is classified as a Class 3 control system. EMS is classified as a Class 3 control system because it shares the same server as SCADA. DMS, as explained in Section A.3 is not as closely integrated with SCADA as EMS is. DMS, OMS and NIS are classified as Class 2 systems. A control and information system's classification governs the strictness of security measures used to protect it.

In NVE's Guide to regulations on emergency preparedness in the power supply [83] control systems are given a separate section, §6-4 Special requirements for control systems.

Special requirement §6-4 b) states the requirement for both physical and electrical access control to a control system. All control systems must have control regimes that effectively protect against internal and remote unauthorized physical and electronic access, distribution of malicious software etc. [83].

Special requirement §6-4 c) states the general requirement for control system security. Class 2 control systems shall be implemented with redundancy up to the individual power supply installations in Class 2 and 3, such that no important functions are lost due to an error or a single undesirable event. Class 3 control systems shall be implemented with full redundancy throughout the system up to the individual power supply installations in Class 2 and 3, and to others relevant Class 2 and 3 operating control systems so that a fault or single event cannot put important functions out of operation. The redundancy should be performed with physical and electronic separation. The operating control system must be designed so robustly that function is maintained under great and long-lasting stress. Class 3 operational control systems shall be capable of operating independently of public networks and telecommunications services [83].

B Appendix II: Data preparation code

This appendix contains the code for the data preparation step wherein the raw AMS-data is extracted from a .txt file and output to another .txt in a format which the machine learning code from `tslearn` can understand. This process is explained in Section 9.1. This Appendix will explain how to adjust the code to other formats of AMS-data which are mentioned in Section 4.4.

B.1 Uploading the AMS-data .txt file to a `pandas DataFrame`

```
#---------------------------PRELIMINARIES----------------------------
import numpy as np
from datetime import datetime
import pandas as pd

# Import data set in .txt file to a pandas data frame
data_frame = pd.read_csv('EksportDataset1.txt', delimiter=';',
   names=('ID','ReadingTime','ReadingValue','Substation'),
   usecols=(0,2,3,5),skiprows=1)
#--------------------------------------------------------------------
```

Any parameter in the `pd.read_csv` may be adjusted to the raw AMS-data .txt file. First, start by writing in the correct name of the .txt file (in this code it is `EksportDataset1.txt`). Then, check whether the delimiter is ";" or something else, in which case the delimiter parameter should be changed. Go on to decide which columns from the raw AMS-data set that should be extracted, and remember that Python starts indexing at 0 not 1. These columns are specified as `usecols` indexes. Each of the chosen columns may be given a "label" in the `pandas DataFrame`. In this case the labels `names=('ID','ReadingTime','ReadingValue','Substation')` are chosen. The `skiprows` parameter is chosen to be 1 because the first row in the data set does not include any AMS-data. This may be seen in Figure 30 in Section 4.4.

Before proceeding it is advisable to run the above code and see if the correct data is uploaded to the `pandas DataFrame` variable, `data_frame`.

B.2 Time management of the `DataFrame`

```
#--------------------------------TIME--------------------------------
# Turn the ReadingTime column into a datetime oject
data_frame['ReadingTime']=data_frame['ReadingTime'].apply(lambda x :
   datetime.strptime(x,"%d.%m.%Y %H:%M:%S"))

# min and max time
min_time = data_frame['ReadingTime'].min()
max_time = data_frame['ReadingTime'].max()
```

```
# Create single column data frame of datetime object from min_time to
    max_time
# Please change 'freq' if the sampling frequency is not hourly.
datetime_list = pd.date_range(min_time,max_time,freq='H')
#-------------------------------------------------------------------------
```

The point of this code is to find the earliest and latest AMS consumption data in time, `min_time` and `max_time`, and make a list of discrete time intervals in between these measurement times. In the first line of code it should be checked whether the order of `%d.%m.%Y %H:%M:%S` corresponds to the order of the date and time measurement in the raw AMS-data which has been uploaded to `data_frame`. View Figure 30 in Section 9.1 to see how date and time was represented in the AMS-data used for this report. Also, check whether the measurement frequency is hourly. If it is not, change the `freq='H'` parameter to a correct measurement interval parameter.

Before proceeding it is advisable to run the above code and check whether the variable `datetime_list` contains a discrete list of `datetime` objects between `min_time` and `max_time`.

B.3 Miscellaneous operations

```
#-----------------------------SORTING & MISC------------------------------
# Changing the "," to "." in column thee and making it a float.
data_frame['ReadingValue']=data_frame['ReadingValue'].apply(lambda y :
    y.replace(",","."))
data_frame['ReadingValue']=data_frame['ReadingValue'].apply(lambda y :
    float(y))

# What datatype are the different fields?
# You should check that data_frame contains the correct data types.
# Customer ID should be an integer (int)
# Reading time should be a datetime object (datetime)
# Reading value should be a floating point integer (float)
# Substation should be an string (object)
what_dtype = data_frame.dtypes

### here we can delete substations ###

# Sort the 'ID' column, ascending. ID values were an int by default.
# Then, sort the 'ReadingTime' values according to the 'ID' column
data_frame.sort_values(by=['ID','ReadingTime'])

# How many IDs do we have?
IDs = data_frame['ID'].max()
#-------------------------------------------------------------------------
```

The first part of this code deals with replacing all commas “,” with a period “.”, as Python sees decimals by periods, not commas. The second part of the code, `what_dtype`, is written as a check for the user to see if the `data_frame` variables now contains data of the right type (it should). The data in `data_frame` is then sorted by ID, and each ID is then sorted in time.

There has also been added a space where specific substations may be removed from the `data_frame`. The code for this operation was not written because it was not needed in the report, but it should not be particularly challenging to implement after a few google-searches on slicing and removing `pandas DataFrame` rows.

Before proceeding it is advisable to run the above code and check whether all data in the `data_frame` variable is of the correct type, by looking at `what_dtype`.

B.4 Making a `DataFrame` with the correct format for `tslearn`

```
#-----------------------------MAKE NEW DATA FRAME------------------------
# Create new data frame which will contain all the ReadingTimes for the
# different IDs along the columns, for each time along the rows
array = np.empty((datetime_list.size,IDs))
array[:] = 0
TimeSeriesDF = pd.DataFrame(array, index = datetime_list)

for i in range(1,IDs):
    # Create a data frame which only contains the ReadingTime and
    # ReadingValue for each ID.
    Value_and_time =
        data_frame.loc[data_frame.loc[:,'ID']==i,['ReadingTime','ReadingValue']]

    # Re-index Value_and_time.
    # All missing ReadingTimes are replaced with NaN.
    Value_and_time = Value_and_time.set_index('ReadingTime')
    Value_and_time = Value_and_time.reindex(datetime_list)

    # Allocate the ReadingValues in Value_and_time to TimeSeriesDF
    TimeSeriesDF.iloc[:,(i-1)] = Value_and_time.iloc[:,0]
#------------------------------------------------------------------------
```

This part of the code does not need a lot of tampering. However, if `names = ('ID', 'ReadingTime', 'ReadingValue', 'Substation')` has been changed in the `pd.read_csv` function then these label names will have to be changed in the code below `read_csv`. This applies of course to the entire code in this appendix, not just the code in this section.

B.5 Specifying the time interval for further analysis

As may be read in the comments on the code in this section, the code user should now specify which time interval he/she wishes to select from the AMS-data set.

```
#---------------- CHOOSE THE TIME SERIES TO WORK WITH ------------------
# Write in when you want the first and last measurement in the time
# series to be.
# Remember to check that it lies in between max_time and min_time.
# The format is: datetime(year,month,day,hour,minute,second)
# Since we have hour-wise readings, the format is:
# datetime(year,month,day,hour,0,0). Do not write 0 in front of any
# nonzero input.
Start_date_time = datetime(2017,10,15,0,0,0)
End_date_time = datetime(2017,10,22,0,0,0)


final_TS = TimeSeriesDF.loc[Start_date_time:End_date_time,:]
final_TS = final_TS.T

# Remove time series with NaN
final_TS = final_TS.dropna(axis=0)
#-----------------------------------------------------------------------
```

B.6 Writing the final data set to a new .txt file

At the end, the data set is saved to a .txt file. The name of the text file may be changes from `final_time_series.txt` to any other name.

```
#-------------------WRITE FINAL TIME SERIES TO .txt FILE ----------------
open('final_time_series.txt', 'w').close()
final_TS.to_csv('final_time_series.txt', sep=',' , header=None ,
    index=None )
#-----------------------------------------------------------------------
```

C Appendix III: Code for assessing the developed customer segmentation programme

C.1 Base case

This is the code used for executing the experiment described in Section C.1.

Generating sine and square waves

The code given in this section generates sine and square wave time series, and stores them in a .txt file called Mixed_TS.txt.

```
import numpy as np
import matplotlib.pyplot as plt
from random import randint
import math
#------------------------------- GENERAL -----------------------------
time_end = 275
time_vector = np.linspace(0,time_end-1,time_end)

SQUARE_size = 50
SINUS_size = 50
#.....................................................................

#--------------------------- SQUARE WAVE -----------------------------
SQUARE_array = np.zeros(shape=(SQUARE_size,time_end))
for i in range(SQUARE_size):
    width = 16#randint(16,17)
    height = randint(1,10)
    square = np.zeros(shape=(1,width*2))

    for j in range(width):
        square[0,j] = height

    residual = (time_end)%(width*2)
    repeat = math.floor(time_end/(width*2))

    SQUARE_fin = square
    for j in range(repeat-1):
        SQUARE_fin = np.append(SQUARE_fin,square)

    if residual > 0:
        if residual < width:
            res = np.zeros(shape=(residual))
            res.fill(height)
            SQUARE_fin = np.append(SQUARE_fin,res)
        else:
            res = np.zeros(shape=(1,width))
            res.fill(height)
            SQUARE_fin = np.append(SQUARE_fin,res)
            SQUARE_fin =
               np.append(SQUARE_fin,np.zeros(shape=(1,residual-width)))

    SQUARE_array[i,:] = np.transpose(SQUARE_fin)

plt.figure()
for xx in range(SQUARE_size):
    plt.plot(SQUARE_array[xx,:], "k-", alpha=.2)
plt.title("Square")
plt.tight_layout()
plt.show()
#.......................................................................

#---------------------------- SINE WAVE ------------------------------
SINUS_array = np.zeros(shape=(SINUS_size,time_end))
for i in range(SINUS_size):
    phase = randint(2,6)
    SINUS_array[i,:] = np.sin(time_vector + np.pi/phase)

plt.figure()
for xx in range(SINUS_size):
    plt.plot(SINUS_array[xx,:], "k-", alpha=.2)
plt.title("Sine curve")
plt.tight_layout()
plt.show()
#......................................................................

#------------------------ FINAL SET ----------------------------------
final_set = np.concatenate((SINUS_array,SQUARE_array))

row,col = final_set.shape

plt.figure()
for xx in range(row):
    plt.plot(final_set[xx,:], "k-", alpha=.2)
plt.title("final")
plt.tight_layout()
plt.show()

open('Mixed_TS.txt', 'w').close()
np.savetxt("Mixed_TS.txt", final_set, delimiter = ",")
#.......................................................................
```

Performing Silhouette analysis and K-Shape clustering

The code in this section concatenates the Mixed_TS.txt data set of square and sine waves with the `tslearn` cached time series. A Silhouette score is then assigned to partitions between variables `start` and `stop`. The data set is then clustered in to `K_temp` clusters. In this appendix all values of `n_init` are set to stable values for the respective data sets. However, `n_init` may be varied to test the K-Shape algorithm stability for each different data set. `max_iter` is left at default value, which is 100 in `tslearn` [88]. This is because `max_iter` was found to not effect the K-Shape algorithm stability.

```
#=====================================================================
import numpy as np
from tslearn.datasets import CachedDatasets
from tslearn.preprocessing import TimeSeriesScalerMeanVariance
from tslearn.clustering import silhouette_score
from tslearn.clustering import KShape
import matplotlib.pyplot as plt
#=====================================================================

#=====================================================================
#---------------------------- DATA SET -------------------------------
ts_2 = np.loadtxt('Mixed_TS.txt',delimiter=',')
np.random.shuffle(ts_2)

X_train, y_train, X_test, y_test = CachedDatasets().load_dataset("Trace")
np.random.shuffle(X_train)
ts_1 = X_train[:,:,0]

ts = np.concatenate((ts_1,ts_2), axis = 0)
ts_norm = TimeSeriesScalerMeanVariance().fit_transform(ts)
#....................................................................

#------------------------------ PLOT ----------------------------------
row,col = ts.shape

plt.figure()
for xx in range(row):
    plt.plot(ts[xx,:], "k-", alpha=.2)
plt.title("Manufactured time series")
plt.tight_layout()
plt.show()

ts_norm_plot = ts_norm[:,:,0]
plt.figure()
for xx in range(row):
    plt.plot(ts_norm_plot[xx,:], "k-", alpha=.2)
plt.title("Time series after normalization")
plt.tight_layout()
plt.show()
#....................................................................
#=====================================================================
#=======================================================================
Test_silhouette = int(input("Do you want to test silhouette score for
   multiple K? Type 1 for yes, and 0 for no: "))

if Test_silhouette:

    start = int(input("Start at which K: "))
    stop = int(input("Stop at which K: "))

    silhouette_vector = [0]*(stop-start)
    list_vector = list(range(start,stop))

    for n_clusters in range(start,stop):
    #------------------------ K-SHAPE -------------------------------
        # Generate random integer (starting point) for K-Shape algorithm
        seed = 0
        np.random.seed(seed)

        # Cluster data into n_clusters number of clusters
        ks = KShape(n_clusters,random_state=seed,n_init=40)
        ks.fit(ts_norm)

        # Extract centroids
        ks_centroids = ks.cluster_centers_

        # Assign each time series to each centroid
        ks_preds = ks.fit_predict(ts_norm)
    #----------------------------------------------------------------

    #---------------------------- SILHOUETTE -------------------------
        silhouette_vector[n_clusters-start] = silhouette_score(ts_norm,
           ks_preds)

        print("For n_clusters =", n_clusters,
             "The average silhouette_score is :",
                silhouette_vector[n_clusters-start])

    plt.plot(list_vector,silhouette_vector)
    plt.title("Silhouette scores")
    plt.show()
    #.................................................................
#=======================================================================

#=======================================================================
K_temp = int(input("What K do you chose? (K_temp): "))

#------------------------- K-SHAPE ----------------------------------
# Generate random integer (starting point) for K-Shape algorithm
```

```
seed = 0
np.random.seed(seed)

# Cluster data into n_clusters number of clusters
ks = KShape(K_temp,random_state=seed,n_init=10)
ks.fit(ts_norm)

# Extract centroids
ks_centroids = ks.cluster_centers_

# Assign each time series to each centroid
ks_preds = ks.fit_predict(ts_norm)
#----------------------------------------------------------------------

#-------------------------- PLOT EACH CLUSTER -------------------------
rows, cols = ts.shape
plot_index = np.transpose(np.linspace(0,rows-1,rows))

for i in range(K_temp):

    plt.figure()
    for xx in ts_norm[ks_preds == i]:
        plt.plot(xx.ravel(), "k-", alpha=.2)
    plt.plot(ks_centroids[i].ravel(), "r-")
    plt.xlim(0, cols)
    plt.ylim(-4, 4)
    plt.title("Cluster %d" % (i + 1))

    plt.tight_layout()
    plt.show()
#......................................................................
#======================================================================
```

C.2 Effect of changing number of time series in one time series group

This is the code used for executing the experiment described in Section 9.2.3.

Generating sine and square waves

```
import numpy as np
import matplotlib.pyplot as plt
from random import randint
import math
#------------------------------- GENERAL -----------------------------
time_end = 275
time_vector = np.linspace(0,time_end-1,time_end)
```

```
SQUARE_size = 50
SINUS_size = 3
#.....................................................................

#--------------------------- SQUARE WAVE ------------------------------
AS BEFORE
#.....................................................................

#---------------------------- SINE WAVE -----------------------------
AS BEFORE
#....................................................................

#------------------------ FINAL SET ---------------------------------
AS BEFORE
#.....................................................................
```

Performing Silhouette analysis and K-Shape clustering

```
#======================================================================
import numpy as np
from tslearn.datasets import CachedDatasets
from tslearn.preprocessing import TimeSeriesScalerMeanVariance
from tslearn.clustering import silhouette_score
from tslearn.clustering import KShape
import matplotlib.pyplot as plt
#======================================================================

#======================================================================
#---------------------------- DATA SET --------------------------------
AS BEFORE
#.....................................................................

#------------------------------ PLOT ----------------------------------
AS BEFORE
#.....................................................................
#======================================================================

#======================================================================
Test_silhouette = int(input("Do you want to test silhouette score for
   multiple K? Type 1 for yes, and 0 for no: "))

if Test_silhouette:

    AS BEFORE

    for n_clusters in range(start,stop):
    #------------------------ K-SHAPE --------------------------------
```

```
        # Generate random integer (starting point) for K-Shape algorithm
        seed = 0
        np.random.seed(seed)

        # Cluster data into n_clusters number of clusters
        ks = KShape(n_clusters,random_state=seed,n_init=600)
        ks.fit(ts_norm)

        # Extract centroids
        ks_centroids = ks.cluster_centers_

        # Assign each time series to each centroid
        ks_preds = ks.fit_predict(ts_norm)
           #------------------------------------------------------------------

    #---------------------------- SILHOUETTE -------------------------
      AS BEFORE
    #...................................................................
#======================================================================

#======================================================================
K_temp = int(input("What K do you chose? (K_temp): "))

#-------------------------- K-SHAPE -----------------------------------
AS BEFORE
#----------------------------------------------------------------------

#-------------------------- PLOT EACH CLUSTER -------------------------
AS BEFORE
#......................................................................
#======================================================================
```

C.3 Testing time invariance of periodic time series

This is the code used for executing the experiment described in Section 9.2.4.

Generating sine and square waves

```
import numpy as np
import matplotlib.pyplot as plt
from random import randint
import math
#------------------------------ GENERAL -----------------------------
time_end = 150
time_vector = np.linspace(0,time_end-1,time_end)

SQUARE_size = 50
SINUS_size = 50
```

```
#.......................................................................

#--------------------------- SQUARE WAVE ------------------------------
AS BEFORE
#.......................................................................

#---------------------------- SINE WAVE ------------------------------
AS BEFORE
#......................................................................

#------------------------ FINAL SET ----------------------------------
Neg_SIN_array = np.negative(SINUS_array)

final_set = np.concatenate((SINUS_array,SQUARE_array,Neg_SIN_array))

row,col = final_set.shape

plt.figure()
for xx in range(row):
    plt.plot(final_set[xx,:], "k-", alpha=.2)
plt.title("final")
plt.tight_layout()
plt.show()

open('Mixed_TS.txt', 'w').close()
np.savetxt("Mixed_TS.txt", final_set, delimiter = ",")
#.......................................................................
```

Performing Silhouette analysis and K-Shape clustering

```
#=======================================================================
import numpy as np
from tslearn.datasets import CachedDatasets
from tslearn.preprocessing import TimeSeriesScalerMeanVariance
from tslearn.clustering import silhouette_score
from tslearn.clustering import KShape
import matplotlib.pyplot as plt
#=======================================================================

#=======================================================================
#---------------------------- DATA SET --------------------------------
ts = np.loadtxt('Mixed_TS.txt',delimiter=',')
np.random.shuffle(ts)
ts_norm = TimeSeriesScalerMeanVariance().fit_transform(ts)
#.......................................................................

#------------------------------ PLOT ----------------------------------
AS BEFORE
```

```
#.........................................................................
#=========================================================================

#=========================================================================
Test_silhouette = int(input("Do you want to test silhouette score for
   multiple K? Type 1 for yes, and 0 for no: "))

if Test_silhouette:

    AS BEFORE

    for n_clusters in range(start,stop):
    #------------------------- K-SHAPE --------------------------------
        # Generate random integer (starting point) for K-Shape algorithm
        seed = 0
        np.random.seed(seed)

        # Cluster data into n_clusters number of clusters
        ks = KShape(n_clusters,random_state=seed,n_init=200)
        ks.fit(ts_norm)

        # Extract centroids
        ks_centroids = ks.cluster_centers_

        # Assign each time series to each centroid
        ks_preds = ks.fit_predict(ts_norm)
            #------------------------------------------------------------------

    #---------------------------- SILHOUETTE -------------------------
      AS BEFORE
    #.....................................................................
#=========================================================================

#=========================================================================
K_temp = int(input("What K do you chose? (K_temp): "))

#-------------------------- K-SHAPE -----------------------------------
AS BEFORE
#-------------------------------------------------------------------------

#-------------------------- PLOT EACH CLUSTER -------------------------
AS BEFORE
#.........................................................................
#=========================================================================
```

C.4 Effect of changing length of data set

This is the code used for executing the experiment described in Section 9.2.5.

Generating sine and square waves

```
import numpy as np
import matplotlib.pyplot as plt
from random import randint
import math
#------------------------------- GENERAL ----------------------------
AS BEFORE
#.......................................................................

#---------------------------- SQUARE WAVE ------------------------------
AS BEFORE
#.......................................................................

#---------------------------- SINE WAVE -----------------------------
AS BEFORE
#......................................................................

#------------------------- FINAL SET ----------------------------------
AS BEFORE
#.......................................................................
```

Performing Silhouette analysis and K-Shape clustering

```
#=======================================================================
import numpy as np
from tslearn.datasets import CachedDatasets
from tslearn.preprocessing import TimeSeriesScalerMeanVariance
from tslearn.clustering import silhouette_score
from tslearn.clustering import KShape
import matplotlib.pyplot as plt
#=======================================================================

#=======================================================================
#---------------------------- DATA SET ------------------------------
ts = np.loadtxt('Mixed_TS.txt',delimiter=',')
ts = ts[:,20:155]
np.random.shuffle(ts)
ts_norm = TimeSeriesScalerMeanVariance().fit_transform(ts)
#.......................................................................

#------------------------------ PLOT ----------------------------------
AS BEFORE
#.......................................................................
#=======================================================================

#=======================================================================
Test_silhouette = int(input("Do you want to test silhouette score for
   multiple K? Type 1 for yes, and 0 for no: "))

if Test_silhouette:

   AS BEFORE

   for n_clusters in range(start,stop):
   #------------------------ K-SHAPE -------------------------------
       # Generate random integer (starting point) for K-Shape algorithm
       seed = 0
       np.random.seed(seed)

       # Cluster data into n_clusters number of clusters
       ks = KShape(n_clusters,random_state=seed,n_init=400)
       ks.fit(ts_norm)

       # Extract centroids
       ks_centroids = ks.cluster_centers_

       # Assign each time series to each centroid
       ks_preds = ks.fit_predict(ts_norm)
   #-----------------------------------------------------------------

   #---------------------------- SILHOUETTE -------------------------
     AS BEFORE
   #.................................................................
#=====================================================================

#=====================================================================
K_temp = int(input("What K do you chose? (K_temp): "))

#-------------------------- K-SHAPE ----------------------------------
AS BEFORE
#---------------------------------------------------------------------

#-------------------------- PLOT EACH CLUSTER ------------------------
AS BEFORE
#.....................................................................
#=====================================================================
```

C.5 Effect of adding randomness to the data set

This is the code used for executing the experiment described in Section 9.2.6.

Generating sine and square waves

```
import numpy as np
import matplotlib.pyplot as plt
```

```
from random import randint
import math
#-------------------------------- GENERAL -----------------------------
AS BEFORE
#.......................................................................

#---------------------------- SQUARE WAVE ------------------------------
AS BEFORE
#.......................................................................

#----------------------------- SINE WAVE ------------------------------
AS BEFORE
#......................................................................

#------------------------- FINAL SET ----------------------------------
AS BEFORE
#.......................................................................
```

Performing Silhouette analysis and K-Shape clustering

```
#=======================================================================
import numpy as np
from tslearn.datasets import CachedDatasets
from tslearn.generators import random_walks
from tslearn.preprocessing import TimeSeriesScalerMeanVariance
from tslearn.clustering import silhouette_score
from tslearn.clustering import KShape
import matplotlib.pyplot as plt
#=======================================================================

#=======================================================================
#----------------------------- DATA SET --------------------------------
ts_2 = np.loadtxt('Mixed_TS.txt',delimiter=',')
np.random.shuffle(ts_2)

X_train, y_train, X_test, y_test = CachedDatasets().load_dataset("Trace")
for i in range(np.random.randint(0,20)):
    np.random.shuffle(X_train)
    random_ts = random_walks(n_ts = 25, sz = 275)
ts_1 = X_train[:,:,0]
ts_3 = random_ts[:,:,0]

ts = np.concatenate((ts_1,ts_2,ts_3), axis = 0)
ts_norm = TimeSeriesScalerMeanVariance().fit_transform(ts)
#.......................................................................

#------------------------------- PLOT ----------------------------------
AS BEFORE
```

```
#.......................................................................
#=======================================================================

#=======================================================================
Test_silhouette = int(input("Do you want to test silhouette score for
   multiple K? Type 1 for yes, and 0 for no: "))

if Test_silhouette:

    AS BEFORE

    for n_clusters in range(start,stop):
    #------------------------- K-SHAPE --------------------------------
        # Generate random integer (starting point) for K-Shape algorithm
        seed = 0
        np.random.seed(seed)

        # Cluster data into n_clusters number of clusters
        ks = KShape(n_clusters,random_state=seed,n_init=200)
        ks.fit(ts_norm)

        # Extract centroids
        ks_centroids = ks.cluster_centers_

        # Assign each time series to each centroid
        ks_preds = ks.fit_predict(ts_norm)
    #------------------------------------------------------------------

    #---------------------------- SILHOUETTE -------------------------
      AS BEFORE
    #.................................................................
#=======================================================================

#=======================================================================
K_temp = int(input("What K do you chose? (K_temp): "))

#-------------------------- K-SHAPE -----------------------------------
AS BEFORE
#----------------------------------------------------------------------

#-------------------------- PLOT EACH CLUSTER -------------------------
AS BEFORE
#.......................................................................
#=======================================================================
```

D Appendix IV: Customer segmentation programme

D.1 FullMethod.py

```
#======================== IMPORT FUNCTIONS ============================
from OD12 import OD12
from OD11 import OD11
from OD2 import OD2
from OD3 import OD3
from KShape import K_Shape
from FindKoptimal import FindKoptimal
#======================================================================

#======================== IMPORT LIBRARIES ============================
import numpy as np
import matplotlib.pyplot as plt
from tslearn.preprocessing import TimeSeriesScalerMeanVariance
#======================================================================

#=============================== DATA =================================
#------------------------ IMPORT DATA SET -----------------------------

## Upload data set
ts = np.loadtxt('???.txt',delimiter=',')
np.random.shuffle(ts)
ts_norm = TimeSeriesScalerMeanVariance().fit_transform(ts)
#.....................................................................

#------------------------ PLOT DATA SET ------------------------------
row,col = ts.shape

plt.figure()
for xx in range(row):
   plt.plot(ts[xx,:], "k-", alpha=.2)
plt.title("Time series data set")
plt.tight_layout()
plt.show()

ts_norm_plot = ts_norm[:,:,0]
plt.figure()
for xx in range(row):
   plt.plot(ts_norm_plot[xx,:], "k-", alpha=.2)
plt.title("Time series after normalization")
plt.tight_layout()
plt.show()
#.....................................................................
#======================================================================

#============================ OUTLIERS ================================
outliers = np.array([],dtype=np.int64).reshape(0,col)
outliers1 = np.array([])
outliers2 = np.array([])
outliers3 = np.array([])
outliers4 = np.array([])
#========================================================================

#========================= IC. FOR WHILE LOOP ===========================
refine = True
#========================================================================

#========================= REFINE INLIER SET ============================
while (refine):

    n_init = int(input("Please enter n_init: "))

    ## Find K_optimal with Silhouette algorithm
    Test_silhouette = int(input("Do you want to test silhouette score for
       multiple K? Type 1 for yes, and 0 for no: "))
    if Test_silhouette:
            FindKoptimal(ts_norm,n_init)

    ## Cluster data with K-Shape algorithm
    K_temp = int(input("What K do you choose?: "))
    ks_centroids,ks_preds = K_Shape(K_temp,ts_norm,ts,n_init)

    #--------------------- OUTLIER ANALYSIS ---------------------------
    print(" ")
    print("You will now get options to remove outliers in the following
       ways:")
    print("1) OD1: Remove undesirable clusters.")
    print("2) OD1: Remove sparse clusters.")
    print("3) OD2: Remove outliers based on extra-cluster dissimilarity.")
    print("4) OD3: Remove outliers based on intra-cluster dissimilarity.")
    print(" ")
    print("Any time series that are removed are not deleted, only moved
       from")
    print("inlier to outlier category. After outliers are removed the
       clusters")
    print("will be re-calculated. You may also choose to check the
       Silhouette")
    print("scores.")
    print(" ")
    print("What would you like to do?")
    print("Type 1 for OD1: Remove undesirable clusters.")
    print("Type 2 for OD1: Remove sparse clusters.")
    print("Type 3 for OD2: Remove outliers based on extra-cluster
       dissimilarity.")
    print("Type 4 for OD3: Remove outliers based on intra-cluster
       dissimilarity.")
    print("Type 5 to not perform outlier analysis, try new clustering")
    print("Type 0 to exit the programme")
    choice = int(input("Please enter a number here: "))



    if (choice == 2):
       ts,ts_norm,outliers1 =
            OD12(K_temp,ks_preds,ks_centroids,ts_norm,ts)
       if outliers1.size != 0:
            outliers = np.vstack([outliers,outliers1])

    if (choice == 1):
       ts,ts_norm,outliers2 = OD11(ks_preds,ts_norm,ts)
       if outliers2.size != 0:
            outliers = np.vstack([outliers,outliers2])

    if (choice==3):
       ts,ts_norm,outliers3 = OD2(ts,ts_norm,ks_centroids,K_temp)
       if outliers3.size != 0:
            outliers = np.vstack([outliers,outliers3])

    if (choice == 4):
       ts,ts_norm,outliers4 = OD3(K_temp,ts_norm,ks_preds,ts,ks_centroids)
       if outliers4.size != 0:
            outliers = np.vstack([outliers,outliers4])

    if (choice == 5):
       print("Back to clustering.")

    ##---------------------------- PLOT -----------------------------
    row,col = outliers.shape
    plt.figure()
    for xx in range(row):
       plt.plot(outliers[xx,:], "k-", alpha=.2)
    plt.title("Selected outliers, not normalized")
    plt.tight_layout()
    plt.show()

    row,col = ts.shape
    plt.figure()
    for xx in range(row):
       plt.plot(ts[xx,:], "k-", alpha=.2)
    plt.title("Time series inlier data set")
    plt.tight_layout()
    plt.show()

    ts_norm_plot = ts_norm[:,:,0]
```

```
    plt.figure()
    for xx in range(row):
        plt.plot(ts_norm_plot[xx,:], "k-", alpha=.2)
    plt.title("Time series inliers after normalization")
    plt.tight_layout()
    plt.show()

    #.....................................................................

    open('outliers.txt', 'w').close()
    open('inliers.txt', 'w').close()
    np.savetxt("outliers.txt", outliers, delimiter=";")
    np.savetxt("inliers.txt", ts, delimiter=";")


    if (choice==0):
        break
#========================================================================
```

D.2 FindKoptimal.py

```
import numpy as np
from tslearn.clustering import silhouette_score
from tslearn.clustering import KShape
import matplotlib.pyplot as plt


def FindKoptimal(ts_norm,n_in):

    start = int(input("Start at which K: "))
    stop = int(input("Stop at which K: "))

    silhouette_vector = [0]*(stop-start)
    list_vector = list(range(start,stop))

    for n_clusters in range(start,stop):
    #------------------------ K-SHAPE ------------------------------------
        # Generate random integer (starting point) for K-Shape algorithm
        seed = 0
        np.random.seed(seed)

        # Cluster data into n_clusters number of clusters
        ks = KShape(n_clusters,max_iter=100,random_state=seed,n_init=n_in)
        ks.fit(ts_norm)

        # Assign each time series to each centroid
        ks_preds = ks.fit_predict(ts_norm)
    #----------------------------------------------------------------------
    #-------------------------- SILHOUETTE -------------------------------
        silhouette_vector[n_clusters-start] = silhouette_score(ts_norm,
            ks_preds)

        print("For n_clusters =", n_clusters,
              "The average silhouette_score is :",
                  silhouette_vector[n_clusters-start])

    plt.plot(list_vector,silhouette_vector)
    plt.title("Silhouette scores")
    plt.show()
    #.........................................................................
```

D.3 KShape.py

```
import numpy as np
from tslearn.clustering import KShape
import matplotlib.pyplot as plt

def K_Shape(K_temp,ts_norm,ts,n_in):
    #------------------------ K-SHAPE -----------------------------------
    # Generate random integer (starting point) for K-Shape algorithm
    seed = 0
    np.random.seed(seed)

    # Cluster data into n_clusters number of clusters
    ks = KShape(K_temp,max_iter=100,random_state=seed,n_init=n_in)
    ks.fit(ts_norm)

    # Extract centroids
    ks_centroids = ks.cluster_centers_

    # Assign each time series to each centroid
    ks_preds = ks.fit_predict(ts_norm)
    #----------------------------------------------------------------------

    #------------------------ PLOT EACH CLUSTER --------------------------
    rows, cols = ts.shape

    for i in range(K_temp):

        plt.figure()
        for xx in ts_norm[ks_preds == i]:
            plt.plot(xx.ravel(), "k-", alpha=.2)
        plt.plot(ks_centroids[i].ravel(), "r-")
        plt.title("Cluster %d" % (i + 1))
        plt.show()
    #......................................................................

    return ks_centroids,ks_preds
```

D.4 OD11.py

```
import numpy as np
from tslearn.preprocessing import TimeSeriesScalerMeanVariance


inliers = 0
inliers_norm = 0


def OD11(ks_preds,ts_norm,ts):

    entire = True
    if(entire):

        which_cluster = int(input("Which cluster do you want to remove?:
            "))

        outliers_norm = ts_norm[ks_preds == (which_cluster-1)]
        inliers_norm = ts_norm[ks_preds != (which_cluster-1)]

        outliers = ts[ks_preds == (which_cluster-1)]
        inliers = ts[ks_preds != (which_cluster-1)]

        more = int(input("Do you want to remove more clusters? 1=Y, 0=N:
            "))
        while (more):

            which_cluster = int(input("Which cluster do you want to
                remove?: "))

            inliers_norm = ts_norm[ks_preds != (which_cluster-1)]
            inliers = ts[ks_preds != (which_cluster-1)]

            outliers2 = ts[ks_preds == (which_cluster-1)]
            outliers = np.concatenate((outliers,outliers2))
            outliers_norm =
                TimeSeriesScalerMeanVariance().fit_transform(outliers)

            more = int(input("Do you want to remove more clusters? 1=Y,
                0=N: "))
            if (more == 0):
                break
    return inliers,inliers_norm,outliers
```

D.5 OD12.py

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from tslearn.preprocessing import TimeSeriesScalerMeanVariance
from tslearn.metrics import cdist_dtw
import math

def OD12(K_temp,ks_preds,ks_centroids,ts_norm,ts):

    sparse = True
    if (sparse):
        #======================== SPARSE ===============================
        #-------------- HOW MANY TS DOES EACH CLUSTER HAVE? ------------
        clusters = np.zeros((1,K_temp),dtype=int)
        for i in range(K_temp):
            for j in ks_preds:
                if (i == j):
                    clusters[0,i] = clusters[0,i] + 1

        index = np.arange(start=1,stop=(K_temp+1),step=1)

        plt.bar(index,clusters[0,:])
        plt.title("Number of time series in each cluster")
        plt.show()

        #..............................................................

        #--------------------------- DTW ------------------------------
        rows, cols = ts.shape
        plot_index = np.transpose(np.linspace(0,rows-1,rows))

        for i in range(0,K_temp):
            dtw_vector = cdist_dtw(ts_norm,np.transpose(ks_centroids[i,:]))
            dtw_index = np.c_[dtw_vector,plot_index]
            dtw_sorted = dtw_index[dtw_index[:,0].argsort()]

            if i == 0:
                dtw_storage = dtw_sorted
            else:
                dtw_storage = np.c_[dtw_storage,dtw_sorted]

        index = np.arange(start = 0, stop = (2*K_temp-1), step = 2 )
        dtw_max = np.amax(dtw_storage[:,(index)])
        dtw_min = np.amin(dtw_storage[:,(index)])
        #..............................................................

        # Set the boundary for which clusters to remove
        boundary = int(input("What sparseness boundary?: "))

        #----------------- REMOVE SPARSE CLUSTERS --------------------

        remove_c = np.where(clusters <= boundary)

        index_kspreds = np.array([])
        for i in remove_c[1]:
            index_kspreds = np.append(index_kspreds,i)

        outliers_bool = np.isin(ks_preds,index_kspreds)
        inliers_bool = np.invert(outliers_bool)

        data_frame_outliers = pd.DataFrame(outliers_bool, columns =
            ['corona'])
        data_frame_inliers = pd.DataFrame(inliers_bool, columns =
            ['corona'])
        data_frame_org = pd.DataFrame(ts)

        data_frame_outliers =
            data_frame_org.loc[data_frame_outliers.corona,:]
        data_frame_inliers =
            data_frame_org.loc[data_frame_inliers.corona,:]

        inliers = data_frame_inliers.to_numpy()
        outliers = data_frame_outliers.to_numpy()

        inliers_norm =
            TimeSeriesScalerMeanVariance().fit_transform(inliers)
        outliers_norm =
            TimeSeriesScalerMeanVariance().fit_transform(outliers)
        #.............................................................

    return inliers,inliers_norm,outliers
```

D.6 OD2.py

```
import numpy as np
from tslearn.datasets import CachedDatasets
from tslearn.generators import random_walks
from tslearn.preprocessing import TimeSeriesScalerMeanVariance
from tslearn.clustering import silhouette_score
from tslearn.clustering import KShape
```

```
import matplotlib.pyplot as plt
from tslearn.metrics import cdist_dtw
import math
import pandas as pd


def OD2(ts,ts_norm,ks_centroids,K_temp):

    Outra_cluster = True

    ts_copy = ts
    ts_norm_copy = ts_norm
    #========================== FIRST TIME =============================
    #---------------------------- DTW ---------------------------------
    rows, cols = ts.shape
    plot_index = np.transpose(np.linspace(0,rows-1,rows))


    dtw_vector=0
    for i in range(0,K_temp):
        dtw_vector = cdist_dtw(ts_norm,np.transpose(ks_centroids[i,:]))
        dtw_index = np.c_[dtw_vector,plot_index]
        dtw_sorted = dtw_index[dtw_index[:,0].argsort()]

        if i == 0:
            dtw_storage = dtw_sorted
        else:
            dtw_storage = np.c_[dtw_storage,dtw_sorted]

    index = np.arange(start = 0, stop = (2*K_temp-1), step = 2 )

    dtw_max = np.amax(dtw_storage[:,(index)])
    dtw_min = np.amin(dtw_storage[:,(index)])
    #...................................................................

    #--------------- PLOT DISTRIBUTION OF CLUSTER COHERENCE --------------

    counter = 0
    plt.figure()
    for i in range(0,(2*K_temp-1),2):
        #plt.subplot(K_temp)
        counter = counter + 1
        plt.hist(dtw_storage[:,i],alpha = 0.4,bins =
            (math.ceil(dtw_max)-math.floor(dtw_min)), label = 'Cluster %d'
            % counter)
    plt.legend()
    plt.title("Histogram of cluster similarity distribution of %d
        clusters" % K_temp)
    plt.tight_layout()
    plt.show()

    #.......................................................................

    boundary = float(input("What is your DTW boundary? Type in a number:
        "))

    for i in range(1,(K_temp*2),2):
        arr = np.where(dtw_storage[:,i-1] > boundary)

        if (i==1):
            index_storage = dtw_storage[arr,i]
        else:
            index_storage = np.append(index_storage,dtw_storage[arr,i])

    data_frame_index = pd.DataFrame(index_storage, columns = ['corona'])
    data_frame_index =
        data_frame_index[data_frame_index.duplicated(keep="first")]

    data_frame_org = pd.DataFrame(ts)

    data_frame_outliers =
        data_frame_org.iloc[data_frame_index["corona"],:]
    data_frame_inliers = data_frame_org.drop(data_frame_index["corona"],
        axis=0)

    ts = data_frame_inliers.to_numpy()
    ts_norm = TimeSeriesScalerMeanVariance().fit_transform(ts)

    outliers = data_frame_outliers.to_numpy()
    outliers_norm = TimeSeriesScalerMeanVariance().fit_transform(outliers)
    #-------------------------- PLOT INLIERS -----------------------------
    plot_inliers_arr = ts_norm[:,:,0]
    row,col = plot_inliers_arr.shape

    plt.figure()
    for xx in range(row):
        plt.plot(plot_inliers_arr[xx,:],"k-",alpha=.2)
    plt.title("Inliers")
    plt.tight_layout()
    plt.show()
    #.......................................................................
    #-------------------------- PLOT OUTLIERS ----------------------------
    plot_outliers_arr = outliers_norm[:,:,0]
    row,col = plot_outliers_arr.shape

    plt.figure()
    for xx in range(row):
        plt.plot(plot_outliers_arr[xx,:],"k-",alpha=.2)
    plt.title("Outliers")
    plt.tight_layout()
    plt.show()
    #..............................................................
    Outra_cluster = int(input("Do you want to continue trying other DTW
       boundaries? Type 0 to keep changes, type 1 to test new boundaries:
       "))
    #======================================================================


    #========================= OTHER ITERATIONS =========================
    while Outra_cluster:

       #-------------------------- DTW ------------------------------
       rows, cols = ts_copy.shape
       plot_index = np.transpose(np.linspace(0,rows-1,rows))


       dtw_vector=0
       for i in range(0,K_temp):
          dtw_vector =
             cdist_dtw(ts_norm_copy,np.transpose(ks_centroids[i,:]))
          dtw_index = np.c_[dtw_vector,plot_index]
          dtw_sorted = dtw_index[dtw_index[:,0].argsort()]

          if i == 0:
             dtw_storage = dtw_sorted
          else:
             dtw_storage = np.c_[dtw_storage,dtw_sorted]

       index = np.arange(start = 0, stop = (2*K_temp-1), step = 2 )

       dtw_max = np.amax(dtw_storage[:,(index)])
       dtw_min = np.amin(dtw_storage[:,(index)])
       #..........................................................

       #------------- PLOT DISTRIBUTION OF CLUSTER COHERENCE ------------
       counter = 0
       plt.figure()
       for i in range(0,(2*K_temp-1),2):
          #plt.subplot(K_temp)
          counter = counter + 1
          plt.hist(dtw_storage[:,i],alpha = 0.4,bins =
             (math.ceil(dtw_max)-math.floor(dtw_min)), label = 'Cluster
             %d' % counter)
       plt.legend()
       plt.title("Histogram of cluster similarity distribution of %d
          clusters" % K_temp)
       plt.tight_layout()
    plt.show()

    #..........................................................

    boundary = float(input("What is your DTW boundary? Type in a
       number: "))

    for i in range(1,(K_temp*2),2):
        arr = np.where(dtw_storage[:,i-1] > boundary)

        if (i==1):
            index_storage = dtw_storage[arr,i]
        else:
            index_storage = np.append(index_storage,dtw_storage[arr,i])

    data_frame_index = pd.DataFrame(index_storage, columns =
        ['corona'])
    data_frame_index =
       data_frame_index[data_frame_index.duplicated(keep="first")]

    data_frame_org = pd.DataFrame(ts_copy)

    data_frame_outliers =
       data_frame_org.iloc[data_frame_index["corona"],:]
    data_frame_inliers =
       data_frame_org.drop(data_frame_index["corona"], axis=0)

    ts = data_frame_inliers.to_numpy()
    ts_norm = TimeSeriesScalerMeanVariance().fit_transform(ts)

    outliers = data_frame_outliers.to_numpy()
    outliers_norm =
       TimeSeriesScalerMeanVariance().fit_transform(outliers)

    #------------------------- PLOT INLIERS --------------------------
    plot_inliers_arr = ts_norm[:,:,0]
    row,col = plot_inliers_arr.shape

    plt.figure()
    for xx in range(row):
        plt.plot(plot_inliers_arr[xx,:],"k-",alpha=.2)
    plt.title("Inliers")
    plt.tight_layout()
    plt.show()
    #..........................................................
    #------------------------ PLOT OUTLIERS --------------------------
    plot_outliers_arr = outliers_norm[:,:,0]
    row,col = plot_outliers_arr.shape
```

```
        plt.figure()
        for xx in range(row):
            plt.plot(plot_outliers_arr[xx,:],"k-",alpha=.2)
        plt.title("Outliers")
        plt.tight_layout()
        plt.show()
        #.................................................................
        Outra_cluster = int(input("Do you want to continue trying other
            DTW boundaries? Type 0 to keep changes, type 1 to test new
            boundaries: "))
        #=================================================================



    return ts,ts_norm,outliers
```

D.7 OD3.py

```
import numpy as np
from tslearn.preprocessing import TimeSeriesScalerMeanVariance
import matplotlib.pyplot as plt
from tslearn.metrics import cdist_dtw
import math
import pandas as pd

def OD3(K_temp,ts_norm,ks_preds,ts,ks_centroids):

    size_of_clusters = np.array([])

    for i in range(0,K_temp):
        dtw_vector = cdist_dtw(ts_norm[ks_preds ==
            i],np.transpose(ks_centroids[i,:]))
        cluster_index_tuple = np.where(ks_preds == i)

        cluster_index = np.array([])
        for j in cluster_index_tuple[0]:
            cluster_index = np.append(cluster_index,j)
        size_of_clusters = np.append(size_of_clusters,cluster_index.size)

        dtw_index = np.c_[dtw_vector,cluster_index]
        dtw_sorted = dtw_index[dtw_index[:,0].argsort()]


        if i == 0:
            dtw_storage = dtw_sorted
        else:
            dtw_storage = np.concatenate((dtw_storage,dtw_sorted),axis = 0)
```

```
dtw_storage_sorted = dtw_storage[dtw_storage[:,1].argsort()]
dtw_max = np.amax(dtw_storage[:,0])
dtw_min = np.amin(dtw_storage[:,0])

plt.figure()
for i in range(K_temp):
    plt.hist(dtw_storage_sorted[ks_preds==i,0],alpha = 0.4,bins =
        (math.ceil(dtw_max)-math.floor(dtw_min)))
    plt.title("Histogram of cluster similarity distribution of %d
        clusters" % K_temp)
    plt.legend("%d" % (i+1))
plt.tight_layout()
plt.show()

print("You will now be asked to provide a DTW boundary for removing
    outliers")
print("based on their DTW score compared to their assigned cluster.")

boundary = float(input("What is your DTW boundary? Type in a number:
    "))

outliers = dtw_storage[:,0] > boundary
inliers = np.invert(outliers)

data_frame_outliers = pd.DataFrame(outliers, columns = ['corona'])
data_frame_inliers = pd.DataFrame(inliers, columns = ['corona'])
data_frame_org = pd.DataFrame(ts)

data_frame_outliers = data_frame_org.loc[data_frame_outliers.corona,:]
data_frame_inliers = data_frame_org.loc[data_frame_inliers.corona,:]

ts = data_frame_inliers.to_numpy()
ts_norm = TimeSeriesScalerMeanVariance().fit_transform(ts)

outliers = data_frame_outliers.to_numpy()
outliers_norm = TimeSeriesScalerMeanVariance().fit_transform(outliers)

#--------------------------- PLOT INLIERS ----------------------------
rows,cols = ts.shape
plot_index = np.transpose(np.linspace(0,rows-1,rows))

plt.figure()
for index,row in data_frame_inliers.iterrows():
    plt.plot(row.ravel(),"k-",alpha=.2)
plt.title("Inliers")
plt.show()
#.....................................................................
#---------------------------- PLOT OUTLIERS --------------------------
rows,cols = ts.shape
```

```
plot_index = np.transpose(np.linspace(0,rows-1,rows))

plt.figure()
for index,row in data_frame_outliers.iterrows():
    plt.plot(row.ravel(),"k-",alpha=.2)
plt.title("Outliers")
plt.show()
#......................................................................

Intra_cluster = int(input("Do you want to continue trying other DTW
    boundaries? Type 0 to keep changes, type 1 to test new boundaries:
    "))

while Intra_cluster:

    #------------- PLOT DISTRIBUTION OF CLUSTER COHERENCE ------------
    plt.figure()
    for i in range(K_temp):
        plt.hist(dtw_storage_sorted[ks_preds==i,0],alpha = 0.4,bins =
            (math.ceil(dtw_max)-math.floor(dtw_min)))
        plt.title("Histogram of cluster similarity distribution of %d
            clusters" % K_temp)
        plt.legend("%d" % (i+1))
    plt.tight_layout()
    plt.show()
    #.................................................................


    boundary = float(input("What is your DTW boundary? Type in a
        number: "))

    outliers2 = dtw_storage[:,0] > boundary
    inliers = np.invert(outliers2)

    data_frame_outliers = pd.DataFrame(outliers2, columns = ['corona'])
    data_frame_inliers = pd.DataFrame(inliers, columns = ['corona'])
    data_frame_org = pd.DataFrame(ts)

    data_frame_outliers =
        data_frame_org.loc[data_frame_outliers.corona,:]
    data_frame_inliers =
        data_frame_org.loc[data_frame_inliers.corona,:]

    ts = data_frame_inliers.to_numpy()
    ts_norm = TimeSeriesScalerMeanVariance().fit_transform(ts)

    outliers2 = data_frame_outliers.to_numpy()
    outliers = np.concatenate((outliers2,outliers))
    outliers_norm =
        TimeSeriesScalerMeanVariance().fit_transform(outliers)

    #------------------------- PLOT INLIERS --------------------------
    rows,cols = ts.shape
    plot_index = np.transpose(np.linspace(0,rows-1,rows))

    plt.figure()
    for index,row in data_frame_inliers.iterrows():
        plt.plot(row.ravel(),"k-",alpha=.2)
    plt.title("Inliers")
    plt.show()
    #.................................................................
    #------------------------ PLOT OUTLIERS --------------------------
    rows,cols = ts.shape
    plot_index = np.transpose(np.linspace(0,rows-1,rows))

    plt.figure()
    for index,row in data_frame_outliers.iterrows():
        plt.plot(row.ravel(),"k-",alpha=.2)
    plt.title("Outliers")
    plt.show()
    #.................................................................

    Intra_cluster = int(input("Do you want to continue trying other
        DTW boundaries? Type 0 to keep changes, type 1 to test new
        boundaries: "))
#=====================================================================

return ts,ts_norm,outliers
```