Master's thesis

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

Espen Waaga

Machine Learning for Automatic Classification of Wind Turbines

Master's thesis in Industriell kybernetikk
Supervisor: Lars Imsland
June 2019

NTNU
Norwegian University of Science and Technology

KONGSBERG

Espen Waaga

Machine Learning for Automatic Classification of Wind Turbines

Master's thesis in Industriell kybernetikk
Supervisor: Lars Imsland
June 2019

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Kunnskap for en bedre verden

Preface

The thesis is written in partial fulfilment of the requirements for the Master's degree in Industrial Cybernetics (2 years master program) at NTNU Trondheim. The work was done during the spring of 2019 and is not a continuation of the project from autumn the year before. It has been a steep learning curve doing this project as I had no prior knowledge about time series or clustering before starting the work in the end of January. The project has been an exciting journey and a great opportunity to learn as much as I have. The first half of this thesis concerns a literature study of clustering techniques for time series clustering. No literature has been provided and everything which is included in the thesis is the results of my own review of the literature during the spring of 2019. The choice and implementation of the different methods is the results of the literature review. Kongsberg Digital provided me with a HP laptop for access to their system and software. The platform provided by Kongsberg Digital is called Kognifai EmPower and is used to extract time series data from turbines within different wind parks in Norway.

The thesis has three listed supervisors. Professor Lars Imsland is my supervisor at NTNU and he has helped me with questions concerning the literature review and put me in contact with the correct people. Professor Adil Rasheed was one of them, who is not a listed supervisor, which provided with helpful discussions. Unfortunately, the method discussed with him did not make it to the thesis but will be included in the future work section. At Kongsberg Digital I've had two co-supervisors: Pierluigi Salvo Rossi and Gerthory Toussaint. Gerthory has been my co-supervisor who has been a great help for introducing me to the Kognifai EmPower platform. Pierluigi is also my co-supervisor and has been my main contact throughout the semester. He has helped me to narrow the scope of the thesis and provided much needed guidance in times of need. I want to thank all my supervisors for their help during my thesis and Kongsberg Digital for providing me with an extensive set of time series to analyse.

Abstract

As wind parks are continuing to grow at an exponential rate, proper automatic classification of wind turbines is required. Clustering techniques to process the real-time data streams of all the individual wind turbines in operations are expected to provide a sufficient level of automation. The objective of this thesis is to do exploratory work for using clustering techniques for wind turbine condition-based monitoring. More specifically, to explore, understand and summarise the practical implications of using clustering algorithms for automatic classification of wind turbines. To solve this objective, a couple of time series clustering techniques have been applied to cluster different data sets which are comprised of either univariate or multivariate time series. These time series are all extracted from an actual wind park located in Norway. A total of four data sets have been clustered; two univariate and two multivariate data sets.

A review of the literature concerning time series clustering has been presented in the thesis and based on the review, a couple of clustering methods has been selected to examine the practical implications of using clustering techniques for automatic classification of wind turbines. The selected methods are the implementation of the hierarchical clustering algorithm in conjunction with either the Euclidean distance or the dynamic time warping (DTW) distance as the similarity measure; this solves the clustering objectives of similarity in time and similarity in shape.

This thesis showed that by clustering the different data sets – either the univariate or multivariate data set – it resulted in automatic classifications of wind turbines based on their dissimilarities - this includes classification of wind turbines experiencing different gearbox temperatures, generator speed, power production and local wind conditions. The thesis also showed that the Euclidean distance could effectively be used in a scaled-up version, whereas the DTW distance cannot because of the quadratic time complexity of the DTW algorithm. For longer time series, the DTW distance can be substituted with the Euclidean distance, as the objectives of similarity in time and similarity in shape are more similar. This thesis forms the basis for further research into the practical implications of using clustering algorithms for automatic classification of wind turbines.

Sammendrag

Ettersom vindparker vokser eksponentielt, er det nødvendig med økende grad av automatisk klassifisering av vindturbiner. Klyngeteknikker for å behandle datastrømmer i sanntid for alle de enkelte vindturbinene i drift er forventet å gi et tilstrekkelig nivå av automatisering. Målet med denne oppgaven er å gjøre utforskende arbeid for bruk av klyngeteknikker for overvåking av vindturbiner. Nærmere bestemt å utforske, forstå og oppsummere de praktiske implikasjonene ved bruk av klyngealgoritmer for automatisk klassifisering av vindturbiner. For å løse dette målet har et par klyngeteknikker for tidsserier blitt anvendt for å gruppere vindturbiner i forskjellige datasett som består av tidsserier med en eller flere variabler. Disse tidsseriene er alle hentet fra en faktisk vindpark i Norge, totalt fire datasett har blitt gruppert; to datasett bestående av tidsserier med bare en parameter, og to datasett bestående av tidsserier med flere parametere.

En gjennomgang av litteraturen om tidsserieklynging har blitt presentert i rapporten og på bakgrunn av denne gjennomgangen har det blitt valgt ut et par klyngemetoder for å utforske de praktiske implikasjonene ved bruk av klyngeteknikker for automatisk klassifisering av vindturbiner. De valgte metodene er implementering av den hierarkiske klyngealgoritmen sammen med enten den euklidske avstanden eller dynamisk tidsfordreining (DTW) avstanden for sammenligning mellom tidsserier; dette grupperer tidsserier basert på enten likhet i tid eller likhet i form.

Denne rapporten viste at ved å implementere klyngeteknikker på de forskjellige datasettene klarte den å klassifisere vindturbiner basert på deres ulikheter – dette inkluderer klassifisering av vindturbiner som opplever forskjellig temperatur i girkassen, generatorhastighet, kraftproduksjon og lokale vindforhold. Rapporten viste også at den euklidske avstanden effektivt kunne brukes i en oppskalert versjon, mens DTW-avstanden ikke kan på grunn av den kvadratiske tidskompleksiteten til DTW-algoritmen. For lengre tidsserier kan DTW-avstanden erstattes med euklidske avstanden, da målene for likhet i tid og likhet i form er mer eller mindre like. Denne oppgaven danner grunnlag for videre forskning av de praktiske implikasjonene ved bruk av klyngealgoritmer for automatisk klassifisering av vindturbiner.

Contents

List of Tables

List of Figures

Abbreviations

DB	=	Database (or data set)
ARMA	=	Autoregressive-Moving-Average Model
HMM	=	Hidden Markov Model
DTW	=	Dynamic Time Warping
DM	=	Dissimilarity Matrix (or Distance Matrix)
ED	=	Euclidean Distance
SSE	=	Sum of Squared Error
MSSSE	=	Mean Sum of SSE or Average SSE
SMA	=	Simple Moving Average (filter)
SMM	=	Simple Moving Median (filter)
LCM	=	Local Cost Matrix (used in the DTW algorithm)

Chapter 1

Introduction

Wind energy is becoming more and more popular and is one of the fastest growing renewable energy resources (Farahani et al., 2012). According to Global Wind Energy Council (2019b), the installed wind capacity at the end of 2018 in America was 11.9GW, which was an increase of 12% from the previous year. Similarly, Global Wind Energy Council (2019a) states that in Africa and the Middle East, during the same time period, had an installed capacity of 926MW, which was more than 300MW from the year before. By 2020, it is estimated that wind power could supply 2.600 TWh, about 11.5 – 12.3% of global electricity supply, increasing to 21.8% by 2030 (Soua et al., 2013).

There is no doubt that wind energy is the future for energy production. With more wind turbines being installed to increase this capacity, the more important it is to maintain healthy conditions of the wind turbines. Consequences for faults and maintenance are miserable for both the market and the owner. It is therefore important to detect faults and failures early on as to minimise the downtime and maximise the productivity (Hossain et al., 2018). Wind turbines are usually installed in remote areas (e.g. offshore) which makes maintenance both hard and costly (Yang et al., 2013). Another consequence of these harsh environments is that wind turbines are more prone to failure (Hossain et al., 2018).

Due to the complexity of wind turbines, there could be a couple of 100 monitoring points required to monitor most of the subsystems of a single wind turbine; multiplying this number for each wind turbine in a wind park, it is obvious that a lot of data is collected and potential of unsupervised learning increases (Zhang and Ma, 2016). As wind parks are continuing to grow at an exponential rate, proper automatic classification of wind turbines is needed. Much effort and focus have been made on developing condition monitoring systems for wind turbines. However, most of these are costly and vary in reliability. A reliable and cost-effective condition-based monitoring technique is still sought today (Yang et al., 2013). In the case of large-size wind parks, manual inspection of every single wind turbine and a corresponding selection of the most effective template for condition-based monitoring is unrealistic and further automation is required.

Clustering techniques to process the real-time data streams of all the individual wind turbines in operations are expected to provide a sufficient level of automation. Time series clustering is a special type of clustering techniques which clusters together similar time series and separate dissimilar ones. Clustering time series of different wind turbines from the same wind park is believed to reveal interesting patterns and might help to obtain any information about the condition of the different wind turbines within the park or the condition of the park as a whole.

1.1 Related work

Clustering time series have become increasingly popular and attracted attention from a wide range of researchers from many different disciplines. For instance, researchers have clustered time series from speaker verification, robot sensor data, financial data, earthquake, gene expression, retail pattern, commercial consumption, brain activity and much more (Zhang et al., 2011). A more detailed review of applications is presented in the literature review in Section 2.2.4.

The applications of time series clustering to wind turbine condition-based monitoring are very limited. The only work found, to the best of the author's knowledge, which implements clustering techniques for wind turbine condition-based monitoring, is done by Zhang et al. (2012). In this paper, they applied data mining algorithms and statistical methods to analyse the jerk data obtained from monitoring the gearbox of a wind turbine. The modified

K-means algorithm was implemented for cluster analysis of the jerk patterns of the 12 sensors from one single gearbox. The shortcoming of the study was the gearbox test condition as it was not actual data from wind parks, but of a faulty gearbox mounted in an artificial environment. The clustering was also limited to univariate time series extracted from the same turbine.

1.2 Objective of the thesis

The objective of this thesis is to do exploratory work for using clustering techniques for wind turbine condition-based monitoring. More specifically, to explore, understand and summarise the practical implications of using clustering algorithms for automatic classification of wind turbines. To explore the practical implications, a carefully selected set of time series clustering techniques will be applied to cluster univariate and multivariate data sets. These time series - which the data sets are comprised of - are extracted from Kongsberg Digital Kognifai EmPower platform and are sensory data from an existing wind park located in Norway.

The practical implications and consistency of the clustering results will be addressed and compared. The choice of appropriate clustering techniques to implement on the data set will be made by performing a comprehensive review of the literature. From reviewing the literature, the scope of the thesis is limited to solving the clustering problem with respect to two different clustering objectives: Similarity in time and similarity in shape. For solving these clustering objectives, Euclidean distance and DTW will be implemented in conjunction with the agglomerative hierarchical clustering algorithm. Obtaining any physical meaning from the clustering results will be the secondary goal of the thesis and will be used to identify the physical implications for using clustering algorithms for automatic classification of wind turbines. To clarify, this thesis will implement a couple of algorithms and evaluate the practical implication for clustering time series acquired from the same wind parks.

1.3 Setup of the report

- In Chapter 2, time series clustering will be reviewed along with additional applications for time series clustering and important considerations. At the end of the chapter, the literature review will be summarised along with an appropriate choice and justification of clustering methods used in the consecutive sections.
- In Chapter 3, the theory behind the implemented methods which are chosen will be provided.
- In Chapter 4, data acquisition using Kongnifai EmPower platform will be addressed along with its limitations and special considerations. Then, different preprocessing techniques will be introduced and explained.
- In Chapter 5, different univariate and multivariate data set will be extracted and presented along with the appropriate preprocessing which was introduced in the previous section.
- In Chapter 6, clustering will be implemented on the univariate time series (or data set) introduced in Section 5.1. The results will be presented and analysed.
- In Chapter 7, clustering will be implemented on the multivariate time series (or data set) introduced in Section 5.2. The results will be presented and analysed.
- In Chapter 8, the practical implications for applying clustering techniques on the different data sets (univariate and multivariate, short and long) will be summarised and discussed.
- In Chapter 9, the conclusion will be presented along with the recommendations for future work.
- In Appendix A, plots of the time series - which are not presented in Chapter 5 - are presented, along with additional time series used for interpretation.
- In Appendix B, additional clustering results for each data set is presented.
- In Appendix C, additional results and remarks which are based on clustering the data set are presented.
- In Appendix D, the Python codes for all implementations are provided.

Chapter 2

Reviewing the literature

In this chapter, the literature regarding time series clustering will be reviewed. The sole purpose of reviewing the literature is to make an educated choice of which clustering techniques to implement on the acquired time series. In Section 2.1 some basic definition and terminology will be introduced before going over to a more detailed review of time series clustering in Section 2.2. In this section, different types of time series clustering, different ways of clustering time series, the challenges associated with time series clustering and a brief review of applications where time series clustering have been applied will be reviewed. In Section 2.3, the major components of time series clustering will be addressed. Finally, in Section 2.4, a summary of the litterateur review will be included followed by the choice and justification of the time series clustering techniques which will be applied to cluster the different time series.

2.1 Basic definition and terminology

There are two primary data types which are common for clustering problem, namely static and dynamic data. Static data refers to feature values which do not change with time or change negligibly (Warren Liao, 2005). Unlike static data, time series is a type of dynamic data where the feature values vary with time. Each data point in a single time series is one observation that is made chronologically (Aghabozorgi et al., 2015). According to Esling and Agon (2012), the definition of time series data is as follows:

Definition 2.1.1. Time series is an ordered sequence of n real-valued variables $F = (f_1, f_2, \ldots, f_n),\ f_i \in R$. A time series is often the result of the observation of an underlying process in the course of which values are collected from measurements made at uniformly spaced time instants and according to a given sampling rate (Esling and Agon, 2012).

Time series data is a type of temporal data which is naturally high dimensional and large in data size (Aghabozorgi et al., 2015). Time series typically comprise of registered continuous real-valued numbers (0.005, 2.998, ...) measured at discrete time intervals. That is, time series are typically purely quantitative variables. One single time series consisting of a large number of data points measured chronologically can also be seen as a single object (Aghabozorgi et al., 2015). Time series can further be divided into two categories, namely univariate and multivariate time series.

Definition 2.1.2. Univariate time series , is the simplest form of temporal data and is a sequence of real numbers collected regularly in time, where each number represents a value. That is, univariate time series is characterised by a single variable changing over time (Iwok and Okpe, 2016).

Definition 2.1.3. Multivariate time series , is an extension of the univariate case which involves two or more variables which are varying over time (Iwok and Okpe, 2016).

During analysis in this paper, the time series will be collected in a data set. A data set is simply a collection of objects. When the objects are time series, some refer to the data set as a *database* (DB) (Iwok and Okpe, 2016). In this thesis, we will refer to a collection of time series as a data set, rather than a database. A data set which is comprised of individual time series can also be referred to an unordered set of time series (Esling and Agon, 2012). Individual time series spans the columns of the data set and the rows are spanned by the discrete time

points; each row can be thought of as a single object. All these objects will be compared and partitioned according to the specific clustering algorithm. To clarify, a univariate data set refers to an unordered set of univariate time series and a multivariate data set refers to an unordered set of multivariate time series.

2.2 Time series clustering

The goal of any cluster analysis is to create groups of objects where objects in different groups are dissimilar to each other and similar if they are in the same group (Trevor et al., 2009; Maini, 2017; James et al., 2013; Wong, 2015). These clustering algorithms have in common that objects in the data set are partitioned into distinct groups based on some kind of similarity or dissimilarity measure (Trevor et al., 2009). For quantitative data features, distance functions are used as a measure of (dis)similarity; for qualitative data features, similarity functions are used (Xu and Tian, 2015a). A special case of clustering is called time series clustering (Aghabozorgi et al., 2015). For this type of clustering, the objects are either univariate time series or multivariate time series. The definition of time series clustering, according to Aghabozorgi et al. (2015), is formulated in the following way:

Definition 2.2.1. Time series clustering , given a data set of *n* time series, DB = $\{F_1, F_2, \ldots, F_n\}$, is the process of unsupervised partitioning of DB into C = $\{C_1, C_2, \ldots, C_k\}$, in such a way that homogeneous time series are grouped together based on a certain similarity measure.

Each of the time series in DB can be seen as a single object and clustering these relatively complex objects can prove to be very advantageous for the discovery of patterns and dissimilarities in time series data sets (Aghabozorgi et al., 2015).

2.2.1 Types of time series clustering

Clustering of time series can be divided into three main categories depending on the type of time series clustering. The proposed categories are as stated by (Aghabozorgi et al., 2015; Huang et al., 2016; Esling and Agon, 2012) (where Huang et al. (2016); Esling and Agon (2012) excludes time point clustering): Whole time series clustering, subsequence clustering and time point clustering.

- *Whole time series clustering* refers to the clustering of individual time series with respect to their similarity. More specifically, algorithms based on this type of clustering tries to group individual time series to a set of clusters, where the set of clusters are typically one of the time series or the means of all time series within that cluster (Aghabozorgi et al., 2015; Esling and Agon, 2012; Rodpongpun et al., 2012).
- *Subsequence clustering*, or subsequence time series clustering (STS), discovers interesting subsequences within a single time series (Rodpongpun et al., 2012; Esling and Agon, 2012; Aghabozorgi et al., 2015). The time series is partitioned and divided into its most similar cluster. This is typically done with what is called a sliding window.
- *Time point* clustering refers to the clustering of time points based on a combination of the similarity of the corresponding value and the temporal proximity of the time points (Rodpongpun et al., 2012). This is different from segmentation as the number of points assigned to a cluster does not necessarily need to be all of them; some of the points can be treated as noise.

Subsequence time series clustering and time point clustering is performed on a single time series. Also, Keogh and Lin (2005) have shown that subsequence clustering does not produce meaningful results. However, after the publication of this paper, there have been many papers which have proposed methods to make this type of clustering meaningful again (Chen, 2005, 2007). Nonetheless, as the objective of this report is to cluster data sets of univariate and multivariate time series, the whole time series clustering approach is the correct choice.

2.2.2 Different ways to cluster time series

Within the whole time series clustering category, clustering of the time series is normally done by three different clustering approaches (Aghabozorgi et al., 2015; Warren Liao, 2005). The first approach is called the **shape-based** approach, or **raw-data-based** approach, as it typically works directly with the raw time series. In this approach, modification of the existing algorithm for static data is done to handle time series data instead. The majority of modification involves replacing the existing similarity/dissimilarity measures for static data with measure suitable

for time series data (Warren Liao, 2005). The second approach is called **feature-based** approach. In a feature-based approach, raw-time series data is converted into the form of static data such that conventional algorithms for static data can be directly used. This involves converting the raw-time series into a feature vector of lower dimension (Aghabozorgi et al., 2015). According to Aghabozorgi et al. (2015), usually, a feature vector of equal length is calculated from each time series followed by Euclidean distance as the similarity measure between these features. The last approach is called **model-based** approach. In this approach raw-time series is fitted to a model (ARMA, HMM, ...) and a conventional clustering algorithm is applied to the extracted model parameters (Aghabozorgi et al., 2015; Warren Liao, 2005). However, this approach has been shown by (Vlachos et al., 2004) and (Mitsa, 2010) to have scalability problems and reduced performance when clusters are close to each other, respectively. The different clustering approaches are visualised in Figure 2.1. A multi-step approach is also included in the figure, but will not be addressed in this thesis (for more information refer to Aghabozorgi et al. (2015)).

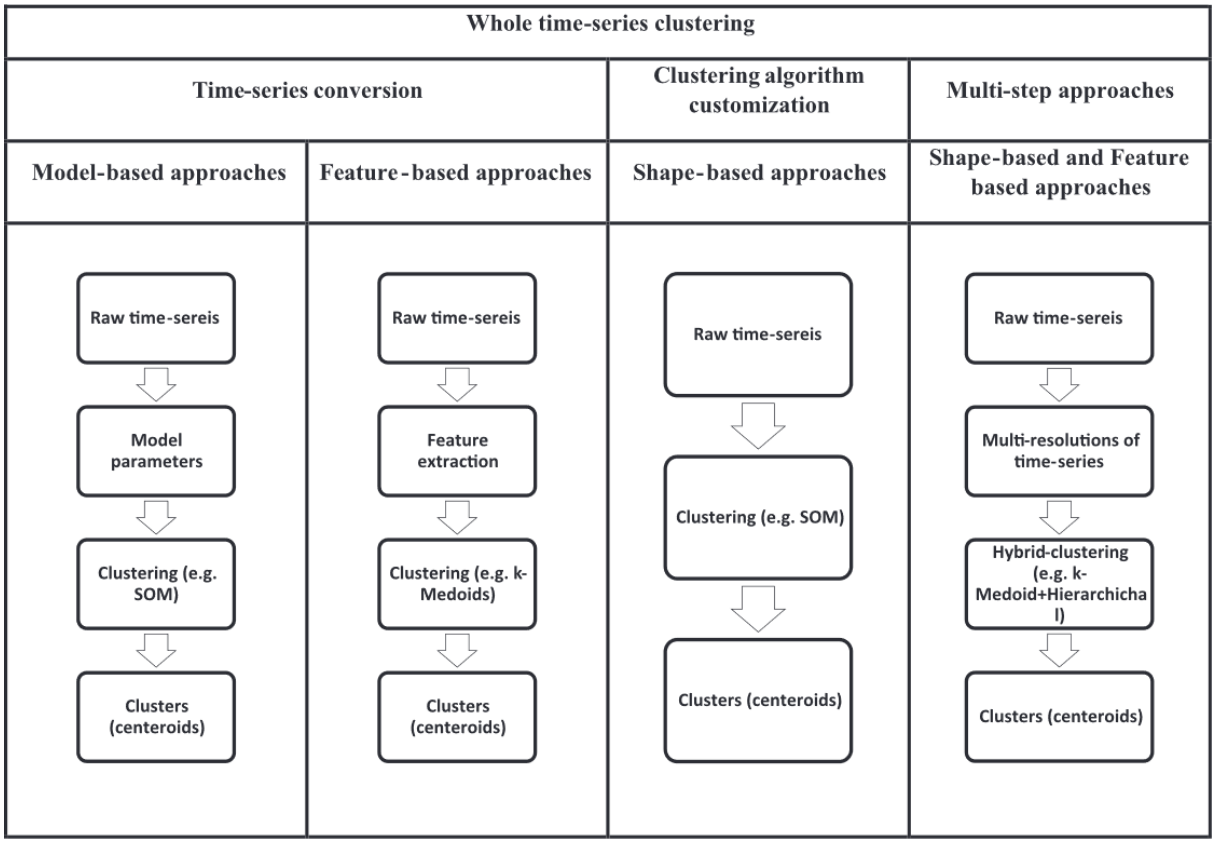


Figure 2.1: Different approaches for clustering time series. Figure is taken from Aghabozorgi et al. (2015)

2.2.3 Challenges with time series clustering

Clustering time series is unintuitive and is significantly harder than clustering static data. Some of the challenges with clustering of time series are summarised by Aghabozorgi et al. (2015). First of all, the time series data set requires a lot of memory and often requires storage on disk which leads to decreased efficiency. Secondly, is that the time series data set are often high dimensional (Lin et al., 2003b; Keogh and Pazzani, 2000) which makes handling the data difficult (Wang et al., 2004) and again slows down the process of clustering (Zhang et al., 2006). Finally, the similarity or dissimilarity measure is fundamental to all clustering algorithms and the choice of the correct similarity measure is complicated for time series clustering. The reason for the complication is that time series data are naturally noisy, include outliers and shifts in time (Lin et al., 2004). Also, the length of the time series is not necessarily equal in length, complicating the choice even more (Aghabozorgi et al., 2015).

2.2.4 Applications for time series clustering

The applications of time series clustering are mostly used to discover interesting patterns in time series data sets. Aghabozorgi et al. (2015) states, after reviewing the literature, that the applications can typically be divided into two categories: The first category referrers to the discovery of patterns which frequently appears in the data set; the second category are methods to discover patterns which happen surprisingly. Clustering time series have become increasingly popular and attracted attention from a wide range of researchers from many different disciplines. For instance, researchers have clustered time series from speaker verification, robot sensor data, financial data, earthquake, gene expression, retail pattern, commercial consumption, brain activity and much more (Zhang et al., 2011). Aghabozorgi et al. (2015) has done e complete review on whole time series clustering methods used before the year of 2015. The different domains which Aghabozorgi et al. (2015) found for clustering of time series are, but not limited to aviation/astronomy, biology, climate, energy, environment and urban, finance, medicine, psychology, robotics, speech/voice recognition, and user analysis.

Firstly, within the biology section, clustering has been adapted to the clustering of gene expression data (Subhani et al., 2010; Fujita et al., 2012; Pyatnitskiy et al., 2014). For example, Fujita et al. (2012) performed gene clustering through the identification of Granger causality between and within sets of time series gene expression data. Within the energy sector, Iglesias and Kastner (2013) used time series clustering to discover energy consumption patterns within buildings. It also compared the most popular similarity distances where Euclidean distance was the measure that obtained the best, balanced general solutions. Within medicine, Gullo et al. (2012) presented a data mining approach for analysis of mass spectrometry data. New representation method called DSA was proposed and applied to K-means clustering with DTW distance measure. The objective was to explore, identify and group pathological cases from MS clinical samples. Further publications from before 2015 related to clustering of time series from the domains listed above and can be reviewed in Aghabozorgi et al. (2015).

Major findings for applications for time series clustering after 2015 are reviewed in this paragraph. Nguyen et al. (2018) clusters time series which was obtained from whole-volume calcium imaging experiments. A model-based functional data analysis methodology via Gaussian mixtures for clustering of data from such visualisations is proposed in this paper. Torabi et al. (2016) performs clustering on high-speed end milling experimental data. The authors show that the clustering methods can coarsely capture the state of the process and can be applied to fault diagnosis and tool condition monitoring (TCM) purposes. Diez et al. (2016) presents a clustering-based approach to group substructures or joints with similar behaviour on a bridge and then detect abnormal or damaged ones (i.e. structural health monitoring). Vibration signals caused by passing vehicles from different joints are classified and damaged joints can then be detected and located. Here K-means based clustering was implemented. These are but a few of the applications which time series clustering has been applied to. To the best of the author's knowledge, time series clustering has not been applied to clustering of time series acquired from wind turbines other than the paper mentioned in the introduction (i.e. related work).

2.3 Components of time series clustering

In this section, the major components of time series clustering will be addressed. Aghabozorgi et al. (2015) have comprehensively reviewed the literature and based on that stated that time series clustering can essentially be divided into the following major components:

1. Representation method
2. Similarity/dissimilarity measure
3. Prototype definition or characteristic of the cluster
4. Clustering algorithm
5. Cluster validation and evaluation metric.

According to Aghabozorgi et al. (2015), the general process for time series clustering is to apply some or all of these steps, depending on the characteristic of the time series and the application. The general process is outlined as follows: First, a representation method or dimensionality reduction is applied to the time series data such that it takes less space in memory. Then, depending on the application, a specific clustering algorithm is chosen along with an appropriate similarity measure and a prototype for characterisation of the cluster centre. Afterwards, the obtained clusters are evaluated with regards to specific evaluation metrics. Lastly, the results are interpreted to acquire meaning out of the obtained clusters. These different components will be addressed in more detail in the following sections.

2.3.1 Representation method for time series

Dimensionality reduction relies on the fact that there is a lot of redundancy in the data and that a small fraction can represent most of the information (Algorithmia Inc., 2018). Unsupervised learning is in many cases used to transform the data into different representation (Zolhavarieh et al., 2014). The definition of time series representation, as stated by Aghabozorgi et al. (2015), can be seen in Definition 2.3.1

Definition 2.3.1. Time series representation , given a time series data $F_i = \{f_1, \ldots, f_t, \ldots, f_T\}$, representation is transforming the time series to another dimensionality reduced vector $F_i^{'} = \{f_1^{'}, \ldots, f_x^{'}\}$ where $x < T$ and if two time series are similar in the original space, then their representations should be similar in the transformed space too.

Motivations for transforming the time series to a lower dimensional space or different representation method are, but not limited to:

- Reducing the memory requirements: Storing raw time series requires a lot of memory, especially when considering 100+ sensors sampled at a high sampling rate.
- Increasing the efficiency of the algorithm: The calculation of similarity measure between raw time series is computationally expensive. The clustering - or calculation of the dissimilarities - can be significantly sped up by reducing the dimensions of the clustering problem.
- Sensitivity to noise: If there is significant noise in the time series, the similarity of two raw time series might be similarity in noise, rather than similarity in shape (Aghabozorgi et al., 2015).
- Tool for visualisation: There is no intuitive way to represent high dimensional data. Reducing the dimensions of the time series is one way to plot the data in two or three axis plot (Sarah Guido, 2013).

Different representation methods

According to Lin et al. (2003a); Ratanamahatana et al. (2005); Bagnall et al. (2006); Shieh and Keogh (2009), representation methods can be divided into the following categories: Data adaptive, non-data adaptive, model-based, and data dictated.

- **Data adaptive**: This representation is performed on all time series in the data set when the objective is to minimise the global reconstruction error. The name, data adaptive, means that the parameters used in the transformation are dependent on the data available. However, these methods tend to make the comparison between the time series difficult (Aghabozorgi et al., 2015; Esling and Agon, 2012).

- **Non-data adaptive**: These representation methods are suitable for time series of fixed length segmentation. The comparison between time series, compared to data adaptive representations, are straightforward (Aghabozorgi et al., 2015).
- **Model-based**: Representation of time series through stochastic models such as a Hidden Markov Model (HMM) or an ARMA model (Aghabozorgi et al., 2015).
- **Data dictated**: For the categories above, the compression ratio is decided by the user. However, for data dictated representation methods, the characteristic of the raw time series automatically determines the compression ratio (Aghabozorgi et al., 2015). Example of this is the data dictated representation method called Clipped.

Non-data adaptive representation methods are most suited for time series of equal length and would, therefore, be most applicable. Some frequently mentioned representation methods within this category are Discrete Fourier Transform (DFT), Perceptually important point (PIP), and Wavelets methods: Discrete Wavelet Transform (DWT), Piecewise Aggregate Approximation (PAA), Chebyshev Polynomials (CHEB), Indexable Piecewise Linear Approximation (IPLA). Regardless of the representation method used, one of the important challenges is to find a similarity measure which is appropriate and compatible with the representation method (Aghabozorgi et al., 2015).

2.3.2 Similarity/Dissimilarity Measure

Time series clustering is highly sensitive to the choice of distance measure (or similarity measure) and is fundamental to all clustering techniques (Aghabozorgi et al., 2015; Trevor et al., 2009). A measure of similarity is required in order to compare time series and partition them into different groups based on their dissimilarity. Depending upon whether the time series are of equal or unequal length, a particular similarity measure might be more appropriate than another. The mathematical definition of distance function between two time series will be introduced in Section 3.1. Similarity measures are divided into several categories with respect to some common features. The first category divides the different similarity measures with respect to their objectives (i.e. the objective of similarity measure). The second category divides the different similarity measures into four groups with respect to the functioning of the similarity measures. First of all, the objective of the clustering process with respect to the similarity measure will be explained.

Objective of the time series clustering

The choice of an appropriate similarity measure for an application depends on the length of the time series (equal or unequal length, and short or long), representation method used, the characteristic of the time series data, and the objective of the time series clustering (Aghabozorgi et al., 2015). The **objective** of the time series clustering can be put into three categories, with its appropriated similarity measure. The three objectives are similarity in time, shape and change.

- **Similarity in time** computes the similarity between two time series at the same time stamps. These distance methods are also known as *lock-step* methods and refer to the one-to-one mapping between time series Roelofsen (2018). Common distance measures suitable for this objective are the Euclidean distance and correlation based distances (Aghabozorgi et al., 2015; Zhang et al., 2011). The computation of distances is costly on raw time series and that is why the computations are normally performed on transformed time series such as Wavelets, Fourier transform or Piecewise Aggregate Approximation (PAA) (Aghabozorgi et al., 2015).
- **Similarity in shape** is a more general case of similarity in time, where the time of occurrence of patterns is not important (Aghabozorgi et al., 2015; Zhang et al., 2011). Because of this, these are sometimes referred to as elastic methods (Esling and Agon, 2012). The difference between **elastic** and **lock-step** methods are that lock-step methods measure the distance from one-to-one (same time in both time series) and elastic methods are more flexible as they can either be measured as one-to-many or one-to-none (Esling and Agon, 2012). A commonly used elastic distance measure is Dynamic Time Warping (DTW). Lock-step methods are generally outperformed by elastic methods when time series are subjected to noise, scales and time shifts.
- **Similarity in change** clusters objects by how they change from time step to time step (Zhang et al., 2011). A model, such as HMM or ARMA, is fitted to the time series and the similarity is measured on the fitted parameters. That is, time series of similar autocorrelation structure are clustered together (Aghabozorgi et al., 2015).

Categories of similarity measures

In addition to the objective of time series clustering, similarity measures are normally categorised into four categories: shape-based, compression-based, feature-based and model-based similarity measures(Aghabozorgi et al., 2015).

- **Shape-based** distance measures aim to find similarity in time and shape. Thus, both the objective of similarity in time and shape are included in this category. Most popular shape-based measures are Euclidean distance and the other L-norms, DTW, LCSS and MVM. As stated by Aghabozorgi et al. (2015), these measures are appropriate for short time series. Shape-based dissimilarities can fail to work for longer time series which contains a significant amount of noise or outliers (Montero et al., 2014). For long and noisy time series either a feature-based or model-based approach might be more appropriate.
- **Feature-based** use any kind of distance function for comparison of features which are describing the different time series (Esling and Agon, 2012). Extraction of the features really depends on the applications. Features which could be of interest, to mention a few, are frequency content, mean, variance, minimum and maximum value, and so on. According to Wang et al. (2006b), feature-based methods are appropriate when time series are subjected to shifts in time or if there is a lot of missing values. The feature-based approach is appropriate for long and noisy time series (Aghabozorgi et al., 2015).
- **Compression-based** similarity measures are based on how well two time series can be compressed together (Esling and Agon, 2012). These are suitable for both long and short time series (Aghabozorgi et al., 2015). Examples of such methods are CDM, Autocorrelation, Pearson's correlation coefficient and related distances, Cepstrum, Piecewise normalisation and Cosine wavelets.
- **Model-based** similarity measures complies with the objective of similarity in change where parameters of the model, such as HMM or ARMA, are fitted to the time series and compared. According to Wang et al. (2006b), clustering the parameters of an ARMA process is not a reliable method; the set of parameters could be different for time series with a similar structure. This would highly affect the clustering results as small changes in these parameters could result in a different cluster assignment. These similarity measures are mostly appropriate for long time series, not modest or short (Aghabozorgi et al., 2015).

Some of the similarity measures are based on a specific representation, such as SAX which is compatible with MINDIS. Others work regardless of representations method and are even compatible with raw time series (Aghabozorgi et al., 2015). Shape-based methods can be used for clustering the time series with the objective of similarity in time and shape. The most common similarity measures used on time series, as stated by Aghabozorgi et al. (2015), are the Euclidean distance, Euclidean distance in a PCA subspace, Dynamic Time Warping (DTW), Longest Common Sub-Sequence (LCSS), Hausdorff distance, modified Hausdorff (MODH), and HH-based distance. Among these, Euclidean distance and DTW are by far the most commonly used methods for time series clustering (Aghabozorgi et al., 2015). Furthermore, Aghabozorgi et al. (2015) cites Lkhagva et al. (2006) which states that Euclidean distance is actually surprisingly competitive, despite its simplicity. These shape-based methods are widely used despite their limitation. Some of the limitations of shape-based methods, as stated by Wang et al. (2006b), are that they cannot deal with missing values or time series of different length (except for DTW) and are very sensitive to noise. These are some of the remarks which need to be dealt with prior to clustering if shape-based methods are applied. On the other hand, feature-based methods can effectively handle time series of longer interval and are less sensitive to noisy or missing data as the feature extraction reduces the dimensionality of the time series. In short, shape-based methods can be used when the time series are not subjected to noise or outliers. If the time series are relatively long and noisy, either model-based or feature-based methods should be applied instead.

2.3.3 Prototype definition

Cluster prototype, or cluster representative, is used by most clustering techniques and represents what characterises each cluster obtained by the clustering algorithm (Duval, 2016). That is, a cluster prototype is a data object which is representative to the other objects within the same cluster (Tan et al., 2005). Aghabozorgi et al. (2015) states that one of the problems that lead to low accuracy of clusters is a poor definition of the prototype and a poor updating sequence for cluster prototypes. He further explains that the accuracy of some clustering methods is highly dependent on the choice of an appropriate clustering prototype. Example of this are partitioning algorithms such as K-means, K-medoids, Fuzzy C-means and Ascendant Hierarchical Clustering. Given a time series clustering problem, the cluster prototypes are the prototypes which minimise the distance between all time series and its

corresponding assigned prototype. This objective corresponds to minimising the well known Steiner sequence (Aghabozorgi et al., 2015; Petitjean and Gançarski, 2011). There are several methods for solving this optimisation problem and Aghabozorgi et al. (2015) points out that there are generally three approaches in literature which are commonly used:

- **The medoid sequence of the set**: Cluster medoid as the prototype is the most common way to approach optimal Steiner sequence in time series clustering (Kaufman, 1990). The medoids are the representative objects of the data set which minimises the distance between all time series in each cluster (Aghabozorgi et al., 2015). In other words, the time series within the clusters which has the lowest sum of the squared error to the rest are chosen to be the centre of that cluster.
- **The average sequence of the set**: This approach is only appropriate for time series of equal length and non-elastic (lock-step) distance measures (Aghabozorgi et al., 2015). The cluster centres are then simply the average of all time series within that cluster. If the time series are of unequal length or an elastic distance measure is used, the actual average shape cannot be captured (Niennattrakul and Ratanamahatana, 2007).
- **The local search prototype**: Prototype is obtained by a local search instead of medoid. However, the improvements over the other prototypes, such as medoid averaging methods, are not clear (Aghabozorgi et al., 2015).

The most commonly used prototype is the medoid, where the averaging method is scarcely used. The reason there are fewer papers on averaging prototype methods is that it requires equal length time series and non-elastic distance measures (e.g. Euclidean distance) (Aghabozorgi et al., 2015). However, Aghabozorgi et al. (2015) concludes that the prototype which yielded the best accuracy was the local search prototype. However, there are few papers on this and its advantages over averaging methods are not clear. Common clustering algorithms which use medoid and average sequence of the set are K-medoids and K-means, respectively.

2.3.4 Clustering algorithm

Clustering algorithms can be divided into two main categories or partitioning methods which describes the nature of how the clusters are constructed:

- Hard clustering (or *crispy clustering*) - Each object in the data set belongs to only one cluster.
- Soft clustering (or *fuzzy clustering*) - Clusters may overlap and for each object there exist a probability or likelihood of that point belongingness to each of the different clusters (Wolfram Research, 2004; Kaushik, 2016; Patibandla and Veeranjaneyulu, 2018).

There are various approaches for applying these partitions based on different models (Patibandla and Veeranjaneyulu, 2018).

Category of clustering algorithms

Clustering is subjective and because of this, there exists a huge amount of algorithms which is used. According to Wong (2015); Liu and Lu (2015), most data clustering problems have been shown to be NP-hard, which explains the number of clustering algorithms available. Kaushik (2016) states there are more than 100 different clustering algorithms where each of them can be placed into a specific category (or several categories). Most of the algorithms can be placed within the following paradigms: Partitional clustering, hierarchical clustering, density-based clustering, model-based clustering, grid-based clustering, correlation clustering, spectral clustering, gravitational clustering, herd clustering, subspace clustering, clustering based on neural networks, and so on (Wong, 2015). Common to surveys of clustering algorithms (Rai and Shubha (2010); Nagpal et al. (2013a); Xu and Tian (2015b); Ahmad and Khan (2018a); Aghabozorgi et al. (2015); Warren Liao (2005)) are the categories:

- Partitioning methods
- Hierarchical clustering
- Density-based clustering
- Model-based clustering

Density-based and model-based clustering have not been broadly used on larger time series data set because of their high time complexity (quadratic or higher) and slow process (Aghabozorgi et al., 2015). Model-based clustering also has the disadvantage of being dependent on user-specified parameters, which are in most cases just assumptions. Partitioning algorithms, on the other hand, have low time complexity and is therefore very efficient. These require the user to specify the number of clusters, which is not feasible or available for many applications (Liu and Lu, 2015; Rodriguez et al., 2019). Hierarchical clustering, on the other hand, does not assume the user to specify the number of clusters, which is a great advantage for such applications (Xu and Tian, 2015a). However, hierarchical clustering algorithms have a rather high time complexity and are therefore mostly applicable for small data sets (Aghabozorgi et al., 2015; MacQueen, 1967; Bradley et al., 1998). Because of these reasons, much of the literature apply either partitioning methods or hierarchical clustering when clustering time series data. These will be addressed in more detail in the consecutive section.

Partitioning methods

Partitioning methods are clustering techniques which partition the data set into k clusters where the number of cluster k is one of its hyper-parameters which is specified by the user. This is in most cases obtained by minimising an objective function, similar to that of the Steiner sequence (Nagpal et al., 2013b). The most commonly used partitioning algorithm is the **K-means**, due to its ease of implementation, simplicity and efficiency. K-means finds a partition such that the distance from each object to its assigned cluster centre (prototype) is minimised (Liu and Lu, 2015; Aghabozorgi et al., 2015). Other members of the partitioning methods are K-medoids (or PAM), CLARA and CLARANS. The PAM algorithm is similar to K-means, only the cluster prototype is the medoid sequence of the set, compared to the average sequence of the set for the K-means algorithm (Aghabozorgi et al., 2015). CLARA and CLARANS are improvements to the K-medoids algorithms for use on spatial databases. Common for these methods is that the clusters are not overlapping (i.e. crispy clustering). On the contrary, Fuzzy c-Means (FCM) algorithm is a well-known partitioning method that constructs fuzzy clusters. Here each object is not only assigned to just one cluster but will have some degree of membership to all the cluster centres (Hautamaki et al., 2008). Most of the literature reviewed in the decade review, Aghabozorgi et al. (2015), use K-means or K-medoids as their clustering algorithm, however, there are also some that use FCM.

Hierarchical clustering

Common for these methods is that the algorithms construct a hierarchy of clusters. The hierarchy can be organised in two ways and is therefore divided into two subgroups: Agglomerative and divisive (Rodriguez et al., 2019). The former is more commonly used in clustering of static data (Rai and Shubha, 2010). In an *agglomerative hierarchical clustering algorithms* the hierarchy is ordered bottom up. This essentially means that all objects first belong to its own individual cluster only containing itself. Iteratively, these sets of clusters are merged together based on a specific linkage criterion. On the other hand, *divisive* hierarchical clustering algorithms start with all objects in one single cluster and is separated into smaller clusters (top-down) (Rodriguez et al., 2019; Aghabozorgi et al., 2015). Typical hierarchical clustering algorithms are BIRCH ,CURE, ROCK and Chameleon.

The advantage of hierarchical clustering algorithm for time series or static data is that it does not require the user to input any parameters such as with many of the other clustering techniques (Aghabozorgi et al., 2015; Wong, 2015). All the different hierarchical clustering methods are clustered in a tree-based representation called *dendrograms*; the number of cuts is determined by viewing the dendrogram (and in most cases the internal evaluation indexes). This process of constructing these dendrograms is deterministic, which means that given a dissimilarity matrix, it would always result in the same assignment (Sardá-Espinosa, 2017). The dendrogram is also a great tool for visualising the time series and their similarity (or dissimilarities). Much information can be obtained by simply viewing the dendrogram. The ability to visualise has made hierarchical clustering approach quite popular in time series clustering. However, it is not capable to effectively deal with large data sets (Wang et al., 2006a) because of its quadratic computational complexity (poor scalability).

2.3.5 Time series clustering evaluation measure

The purpose of evaluation indicators is to validate the results from the algorithm. These evaluation indicators are divided into two main categories: the internal evaluation indicators and the external evaluation indicators.

Internal evaluation indicators

The internal evaluation indicators use the internal data to test the validity of the clustering results. They are usually used if the ground truth is not available and the use of external evaluation indicators are not possible. Internal evaluation indicators are mainly used for choosing optimal clustering algorithms to use on a specific data set, rather than validation of the clusters. According to Aghabozorgi et al. (2015), this is because these indicators can only make comparisons between clustering approaches used on the same data set and should not be compared to different data sets. There are tens of internal indexes: R-squared index, Root-mean-square standard deviation (RMSSTD) index, Davies-Bouldin indicator, Dunn indicator, Homogeneity index, Separation index and Silhouette coefficient, to mention a few. For evaluating the clusters in terms of accuracy, the sum of squared error can be used, which is implemented in different works such as in (Lin et al., 2004; Vlachos et al., 2003). Here the error for each time series is the distance from that time series to its the nearest cluster (Aghabozorgi et al., 2015).

External evaluation indicators

External evaluation indicators use external data to test the validity of algorithms (Xu and Tian, 2015a; Rodriguez et al., 2019; Aghabozorgi et al., 2015). This requires us to know the correct partitioning of the data set. Common to all external indicators is that it requires the knowledge of the ground truth and compares the resulting clusters with that of the ground truth (Manning et al., 2010). (Aghabozorgi et al., 2015) has done a literature review of the external indicators. They state that external indicators used by previous time series clustering literature are: Purity, CSM, Rand Index, Adjusted Rand Index (Hubert and Arabie, 1985), Entropy, Jaxard, F-measure, FM (Fowlkes and Mallows, 1983) and MNI (Hubert and Arabie, 1985). Common for all these measures is that they range from 0 to 1, where 1 indicates a 100% correspondence to the ground truth.

There are many validation tools to choose from, where it is unsure whether these will yield a good metric of comparison between different models or performance of the cluster analysis. James et al. (2013) states that any time clustering is performed on a data set, regardless of the shape of the data set, clusters will be found. But what we really want to get an answer to is if these clusters represent true subgroups of the data set, or whether these are just noise. Aghabozorgi et al. (2015) further speculates that there has been no consensus on a single best approach for evaluation of the cluster assignment. Because the ground truth is not known, the evaluation measure is limited to internal evaluation indicators and visual confirmation

2.4 Summary

This section will briefly summarise the important concepts introduced in Chapter 2 and make a choice of which clustering technique and methods which will be used. First of all, we introduced the concept of different types of time series clustering in Section 2.2.1, namely whole time series clustering, STS clustering and time-point clustering. As the objective of this report is to cluster independent time series from different wind turbines, the type of clustering is restricted to whole time series clustering. In Section 2.2.2, different ways to cluster time series was introduced. The different approaches are either shape-based (or raw data-based), feature-based or model-based approach. The model-based approach has been shown to have scalability problems and reduces performance when clusters are close to each other (Vlachos et al., 2004; Mitsa, 2010). Furthermore, Wang et al. (2006b) states that the ARMA process is not a reliable method as the set of parameters could be different for time series with a similar structure. Therefore, to narrow in the scope of this thesis, the model-based approach will be disregarded and not analysed. Fitting the time series to an ARMA process would solve the clustering problem with respect to the objective of similarity in change; therefore, the objective of similarity in change is automatically excluded. The remaining approaches are then shape-based and feature-based approach. Shape-based typically work directly with raw time series whereas the feature-based approach converts the raw time series into a feature vector which is then clustered. The shape-based approach is appropriate for time series of short to modest length and feature-based are more appropriate for longer time series. The limitations of the shape-based approach for longer time series are significant if the time series are subjected to noise and/or missing values; the shape-based approach cannot handle missing values and the clustering results are quite sensitive to noise. This is also the case for shorter time series. If a shape-based approach is used, the time series requires effective preprocessing steps which deal with the missing values, noise and outliers prior to clustering. The most common approach, according to Aghabozorgi et al. (2015), is clustering time series through the shape-based approach. However, the feature-based approaches are not uncommon and are more appropriate for time series of longer length (not necessary equal length time series). The limitations of the shape-based approach become less significant if the time series are properly preprocessed. The thesis will, therefore, be limited to clustering time series with respect to a shape-based approach. In Section 2.3, the major components of time series clustering was introduced. These are the representation method, similarity measure, prototype, clustering algorithm, and cluster evaluation metric. A brief summary of the components followed by a choice of which methods to apply will be included in the following paragraphs. The most critical choice for time series clustering - for the components above - is the choice of the similarity measure.

Representation method: Representation methods are divided into four sub-categories: data adaptive, non-data adaptive, model-based and data dictated. As the shape-based (also called raw-data-based) approach for the clustering will be adopted, no specific representation method is required. The thesis will not concern itself with limitation such as memory or storage problems. The time series extracted will not be large enough to cause these problems. But in the case this approach will be scaled up, a choice of a non-adaptive representation method is recommended. Non-data adaptive representation methods can be applied because they are most suited for time series of equal length and the comparison between time series are straightforward.

Similarity measure: Many regard the choice of similarity measure as the most important consideration when doing time series clustering. The similarity measure is first divided into similarity measures fit to the objective of the clustering. The two objectives of interest are similarity in time and similarity in shape. The scope of the thesis will be limited to these two objectives, as the shape-based approach will be used for clustering the time series. Shape-based methods contain similarity measures which are specific for both objectives. The most commonly used similarity measures are the Euclidean distance and Dynamic Time Warping (DTW) distance; both are shape-based methods but with different clustering objectives: Euclidean distance is a typical lock-step method which clusters time series which are similar in time; DTW is an elastic method which clusters time series which are similar in shape. Both similarity measures are applicable on raw-time series and most non-data adaptive representation methods. Still, care must be taken when finding a match (if a representation method is used). The specific preprocessing steps required for the time series, for both objectives, will be discussed and described in detail in Chapter 4.

Cluster prototype: The cluster prototype will be either the medoid sequence of the set or the average sequence of the set. The average sequence of the set is scarcely used as compared to the medoid sequence of the set because the former is limited to equal length time series and non-elastic distance measure. Thus, the average sequence

of the set will be used as the prototype if Euclidean distance is used, and the medoid sequence of the set as the prototype if DTW is used as the similarity measure.

Clustering algorithm: Furthermore, the clustering algorithm will be limited to that of partitioning methods and hierarchical clustering, which are the most popular time series clustering algorithms. Partitioning algorithms are advantageous over hierarchical algorithms in terms of their low time-complexity and simplicity. This has made partitioning algorithms highly suitable for time series clustering. The most common partitioning algorithm is the K-means, due to its ease of implementation, simplicity and efficiency. However, this method requires the user to specify the number of clusters and is limited to time series of equal length only. Hierarchical clustering algorithms do not require the user to specify the number of clusters prior to clustering and can cluster time series of unequal length if an appropriate elastic distance measure, such as DTW, is used. However, hierarchical clustering has relatively high time complexity compared to partitioning methods. Nonetheless, these methods have great power of visualisation because of the construction of dendrograms which are visually pleasing and informative. Both K-means and hierarchical clustering will be implemented and a comparison between the algorithms will be made.

Evaluation indicators: Lastly, the evaluation measures are divided into external and internal evaluation indicators. Because the ground truth or the actual cluster assignment is not known, the evaluation is limited to internal evaluation indicators and visual confirmation. The internal indicators which will be used to determine the quality of the clusters and the number of clusters (or cuts to the dendrogram) are the Silhouette index and the sum of squared error within each cluster (SSE). In the next chapter, the theory behind the specific similarity measure, clustering algorithm and evaluation indicators will be addressed.

Chapter 3

Theory - Time series clustering

In this chapter, the theoretical background behind the different components of time series clustering will be addressed. Because the time series will be clustered according to the shape-based approach, some modifications to the conventional clustering algorithms must be made such that it is capable of dealing with time series. Most of the modifications to the conventional clustering algorithm concerns the modification to the similarity measure. In Section 3.1, the similarity measure (or distance function) will be defined along with the dissimilarity matrix where all similarity measures are stored prior to feeding it to the clustering algorithm. The Euclidean distance and the DTW distance will be introduced followed by a brief comparison between the methods. The theory behind each of the clustering algorithms implemented - K-means and Hierarchical clustering - will be introduced in Section 3.2 followed by the definition of the internal indicators in Section 3.3. These are the components of time series clustering and the Python implementations can be seen in Appendix D.

3.1 Similarity measure

A measure of similarity is required in order to compare time series and partition them into different groups based on their (dis)similarity. Aghabozorgi et al. (2015) defines the distance between two univariate time series **x** and **y**, both of length *n*, as the sum of the pairwise distance between the two vectors

$$D_u(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^{n} d_t(x_t, y_t), \tag{3.1}$$

where $d_t(x_t, y_t)$ is the distance between two time series at time t. For a multivariate case, where d refers to the number of dimensions, the distance between two multivariate time series **x** and **y** is given by:

$$D_m(\mathbf{x}, \mathbf{y}) = \frac{1}{sum(\boldsymbol{\alpha})} \sum_{i=1}^{d} \alpha_i \sum_{t=1}^{n} d_t(x_{i,t}, y_{i,t}) = \frac{1}{sum(\boldsymbol{\alpha})} \sum_{i=1}^{d} \alpha_i D_u(\mathbf{x}, \mathbf{y}), \tag{3.2}$$

where $\boldsymbol{\alpha} = \{\alpha_1, \ldots, \alpha_d\}$ corresponds to the weighting vector. The weighting vector in the function allows the user to weight the importance of different univariate data set in a multivariate case according to the application.

3.1.1 Dissimilarity Matrix

Regardless of the measure of dissimilarity is used, most algorithms presume a dissimilarity matrix (or distance matrix) as their input. The dissimilarity matrix (DM) corresponds to a $N \times N$ matrix, where *N* is the number of objects (number of individual time series) in the data set. Each element $D(i, j)$ in the matrix refers to the (dis)similarity between the *i*th and *j*th object (or time series) in the data set (Trevor et al., 2009). Most clustering algorithms presumes that the dissimilarity matrix is symmetric with non-negative entries and zero diagonal elements. The

dissimilarity matrix can be defined as:

$$DM = \begin{bmatrix} 0 & & & & \\ D(2,1) & 0 & & & \\ D(3,1) & D(3,2) & 0 & & \\ \vdots & \vdots & & \ddots & \\ D(N,1) & D(N,2) & \dots & D(N,N-1) & 0 \end{bmatrix} \tag{3.3}$$

Calculation of the dissimilarity matrix for a data set comprising of N time series requires $\frac{N(N-1)}{2}$ different computations of distances, $D(i,j)$. This is without including the time-complexity of calculating the individual distances between time series. Any algorithm which requires a dissimilarity matrix cannot have lower time complexity than $O(N^2)$. The time-complexity of calculating each distance $D(i,j)$ depends on the distance metric used. A final remark, when referring to a condensed distance matrix, we refer to a list of size $\frac{N(N-1)}{2}$ which is just the elements below the diagonal (from top left to bottom right).

3.1.2 Lock-step measures

Shape-based similarity measures are divided into two categories: lock-step and elastic methods. In this section, lock-step methods will be defined. Lock-step methods measure the distance between two time series in a one-to-one fashion and therefore requires the time series compared to be of equal length. They are used when the objective of clustering the time series is similarity in time. Another major limitation to lock-step methods is that they cannot handle time shifts or lags (Roelofsen, 2018). If the time series are out of phase, lock-step methods would result in a poor comparison. Common lock-step measures are Minkowski distances (or L_p-norms) and correlations based distances. Minkowski is the generalisation of the well known Euclidean distance ($p = 2$) and Manhattan distance ($p = 1$). Euclidean distance was chosen as the similarity measure for the objective of similarity in time. The Euclidean distance is one of the most commonly used similarity measures within time series clustering and will be defined in the following section.

Euclidean distance

Euclidean distance (ED) is one of the most common lock-step method used as similarity measures in time series clustering (Aghabozorgi et al., 2015; Mori et al., 2016). The ED has the following definition:

$$D_{euc}(\mathbf{x},\mathbf{y}) = \sum_{t=1}^{n} d_t(x_t, y_t) = \sqrt{\sum_{t=1}^{n}(x_t - y_t)^2}. \tag{3.4}$$

ED is very efficient and has a time complexity of $O(n)$, which results in an overall time complexity of $O(nN^2)$ for calculating the entire distance (or dissimilarity) matrix (3.3). It also satisfies the triangle inequality (i.e. the distance is metric) (Cassisi et al., 2012). However, ED can only deal with time series of equal length and is highly susceptible to outliers, missing values and/or noise. This common limitation for lock-step method limits the objective of the clustering to objective of similarity in time only (Mori et al., 2016; Cassisi et al., 2012). The one-to-one mapping of the ED between two time series can be further visualised in Figure 3.1.

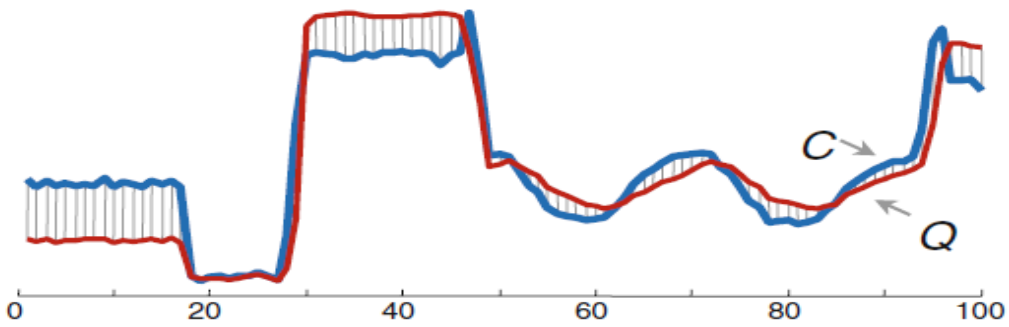


Figure 3.1: Euclidean distance metric for two time series Q and C. Picture taken from Elsworth (2017). The Euclidean distance is the sum of all the vertical grey lines in the plot.

3.1.3 Elastic measures

In contrast to lock-step methods, elastic methods are more flexible and can either be measured as one-to-many or one-to-none point matching. These can, therefore, be used when time shifting or unequal length time series is the problem. Time of occurrence is not important for these measures and therefore the objective falls into the category of similarity in shape. However, the drawback of these measures is a drastic increase in time complexity; most of these are of quadratic time complexity, such as the Longest Common Subsequence (LCSS) and Dynamic Time Warping (DTW). Both of these methods do not satisfy the triangle inequality (i.e. the distance is non-metric) (Cassisi et al., 2012; Elsworth, 2017). Regardless, the most commonly used elastic method is DTW and will be addressed in details in the following section.

Dynamic Time Warping

DTW was first proposed by Berndt and Clifford (1994) and overcomes some of the disadvantages of the Euclidean distance at the cost of increased time and space complexity; DTW can cope with both time distortion and varying sampling times of the time series compared (Roelofsen, 2018). DTW starts by constructing a $n \times m$ matrix, called local cost matrix (LCM), where the (i, j) element in the matrix refers to the distance between the time point i and j in the time series **x** and **y**, respectively. The LCM is presented in the following equation:

$$LCM = \begin{bmatrix} d(x_1, y_1) & d(x_1, y_2) & \dots & d(x_1, y_m) \\ d(x_2, y_1) & d(x_2, y_2) & & \\ \vdots & & \ddots & \\ d(x_n, y_1) & \dots & & d(x_n, y_m) \end{bmatrix} \tag{3.5}$$

For calculating the distance $d(x_i, y_j)$ between the elements in each time series, the squared error or the root squared error is typically implemented. Next, a warping path $W = \{w_1, w_2, \dots, w_k\}$ of length k is computed which satisfies the following conditions:

1. **Boundary condition**: A warping path has to start at the bottom left corner and end at the top right corner. That is, the warping path w needs to start at DM(1,1) and end at DM(n,m).
2. **Continuity condition**: Only adjacent elements are considered valid steps for the warping path. This also includes diagonal elements.
3. **Monotonicity condition**: The path must be monotonic, meaning that it cannot make a step in a non-increasing direction. That is, the only direction which is valid is right, diagonally to the right and up.

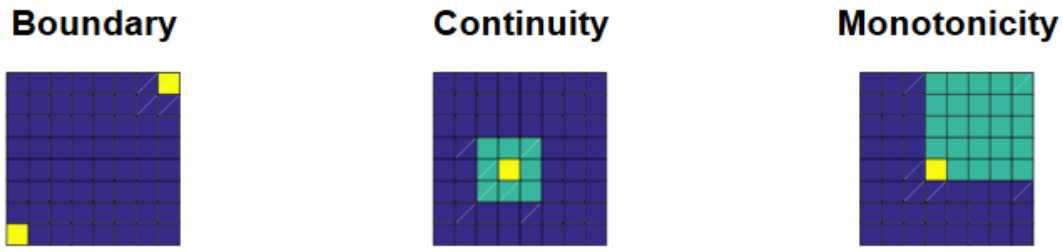


Figure 3.2: Illustration of the conditions for the warping path of DTW. Picture taken from Elsworth (2017).

These three conditions are visualised in Figure 3.2. The path which satisfies the conditions above and has a minimum cumulative total distance is defined as the DTW distance. A path which satisfied these conditions can be obtained by a $O(nm)$ algorithm which is based on dynamic programming. The DP recurrence that finds the minimum cumulative distance is given by:

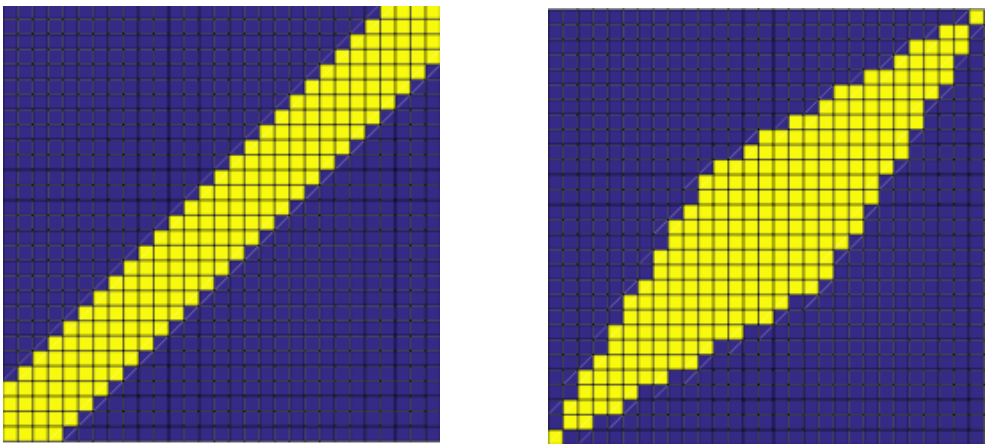
$$d_{cum}(i, j) = d(x_i, y_j) + \min\{d_{cum}(i-1, j-1),\ d_{cum}(i-1, j),\ d_{cum}(i, j-1)\}. \tag{3.6}$$

The minimum weighted cumulative total distance between the two boundary points represents the DTW distance. The path W which minimises the total sum of weights along the traversed path is the only valid path. This

corresponds to the following definition (Berndt and Clifford, 1994) of the DTW distance:

$$D_{DTW}(x, y) = \min \sum_{t=1}^{n} w_k \tag{3.7}$$

Determining the distance matrix (3.3) with DTW has a overall time complexity of $O(nmN^2)$. In some cases, this method can provide undesired effects, such as the construction of a path which is far from the diagonal. This might map a large number of points to a single point. This issue can be overcome by restricting the warping path close the diagonal (Cassisi et al., 2012). Two popular global restrictions which are normally implemented are called the *Sakeo-Chiba band* and *Itakura parallelogram* (Cassisi et al., 2012; Elsworth, 2017). Both of these methods restrict the path along the yellow squares as illustrated in Figure 3.3.



(a) Sakoe-Chimba band

(b) Itakura parallelogram

Figure 3.3: Yellow cubes indicates the restricted warping path. Figures are taken from Elsworth (2017)

The mapping of points in the DTW algorithm is illustrated and visualised in Figure 3.4. The weights which is summarised in the DTW distance in Equation (3.7) is illustrated by the grey lines between the two time series. The actual weighting value corresponds to the horizontal distance (y-axis) between the connected points. The DTW distance is the sum of all of these distances from start to end. In a real example, the offset of the time series has to be removed by normalisation to make a meaningful comparison between the two time series (Rakthanmanon et al., 2012). This is especially important if the objective of clustering is similarity in shape.

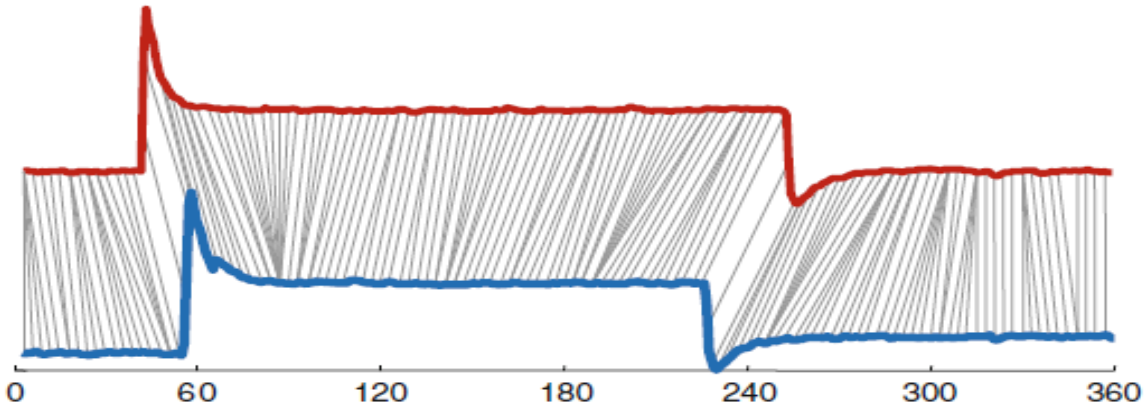


Figure 3.4: Illustration of the DTW distance between two time series Q and C. Picture taken from Elsworth (2017). Note that the DTW distance, which causes the current mapping, was calculated on the normalised time series and the mapping (in the figure) is visualised with an offset.

3.1.4 Comparison between lock-step and elastic methods

Both lock-step and elastic methods are distance measures which are categorised as shape-based dissimilarity measures. For lock-step and elastic methods, the objective of clustering time series is slightly different: Lock-step methods aim to find similarity in time; elastic methods are more generalised and the aims to find similarity in shape. This can be visualised in Figure 3.1 and 3.4. The grey lines - which indicates the mapping between the time series - are different in both of these figures. For the Euclidean distance (i.e. lock-step methods), these grey lines are vertical (i.e. one-to-one mapping). However, for the elastic methods, the grey lines are not necessarily vertical or mapped one-to-one. These are also allowed to mapped points in a one-to-many or one-to-none fashion.

Lock-step methods are limited to time series of equal length and cannot handle shifts in time. Elastic methods overcome most of the limitations which lock-step methods have. Elastic methods can find similarity between similar time series subjected to time shifting and are applicable for time series of unequal length. Generally, elastic distance measures have higher accuracy when compared to lock-step methods (Górecki, 2018; Roelofsen, 2018). However, even though these methods have an increased accuracy they also come with significantly larger time complexity. The time complexity of the different methods can be viewed in Table 3.1. As mentioned in the literature review, Euclidean distance was shown to be surprisingly competitive, despite its simplicity.

Table 3.1: Time complexities of distance measures. The length of the time series in the data set is indicated by the letter n and N is the number of time series in the data set. Letter m, in the time complexity of DTW, indicates the length of the second time series as DTW can handle time series of unequal length.

Distance Measure	Time complexity distance measure	Time complexity distance matrix
Euclidean distance	$O(n)$	$O(nN^2)$
DTW	$O(nm)$	$O(nmN^2)$

3.2 Clustering algorithms

3.2.1 K-means

K-means finds a partition such that the distance from each object to its assigned cluster centre (prototype) is minimised (Liu and Lu, 2015; Aghabozorgi et al., 2015). The prototype typically corresponds to *the average sequence of the set* and the distance measure is typically the Euclidean distance. Other combinations of similarity measures (lock-step methods) and prototypes are possible, but these do generally not produce an improvement in the results (Singh et al., 2013). The original K-means uses the former configuration of prototype and similarity measure. Thus, the objective of K-means is to find a configuration which minimises the following objective function across all *K* clusters:

$$J(c_k) = \sum_{F_i \in c_k} \|F_i - F_{\mu_k}\|^2 \tag{3.8}$$

where F_{μ_k} corresponds to the mean of the cluster c_k. In literature, the mean of the clusters is commonly known as 'centroids' (Hubert and Arabie, 1985). For time series clustering this refers to the mean vector of all the objects (time series) within that cluster. The goal of K-means is to minimise the objective function in equation 3.8 across all *k* clusters. This corresponds to minimising the following objective function.

$$J(C) = \sum_{k=1}^{K} \sum_{F_i \in c_k} \|F_i - F_{\mu_k}\|^2 = \sum_{k=1}^{K} \sum_{F_i \in c_k} d_{ki} \|F_i - F_{\mu_k}\|^2 \tag{3.9}$$

$$where,\ d_{ki} = \begin{cases} 1 & \text{if} \quad x_i \in c_k \\ 0 & \text{if} \quad x_i \notin c_k \end{cases}$$

Minimising this objective function has shown to be an NP-hard problem (Aloise et al., 2009). Because the algorithm finds a local minimum rather than a global minimum, the results are highly sensitive to the cluster centre initialisation. One way to minimise this effect is to run the algorithm multiple times with different cluster centre initialisation and chose the results which yields the lowest value of the objective function (James et al., 2013). Other hyper-parameters which affects the results of the algorithm is the number of cluster centres and choice of distance metric. Liu and Lu (2015) and Rodriguez et al. (2019) states that the most critical choice is the number of cluster centres. The choice of the correct number of clusters is in many applications not feasible to determine or available (Aghabozorgi et al., 2015). This is known as the main limitation for these methods in clustering of static data (Wang et al., 2006a) and also for time series data (Antunes and Oliveira, 2001; Aghabozorgi et al., 2015). Nonetheless, people tend to appeal to the relatively low time complexity and simplicity of the clustering algorithms which has made them very suitable for time series clustering and many works apply K-means and K-medoids to clustering of time series (Lin et al., 2004; Guo et al., 2008; Hautamaki et al., 2008; Bagnall and Janacek, 2005; Beringer and Hüllermeier, 2006). It is also sensitive to outliers which in many practical situations can be used for anomaly detection (Sequeira and Zaki, 2002).

The procedure of K-means is illustrated in Algorithm 1. The algorithm has three steps. The first step is to initialise the partition of *k* clusters. The choice of these seed point is an important consideration as an incorrect choice may give an incorrect solution. After the first step, the K-means algorithm loops through the two last steps until the termination criterion is fulfilled. Normally, the termination criterion is when no significant change in the objective function is observed. The second step is to assign each point to its closest cluster centres. The third step is to compute new cluster centres and a new value of the objective function. The implemented Python script can be viewed in Appendix D.2. Note that the algorithm does not expect a dissimilarity matrix as its input, but the time series itself. This is because the distances are calculated within the algorithm.

Algorithm 1 K-means

1: Select an initial partition of K clusters
2: **while** Termination criteria **do**
3: Assign each time series F_i to its most similar cluster centre in C.
4: Update the cluster mean and objective function
5: **return**

3.2.2 Hierarchical Clustering

Common to these hierarchical clustering algorithms is that they build a hierarchy of clusters. The way this hierarchy is built can be done in two way and hierarchical clustering algorithms are therefore divided into two subgroups: Agglomerative and divisive (Rodriguez et al., 2019). The former is more commonly used in clustering of both static data and time series (Rai and Shubha, 2010; Hastie et al., 2009). In an *agglomerative* hierarchical clustering algorithms the hierarchy is ordered bottom up. This essentially means that all objects first belongs to its own individual clusters only containing itself. Iteratively, these set of clusters are merged together based on a specified linkage criterion which will be addressed shortly. On the other hand, *divisive* hierarchical clustering algorithms start with all objects in one single cluster and then gradually separate it into smaller clusters until clusters with only one object remain (top-down) (Rodriguez et al., 2019; Aghabozorgi et al., 2015). In this thesis, agglomerate hierarchical clustering will be applied.

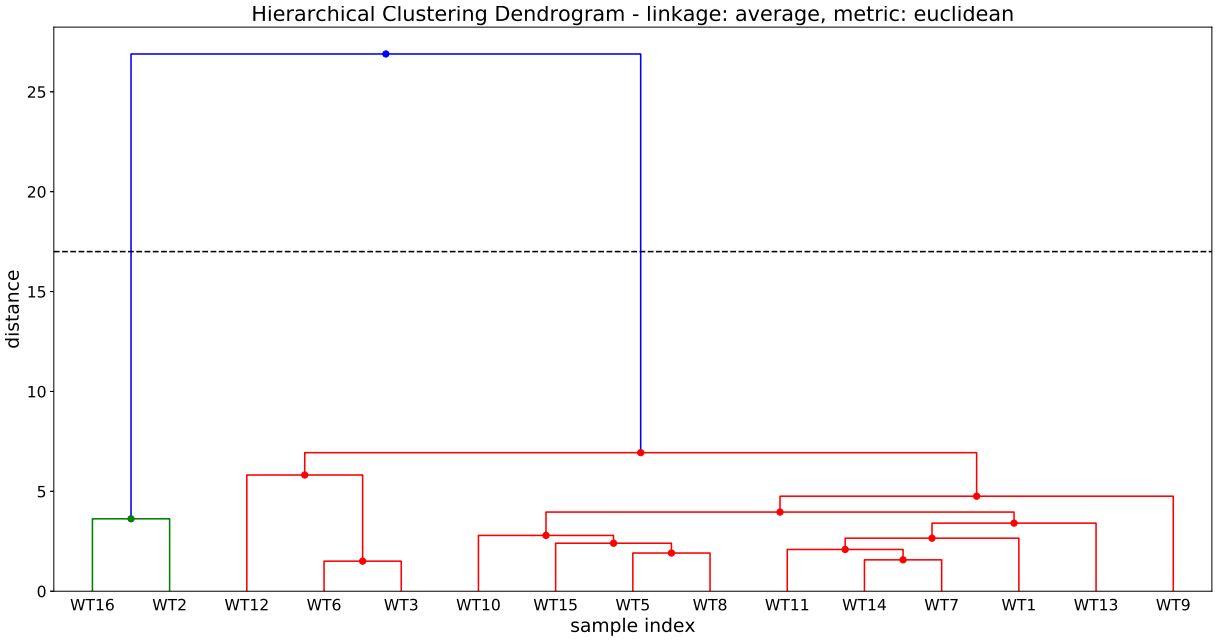


Figure 3.5: Illustration of a dendrogram with a cut to form two clusters. The linkage criterion is set as average and distance as Euclidean distance.

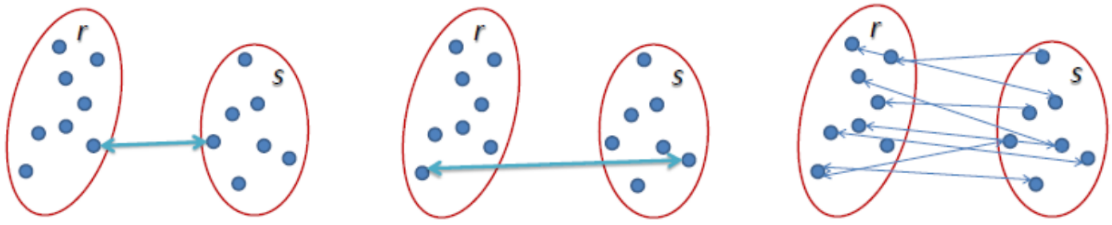
Common to all hierarchical clustering methods is the construction of a tree-based representation called *dendrogram*. The x-axis corresponds to all the different objects clustered and the y-axis corresponds to the distance between these time series. Clusters are obtained by making a cut in the dendrogram. For example, the dashed horizontal line in Figure 3.5 illustrates a cut which divides the objects into two clusters: {16, 2} and {12, 6, 3, 10, 15, 5, 8, 11, 14, 7, 1, 13, 9}. The height at which two objects are merged together corresponds to the distance between these two clusters. These distances are defined with the help of an internal distance function and a linkage criterion which states how the difference between two objects is measured.

The advantage of hierarchical clustering algorithm for time series or static data is that it does not require the user to input any parameters such as with many of the other clustering techniques (Aghabozorgi et al., 2015; Wong, 2015). The number of cuts is determined by viewing the dendrogram. This process of constructing the dendrogram is deterministic, which means that, given a dissimilarity matrix, it would always result in the same assignment/results (Sardá-Espinosa, 2017). A dendrogram is also a great tool for visualising the time series and their (dis)similarities. Much information can be obtained by simply viewing the dendrogram. Cutting the dendrogram can be performed by either visually evaluate the height at which the largest change in dissimilarity occurs, specify the desired number of clusters and cutting the dendrogram appropriately, or decide on an upper threshold value which corresponds to the largest dissimilarity allowed within each cluster and cutting the dendrogram appropriately. However, Hastie et al. (2009) states that the dendrograms may be deceptive, as a hierarchical structure is imposed by the algorithm even if such structure is not inherent to the data. That is why when the number of clusters will be determined by reviewing a set of internal indexes.

Linkage criterion

The linkage criterion in hierarchical clustering determine the pairwise distances between the clusters (Ahmad and Khan, 2018b; Vlachos et al., 2003). That is, the linkage criterion determines how the distance between each merged object should be measured. According to Hubert and Arabie (1985), the linkage criterion between the objects is divided into four groups: single, complete, average and ward linkage. SciPy (Python library) includes several others, such as weighted, centroid and median linkage. However, in this thesis, we will consider four of the most commonly used methods for agglomerative hierarchical clustering, namely, single, complete, average and ward linkage.

- **Single linkage**: Defines the distance between clusters as the shortest distance between two objects in each cluster. The advantage of using the single linkage criterion is that it is capable of finding arbitrary shaped clusters, but because of this, it is also highly sensitive to outliers and noise (Jain et al., 1988; Roelofsen, 2018).
- **Complete linkage**: Defines the distance between clusters as the largest distance between two objects in each cluster. In contrast to single linkage criterion, the complete linkage criterion is less sensitive to outliers and noise but cannot handle arbitrary shaped clusters (Roelofsen, 2018).
- **Average linkage**: Defines the distance between clusters as the average distance between each object in the first cluster to every object in the second cluster.
- **Ward linkage**: Defines the distance between two clusters as the increase in the sum of squared error from merging the two (Roelofsen, 2018). The ward linkage minimises the sum of squared differences within all clusters. In other words, it minimises the total within-cluster variance of the clusters being merged.



(a) Single linkage **(b)** Complete linkage **(c)** Average linkage

Figure 3.6: Different linkage criteria for hierarchical clustering. Taken from Sayad (n.d.).

Agglomerative hierarchical clustering

In an *agglomerative* hierarchical clustering algorithms the hierarchy is ordered bottom-up. From the perspective of the dendrogram, bottom-up refers to building the tree bottom-up. This essentially means that all objects first belongs to its own individual clusters only containing itself. Iteratively, these set of clusters are merged together based on a specified linkage criterion.

Algorithm 2 Agglomerative hierarchical clustering

1: Compute the dissimilarity matrix
2: Initialise all N objects into N individual clusters.
3: **while** there are more than one cluster **do**
4: Merge the objects which are closest.
5: Update the dissimilarity matrix with chosen linkage criteria.
6: **return** dendrogram

The procedure for agglomerative hierarchical clustering is shown in Algorithm 2. These algorithms initially compute the dissimilarity matrix of all objects of size $N \times N$ and then initialises all the N objects into N individual objects. In step 4 the closest objects are merged into one cluster and then in step 5, the dissimilarity matrix of N-1 clusters are recalculated with the chosen linkage criterion. The new dissimilarity matrix now is of size N-1 $\times$ N-1. Step 4 and 5 are repeated until there is only one cluster remaining (or the dissimilarity matrix is just a scalar).

An obvious disadvantage of the algorithm is that is has a time and memory complexity of $O(N^2)$. Another disadvantage is the lack of flexibility of the algorithm; the algorithm cannot adjust the clusters after merging or splitting the objects during the agglomerative or divisive process. To remedy this, hierarchical clustering algorithms are in some cases combined with another clustering algorithm (a hybrid clustering approach) (Aghabozorgi et al., 2015; Sardá-Espinosa, 2017).

3.3 Internal evaluation indexes

As mentioned in Section 2.3.5, the evaluation indicators are divided into two main categories: The internal evaluation indicator and the external evaluation indicator. Common for all data set analysed in this thesis is that the ground truth is not known. Internal evaluation indicators are usually used if the ground truth is not available and external evaluation indicators are not possible to perform. These internal evaluation indicators will be used to find the appropriate linkage criteria and the number of clusters for the hierarchical clustering algorithm. There exist tens of internal indexes (or internal evaluation indicators). The internal indexes which will be used are the Cophenetic Correlation Coefficient, Silhouette index and the sum of squared error (SSE). The sum of squared error will be used in order to evaluate the clusters in terms of their accuracy (i.e. their intra-cluster similarities).

3.3.1 Cophenetic Correlation Coefficient

Cophenetic correlation coefficient can be used to compare dendrograms. Lapointe and Legendre (1995) states that because a dendrogram is simply a graphical representation of an ultrametric (=cophenetic) matrix, it can be compared to one another by comparing their cophenetic matrices. Lets say we have a dendrogram $\{T_i\}$ modelled by the set of original data $\{X_i\}$, then Saraçli et al. (2013) defines the cophenetic correlation coefficient c as:

$$c = \frac{\sum_{i<j}(x(i,j) - \bar{x})(t(i,j) - \bar{t})}{\sqrt{[\sum_{i<j}(x(i,j) - \bar{x})^2][\sum_{i<j}(t(i,j) - \bar{t})^2]}} \tag{3.10}$$

where $x(i,j) = |X_i - X_j|$ which is the Euclidean distance between the i^{th} and the j^{th} observations. $t(i,j)$ is the "dendrogramatic" distance between the model points T_i and T_j (i.e. the height at which the nodes connect in the dendrogram, the horizontal lines). $\bar{x}$ and $\bar{t}$ is the average of $x(i,j)$ and $t(i,j)$, respectively. The distance between the model points is specified by the linkage criteria chosen as well as the internal distance (the distance metric used to calculate the distance of the linkage criteria of the precomputed distance matrix). For instance, DTW can be used to calculate the distance matrix between all time series within the data set. Then, the distance matrix is fed into the hierarchical clustering algorithm with a chosen linkage criteria and an internal metric (usually the Euclidean distance) which calculates the model points T_i. The different linkage criteria to chose from are listed in Section 3.2.2. The cophenetic correlation coefficient refers to the correlation between the obtained results from the cluster configuration and the original distances (distance matrix) (NCSS, LLC, 2019). The value of the cophenetic correlation coefficient range between -1 and 1: An absolute value close to one indicates that the dendrograms represents a high-quality solution and an absolute value close to zero indicates no correlation and a poor fit (Teknomo, 2017); NCSS, LLC (2019) states that the values above 0.75 refer to a high-quality solution.

3.3.2 Silhouette index

The Silhouette index (Rousseeuw, 1989) assumes that the the data has been clustered into k partitions. The data set X is partitioned into k clusters, C = $\{C_1, C_2, \ldots, C_k\}$, where C_k refers to the k^{th} cluster. The silhouette value (also called *silhouette width*) of one point x is defined as:

$$s(x) = \frac{b(x) - a(x)}{\max\{a(x), b(x)\}} \tag{3.11}$$

if $|C_x| > 1$; otherwise $s(x) = 0$ ($|C_x|$ refers to the number of time series assigned to cluster C_x). This follows that the silhouette width can also be written as:

$$s(x) = \begin{cases} 1 - a(x)/b(x), & if\, a(x) < b(x) \\ 0, & if\, a(x) = b(x) \\ b(x)/a(x) - 1, & if\, a(x) > b(x) \end{cases} \tag{3.12}$$

The within-cluster mean distance is denoted $a(x)$ and is defined as the average distance between **i** and all other data points within the same clusters C_x (Starczewski and Krzyżak, 2015). The within-cluster mean distance can be interpreted as how well *x* is assigned to its cluster. On the other hand, $b(x)$ is the mean nearest-cluster distance for each sample which implies how badly x is matched to its nearest neighbouring cluster. In other words, the smallest average distance of x to all points which x is not a member of. These expressions have the following mathematical definitions:

$$a(x) = \frac{1}{|C_x| - 1} \sum_{y \in C_x, x \neq y} d(x, y), \qquad b(x) = \min_{x \neq y} \frac{1}{|C_y|} \sum_{y \in C_y} d(x, y)$$

The silhouette width in Equation (3.11) range from -1 to 1, for each object in X. This can easily be verified by reviewing Equation (3.12). A silhouette width close to 1 indicates that $a(x) << b(x)$, which means that the point x is well matched to its own cluster and poorly matched to its neighbouring clusters; the data is well clustered. If the silhouette index is close to -1 then $a(x) >> b(x)$ and x is poorly clustered; it indicates that it would be better if it was clustered in its neighbouring cluster. Lastly, if the value is close to 0 (not zero) it indicates that the object, x, lies between two clusters (Rousseeuw, 1987). The *silhouette width* for a given cluster C_k can now be defined as:

$$S(C_k) = \frac{1}{|C_k|} \sum_{x \in C_k} s(x) \tag{3.13}$$

Finally, the *Silhouette index*, as outlined in Rousseeuw (1989), is defined as:

$$SI = \frac{1}{K} \sum_{k=1}^{K} S(C_k) \tag{3.14}$$

3.3.3 Sum of squared error (SSE) and the MSSSE

The sum of the squared error (SSE) is exactly as stated by the K-means algorithm in Section 3.2.1. The SSE is the sum of squared distance from each time series - assigned to cluster C_k - to its corresponding cluster prototype.

$$SSE(k) = \frac{1}{|C_k|} \sum_{F_i \in C_k} \|F_i - F_{\mu_k}\|^2 \tag{3.15}$$

where F_{μ_k} corresponds to the cluster prototype of cluster C_k and $|C_k|$ is the number of time series assigned to cluster C_k. It is divided by $|C_k|$ in order to get a standardised number which is comparable across models. The prototype will vary according to the specific similarity measure chosen. When the cluster analysis is performed with Euclidean distance as the similarity measure (i.e. similarity in time), the average sequence of the set is the corresponding prototype. When the cluster analysis is performed with DTW as the similarity measure (i.e. similarity in shape) the medoid sequence of the set is the corresponding prototype. Furthermore, the sum of squared error for each cluster will be added together and divided by the number of clusters to get the Mean Sum of SSE (MSSSE) or the average SSE across all cluster. In an univariate case the MSSSE is defined as follows:

$$MSSSE_u(d) = \frac{1}{k} \sum_{j=1}^{k} SSE(j) \tag{3.16}$$

where *k* is the number of clusters and $SSE(j)$ is sum of squared error of cluster j, as defined in Equation (3.15). When calculating the MSSSE value for a multivariate cluster analysis, the MSSSE value is basically the summation of the MSSSE value for each dimension divided by the number of dimensions in the data set:

$$MSSSE_m = \frac{1}{d} \sum_{j=1, \alpha(j) \neq 0}^{d} MSSSE_u(j) \tag{3.17}$$

where d is the dimension of the data set which is the number of nonzero elements in the corresponding alpha value. Additionally, it is worth noting that the MSSSE value is only summarised if the corresponding alpha is nonzero. If the alpha for the corresponding univariate data set is zero, then it is the same as not including the time series in the analysis at all.

Chapter 4

Data acquisition and preprocessing

This chapter contains information about the software which time series will be acquired from, the limitations and consideration of the software, and how the time series are preprocessed prior to clustering.

4.1 Data acquisition using Kognifai EmPower

Time series will be extracted from a software called Kognifai EmPower. Kognifai EmPower is a turbine independent decision support system provided by Kongsberg Digital AS. It surpasses the functionality found in wind farms supervisory control and data acquisition system. The software has four modules: Performance monitoring, condition monitoring, production forecasting and wind farm control (Kongsberg Digital, 2019). However, the use of the software will be limited to data extraction and visual analysis of different time series if needed. Time series are collected from a sensory network monitoring different monitoring points on several wind turbines within wind parks in Norway. Over 100 of monitoring points for each wind turbine are monitored and stored within the database. This could, for example, be the power of the individual wind turbines, the temperature of specific components, wind speed, rotor speed and so on. Time series can be extracted with regards to a specific time interval, aggregation interval and resampling interval. First of all, the limitations and considerations will be addressed.

4.1.1 Limitations and special considerations

The software provides the user with the possibility to extract stored time series during different intervals. The stored data are real-time data which is stored over several years. There are therefore many possible intervals for extraction of data with different aggregation and resampling interval. However, the software has some limitations and consideration which needs to be taken into account before any extraction or cluster analysis is performed.

The major limitation to the extraction of data is large periods of missing values. In some cases, the interval of missing data is as long as several weeks or even months. The explanation of these large periods of missing data is that there is some kind of sensory fault in the network. Therefore, any interpolation of data during these periods is pointless and would result in invalid time series. Thus, any potential clustering of data during these periods are pointless and extraction during these intervals will be avoided at any cost. Secondly, the sampling of the sensors is sampled unevenly which will result in missing data during extraction. The sensors sample data the following way: Sensor values are monitored at a fixed sampling rate. If from the previous time step the measured value on the current time step has not changed significantly, the sensor values will not be sampled. Therefore, missing values are located in the time series where the sensors values have not changed significantly from previous values. These missing values can easily be replaced with a forward filling. Forward filling means replacing the missing values with the value from the preceding time step. Care must be taken when filling the time series. One would not want to fill in intervals where the missing values are over a larger period. As mentioned first, these periods will be avoided at all cost and the missing values of the extracted time series are assumed to be the results of the second remark.

4.1.2 Parameters for extracting time series

The software includes several options for extracting time series. Time series can be extracted at any time interval from now and a few years back and is specified by the user. Alongside this, extraction of data also has the following parameters which are set by the user:

1. **Aggregation interval**: Data aggregation is a process where raw data are collected and expressed in a summary form (IMB, n.d.; Wikimedia Foundation Inc., 2019). This can, for example, be the average, minimum, maximum, sum, etc. By default, the average aggregation will be used during extraction.

2. **Resampling interval**: Resampling involves changing the frequency of the time series observations. Resampling is further divided into either down-sampling or up-sampling. Up-sampling refers to increasing the frequency of observations. This might in many cases lead to interpolating between actual values. Down-sampling is reversed, where the frequency of observations is reduced. Multiple observation is summarised to make new aggregated values (Brownlee, 2016b). According to (Argyros and Ermopoulos, 2003), down-sampling longer time series has been shown to be fast and robust.

In short, the resampling interval refers to the sampling interval of the extracted time series and the aggregation interval refers to how these new resampling points are collected. For instance, the resampling interval could be 2 seconds where the aggregation interval is 4 seconds. Then, for every 2 seconds interval of the original time series, an aggregated data point is constructed from a 4 seconds interval at that time step. Similarly, Figure 4.1 illustrates an aggregation interval and resampling interval of both 20 minutes. These extraction methods ensure that the time series are all of equal length and also reduces some of the noise often associated with time series. Aggregation and resampling of the time series ensure that the time series are of equal length and if the time series is down-sampled some of the noise associated with the time series is reduced.

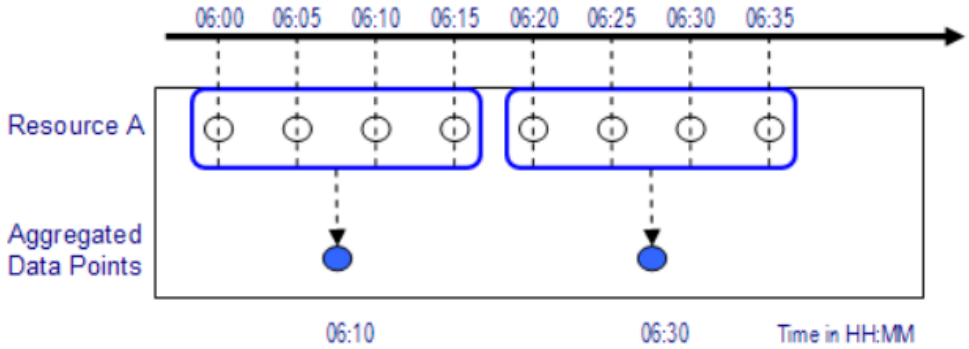


Figure 4.1: Aggregation and resampling interval of 20 minutes. The original sampling of resource A are every 5 minutes. Taken from IMB (n.d.)

4.2 Preprocessing of the extracted time series

According to Indian Agricultural Statistics Research Institute (2011), preprocessing of data is typically divided into four categories: data integration, data cleaning, data transformation, and data reduction (or representation). These categories will be addressed in the consecutive sections followed by a preferred preprocessing method.

4.2.1 Data integration

Data integration refers to the process of collecting different readings from different sources into a collected data set Indian Agricultural Statistics Research Institute (2011). For time series data this means the collection of individual time series and store them in the corresponding data sets (or databases). Particularly, time series sampled during the same interval, for different wind turbines in the same wind park (i.e. construction the univariate and multivariate data sets).

4.2.2 Data cleaning

Data cleaning mainly refers to the inconsistencies in the data such as missing values, noise and/or outliers. In real-life scenarios, time series are often subjected to quite a significant amount of noise and outliers (Zappi et al., 2007). For the extracted time series used in this thesis, we also know that it contains a lot of missing values. Proposed methods for dealing with these will be introduced now.

Missing values

As already mentioned in Section 4.1, the data may contain missing values. Because the interval of extraction is carefully inspected such that the interval does not contain large intervals of missing values (i.e. malfunctioning in the system), only scenario two from above needs to be dealt with. Since we know that the missing values located in the time series were the results of insignificant changes in the sensor values, these missing values can easily be handled with the method of *forward filling*. Forward filling simply fills in all NaN entries with the previous value. In the case where the time series have no previous values (i.e. the time series have missing values at time equal to zero) an approximation will be made where the time series is then *backward filled*; simply filling in the NaN entries with the future value measured. Any better approximation cannot be made without making assumptions about the dynamic behaviour of the turbines. Therefore, the time series will first be subjected to forward filling. If the time series contains any missing values after a forward filling, then these are leading missing values and will be filled with backward filling.

Noise and outliers

Time series are naturally noisy and may include outliers. Down-sampling which was done during the extraction of the time series reduces some of the noise associated with the time series. However, there may still be some noise in the time series which we do not wish to model. Noise can ultimately be reduced or diminished by applying digital filters or wavelet thresh-holding on the time series data (Esling and Agon, 2012). Low pass filters (or smoothers) for time series are used to reveal low-frequency trends in the time series (Stephenson, 2000). One of the simplest low pass filters in the time domain is the simple moving average (SMA) (Stephenson, 2000). This filter is categorised as a linear non-recursive (FIR) low pass filter (Gruber, 2017). It filters out most of the noise and approximates the underlying trends very well (Viswanathan, 2010). However, the SMA (and other FIR low pass filters) are quite sensitive to outliers and not well suited for suppressing outliers (Gruber, 2017). These filters can only reduce the effect of the outliers. Therefore, a simple moving median (SMM) is applied to the time series prior to the SMA. Simple moving median (SMM) efficiently suppresses outliers (Yu, 2015; Serega, 2015). Instead of taking the mean of the rolling window, the median is extracted instead. This increases the probability that noise will not affect the new smoothed time series. The SMM over n time points is given by the following equation:

$$x_i = Median(x_i, x_{i-1}, \ldots, x_{i-n+1}) \tag{4.1}$$

It is worth noting that when the SMM is centralised, the smoothing operations correspond to the *median filter*. After the SMM has been applied to the time series to remove the outliers, the additional noise can be removed by applying an SMA filter. A larger length of the SMM filter could also be increased in order to remove the outliers and much of the noise associated with the time series in one iteration.

4.2.3 Data transformation

Depending on the goal of the time series, scaling, detrending or deseasonalising the time series prior to clustering may be desired (Roelofsen, 2018). Detrending and deseasonalising the data will not be done for the time series extracted as these are important features which will help to separate them. Many clustering algorithms, such as *K-means*, are highly sensitive to scaling. Its therefore common practices in the clustering community to adjust the features such that they are more suitable for the clustering algorithms (Sarah Guido, 2013; Mori et al., 2016). Normalisation methods aim to remove the scale difference that could exist, or as Dunn (2010) states it, the reason for mean centring is to remove the arbitrary bias from measurements that we do not wish to model and scaling erases the fact that variables are measured in different units of measurements. There are many different types of procedure for normalising the data.

Caution should be taken when normalisation flat but noisy time series (Almaliki, 2018; Vlachos et al., 2002). According to Vlachos et al. (2002), normalising such time series would not guarantee the best match between two time series. This is caused by a significant amount of noise which can distort the average value and/or standard

deviation of the time series, leading to improper translation. However, with proper filtering and outlier removal, this can be avoided. Additionally, normalisation requires the user to know - or accurately estimate - the minimum and maximum for the appropriate time series (Brownlee, 2016a). Normalisation should not be used if the time series is trending up or down, as estimating of the minimum and maximum values might be difficult. The maximum and minimum values used could also be the overall maximum and minimum value observed in all of the different time series (for those parameters). This simply scales the univariate data set to simply have a maximum value of 1 and a minimum value of 0. This is different from normalising, as described above, which normalises each individual time series. Standardisation (or Z-score normalisation) and Min-Max scaling are two of the most common approaches. Z-score normalisation is a common method for normalisation within time series clustering literature (Mori et al., 2016; Aghabozorgi et al., 2015; Rodpongpun et al., 2012). The choice between normalisation and standardisation is determined by the objective of the clustering. Normalisation is appropriate if the objective is to cluster time series of similar shapes; standardisation is appropriate if the objective is to cluster time series with respect to their variance.

Standardisation (or Z-score normalisation)

Standardisation is the process which aims to rescale the features or time series such that they have the properties of a Gaussian distribution: Mean (μ) equal to zero and a standard deviation (σ), equal to one. Thus, standardisation assumes that the observations within the time series fit a Gaussian distribution. If this is not the case and the time series is standardised, the transformed time series may not be reliable (Brownlee, 2016a). The standardised values can be calculated from the following equation:

$$z = \frac{x - \mu}{\sigma} \tag{4.2}$$

Normalisation

Normalisation is often called Min-Max scaling and scales the data between 0 and 1. This normalisation is used when standardisation is not preferred. Standardisation is not preferred when the distribution is not Gaussian or the standard deviation is very small (Almaliki, 2018). Min-Max scaling is typically done with the following equation

$$z = \frac{x - \min(x)}{\max(x) - \min(x)}. \tag{4.3}$$

For clustering the time series with the objective of similarity in shape, each time series will be scaled individually to range from 0 to 1. That is, the individual time series will be scaled with their corresponding minimum and maximum value. In the case is similarity in time, the entire univariate data sets will be scaled to have a maximum value of 1 and a minimum value of 0. The maximum and minimum value observed in the entire univariate data set will be used to scale individual time series. This is done to achieve comparable internal indexes between data sets. This can be seen in the following equation

$$z = \frac{x - \min(X)}{\max(X) - \min(X)}, \tag{4.4}$$

where x is the current time series being transformed and X is the entire univariate time series. Scaling in equation (4.4) will be utilised if the data set contains flat but noisy time series or if the objective is similarity in time.

Chapter 5

Data extraction and Preprocessing

In this chapter, the univariate and multivariate data sets, which are going to be clustered and analysed, will be presented. The univariate time series is presented in Section 5.1, and in Chapter 6 these time series will be clustered and analysed. The multivariate time series is presented in Section 5.2, and in Chapter 7 these time series will be clustered and analysed. All of the time series are extracted from the Kognifai Empower software introduced in the previous chapter. The time series will be collected from a wind park with a total of 16 turbines. As one of the turbines did not contain any measurements, only 15 turbines were available for extractions. Additionally, all data sets will be preprocessed according to the preprocessing steps outlined in Section 4.2.

5.1 Univariate time series

In this section, the univariate data sets, which are going to be clustered in the next chapter, will be introduced and preprocessed. These data sets contain univariate time series, which are measurements of the oil temperature in the gearbox, extracted from different turbines, during a period of 24 hours. In order to strengthen the physical interpretation of the clustering results from clustering the first dataset, the experiment will be repeated on a different set of measurements (same turbines and sensors sampled at a different interval). Regardless of the results, it will either strengthen or weaken the physical interpretation of the first experiment - both would be equally interesting. The first univariate data set is introduced in Section 5.1.1 (from now on called 'Univariate_V1') and the second univariate data set is introduced in Section 5.1.2 (from now on called 'Univariate_V2'). The details of the extraction are listed in the following table, Table 5.1.

Table 5.1: parameters for extracting of the univariate data sets. The column for 'From' and 'To' have the following time format: [DD.MM.YYYY hh:mm:ss]

#	Name of data set	From	To	Aggregation Interval [s]	Resampling Interval[s]
1	Univariate_V1	22.01.2019 02:00:00	23.01.2019 02:00:00	30	30
2	Univariate_V2	23.02.2019 00:00:00	24.02.2019 00:00:00	30	30

5.1.1 Univariate_V1

The first data set will be a collection of time series describing the oil temperature of the gearbox (called "GearOilTempSump" in software). The time series will be stored in a 3-dimensional array whose shape is (d, k, n) where *d* is the dimension, *k* is the number of turbines, and *n* is the length of the time series. The time series are collected from a wind park consisting of 15 wind turbines in total. These individual time series are extracted over a period of 24 hours with the specific intervals specified in Table 5.1; this results in a univariate data set of shape $(1, 15, 2811)$. The time series prior to any preprocessing can be viewed in Figures 5.1 and 5.2. The number of data points (count), mean and the standard deviations (std) are is presented in Table 5.2 for each turbine.

Table 5.2: General statistics for 'Univariate_V1' data set

	WT1	WT2	WT3	WT5	WT6	WT7	WT8	WT9	WT10	WT11	WT12	WT13	WT14	WT15	WT16
count	2881	2881	2881	2881	2881	2881	2881	2881	2881	2881	2881	2881	2881	2881	2881
mean	57.2	17.3	45.7	51.5	45.8	55.9	52.0	57.3	52.9	56.3	49.5	54.0	55.6	50.8	13.8
std	4.7	2.3	3.8	6.7	3.3	6.3	6.8	6.3	8.5	7.3	8.1	3.9	7.1	6.6	3.0

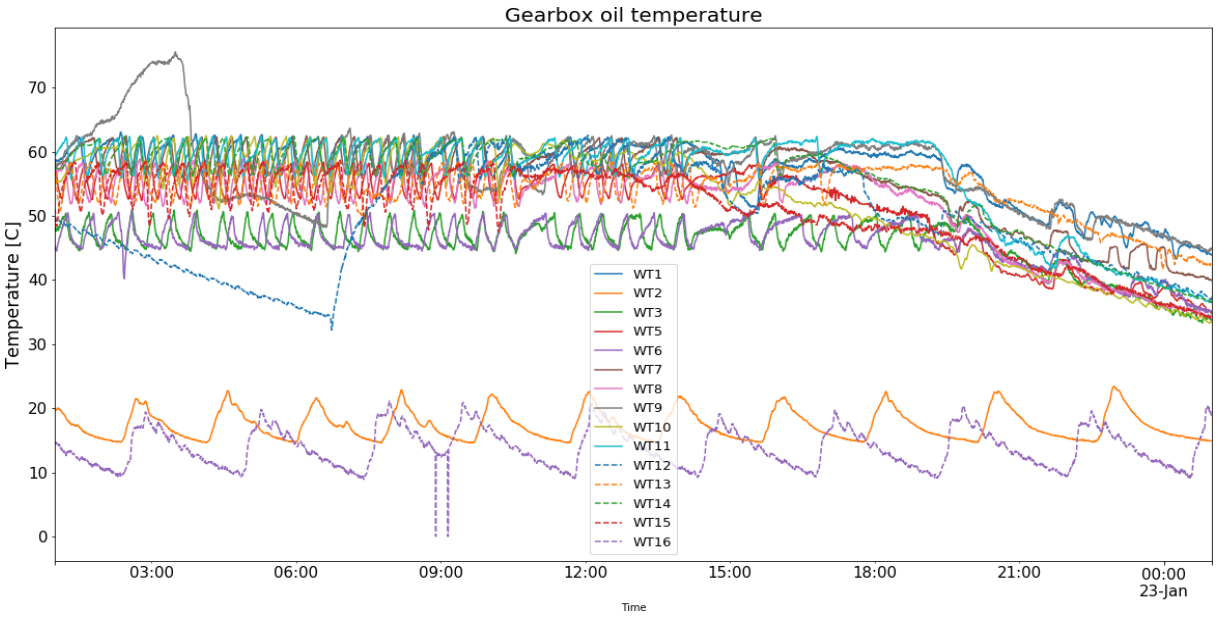


Figure 5.1: The initial time series of 'Univariate_V1' plotted during an interval of 24 hours.

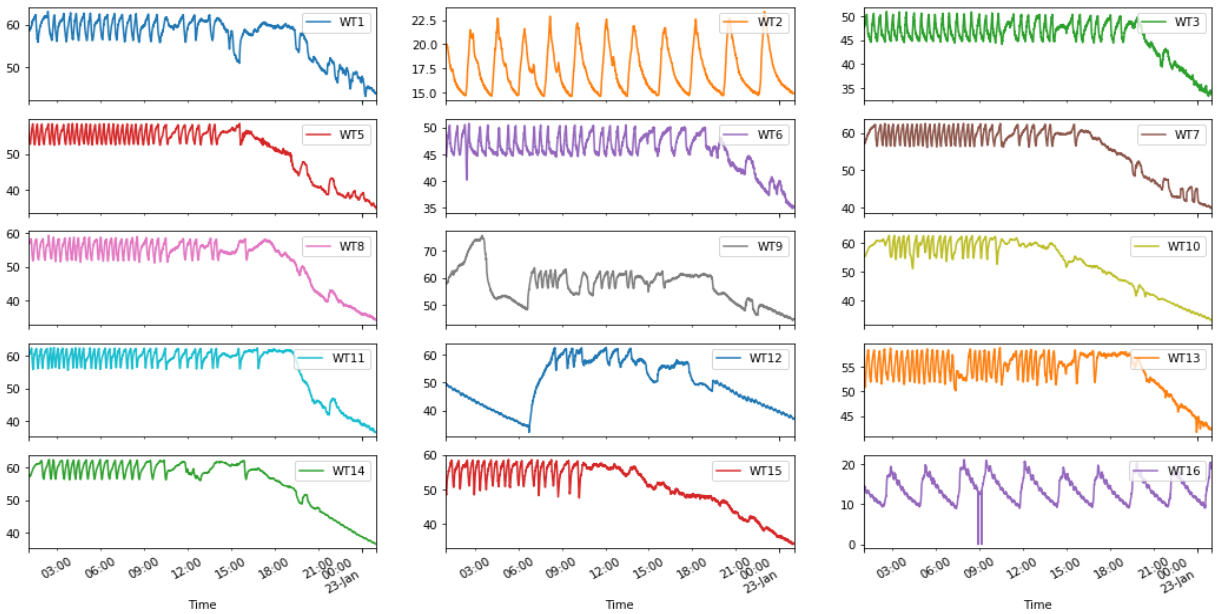


Figure 5.2: Subplots of the initial time series of 'Univariate_V1' during an interval of 24 hours.

As stated in Section 4.2, preprocessing is typically divided into four categories: data integration (this has already been done, as the data set is already collected prior to any preprocessing), data cleaning, data transformation and data reduction. The corresponding data set will be preprocessed according to the first three steps.

Data cleaning

The first step is to clean the data set. This includes handling missing values, outliers and noise present in the data set. First of all, the missing values in the data set are handled. As justified in Section 4.2, missing values will be dealt with through forward filling and then backwards filling if there were leading missing values present in the data set. Initially, 18 *NaN* entries were found in the data set; each was handled with the proposed methods. Secondly, the obvious outliers which we do not wish to model need to be removed. Obvious outliers can especially be observed for the time series for turbine 16 in both figures (see two outliers at time steps around $09:00$). These outliers - and potentially others - are removed by applying an SMM filter (Equation (4.1)) with a relatively small window size of 5. The size of the window is kept relatively low in order to only remove the obvious outliers and retain the shape and potential anomalies in the time series.

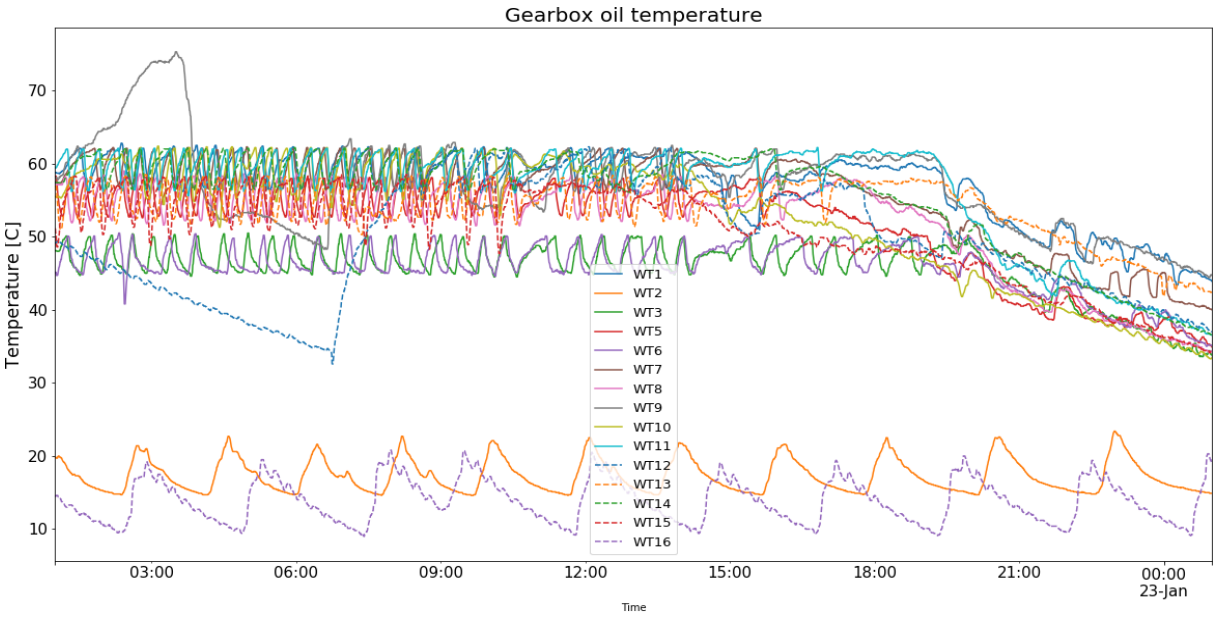


Figure 5.3: The resulting time series of 'Univariate_V1' after implementation of a simple moving median filter of length 5.

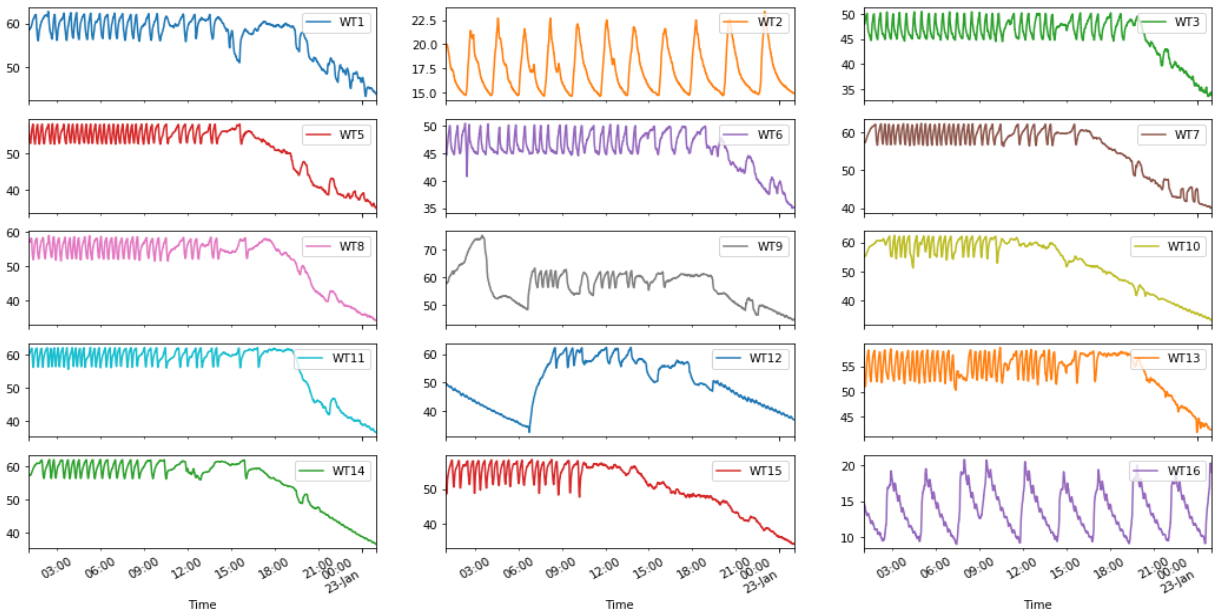


Figure 5.4: Subplots of the resulting time series of 'Univariate_V1' after implementation of a simple moving median filter of length 5.

The resulting time series can be viewed in Figure 5.3 and 5.4. It can be observed that the overall shape is maintained and any (obvious) outliers are removed and replaced. No significant noise is observed in any of the time series and therefore an SMM with a length of 5 is sufficient to remove both the outliers and the noise present in the time series.

Data transformation

Before clustering the data sets, the time series needs to be transformed appropriately. This is because many of the clustering algorithms, including K-means, are highly sensitive to scaling. Before clustering the data set with respect to the objective of similarity in time, the univariate data set is scaled between 0 and 1 (as in Equation (4.4)). The main reason for scaling the univariate data set before clustering is that it is required in the multivariate case; all univariate data sets in a multivariate data set has its features adjusted such that they range between 0 and 1. This will ensure that the calculation of the (dis)similarity matrix is correct and equally weighted across dimensions. Note that a multivariate data set is just a set of univariate data sets. This will be addressed in more details when clustering the multivariate data sets. Another reason for scaling the data set is that by scaling all data set equally, the clustering models and internal indexes are comparable across different data sets. The scaled time series can be seen in Figure 5.5. It can be observed that the plots for the scaled time series and the plot for the unscaled time series in Figure 5.3 are identical, only the y-axis values are different. The scaled time series can be observed to maintain the overall shape of the time series without removing their offset or trend. These scaled time series will be clustered when the objective is similarity in time.

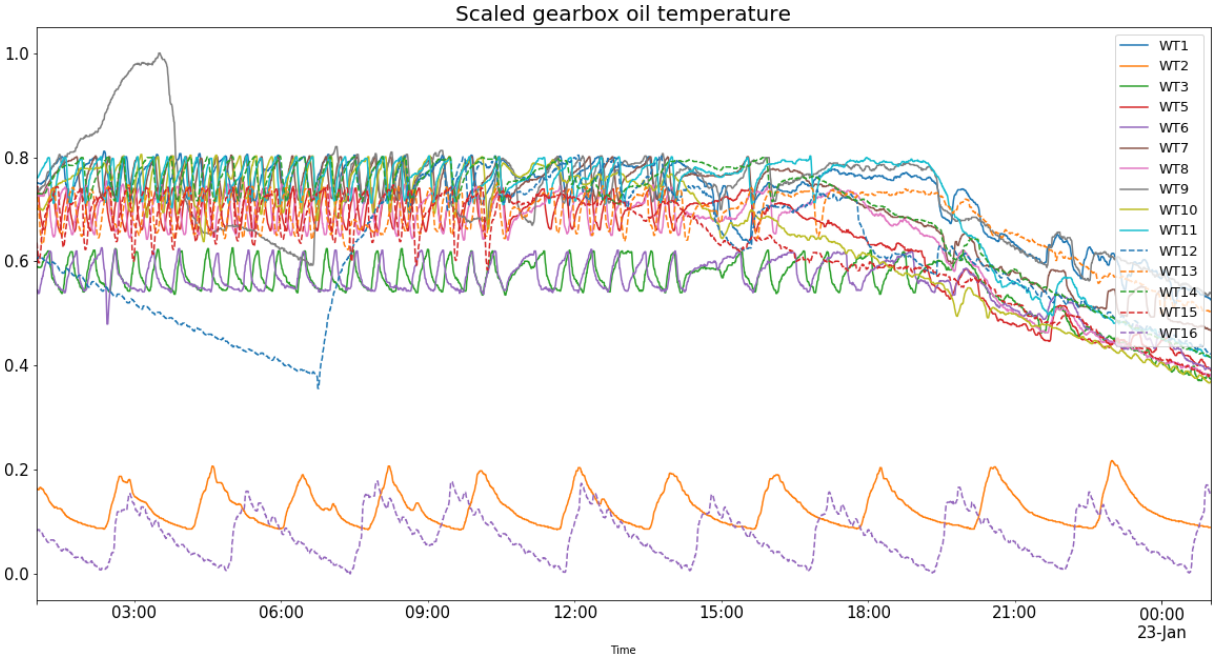


Figure 5.5: Time series of 'Univariate_V1' after scaling:. Data set is scaled between 0 and 1

In the case of similarity in shape - as long as there is no flat but noisy time series present - the time series needs to be normalised according to Equation (4.3). This scales individual time series between 0 and 1, and not the entire data set as done above in Figure 5.5. The normalised time series can be viewed in Figure 5.6. It can be observed that those experiencing similar trends are mostly overlapping. Similarly, those experiencing no trends are also overlapping but are also quite different compared to the others. This is exactly what we want when clustering the time series with respect to similarity in shape. Furthermore, if the objective is to cluster the variance of the different time series, standardisation should be used. As the time series will not be clustered with regards to their variance, the plot is only presented to make a comparison between the standardised and the normalised time series. The observation to be made is that the normalised time series are more offset than those of the standardised. This essentially is the reason that clustering the standardised time series cluster the time series with respect to their (dis)similarity in variance, rather than their shape.

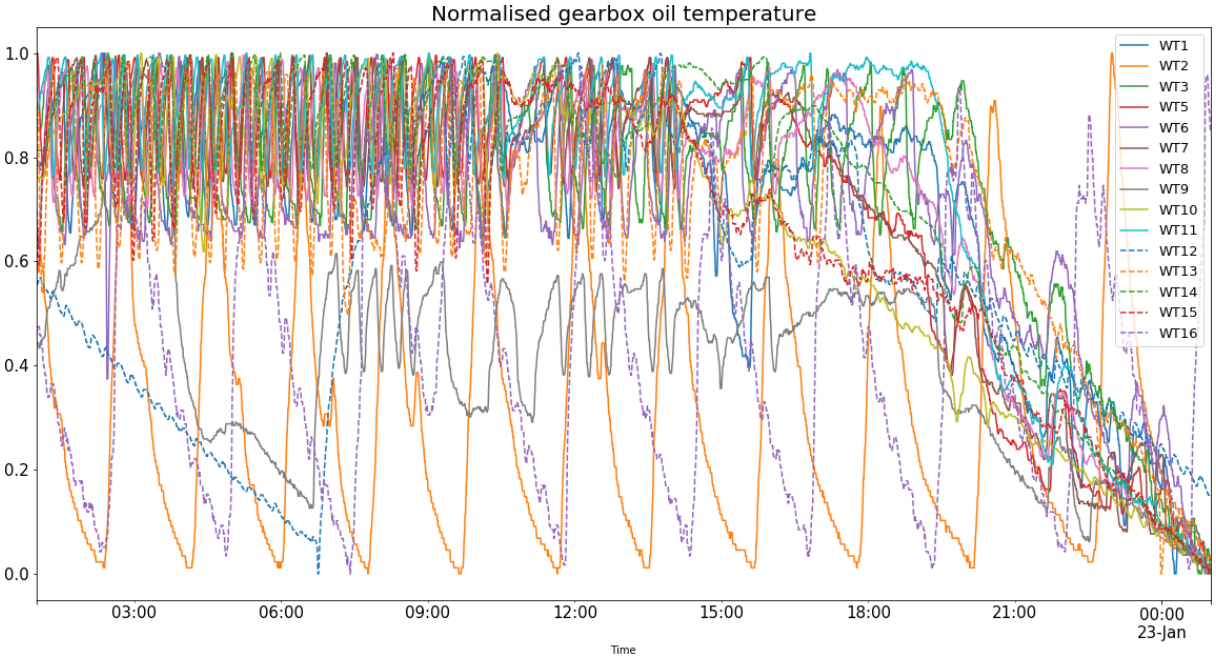


Figure 5.6: Time series of 'Univariate_V1' after normalisation: time series is scaled between 0 and 1

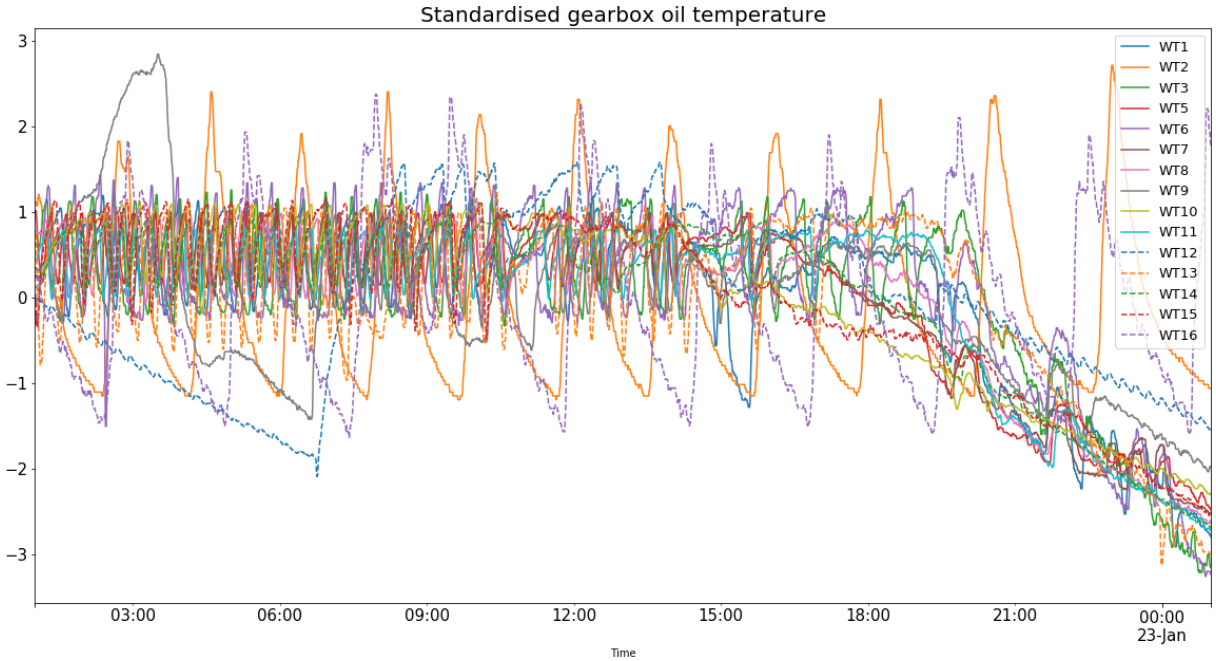


Figure 5.7: Time series of 'Univariate_V1' after standardisation: mean equal to zero and a standard deviation equal to one

5.1.2 Univariate_V2

The second univariate data set which is going to be clustered is 'Univariate_V2'. The data set is identical to that of the previous data set in Section 5.1.1, only the interval of extraction is different. Therefore, the shape of the data set is similar to 'Univariate_V1'; the shape is $(1, 15, 2811)$. The time series collected can be viewed in the Figures 5.8 and 5.9. The number of data points (count), mean and the standard deviations (std) are is presented in Table 5.3 for each turbine.

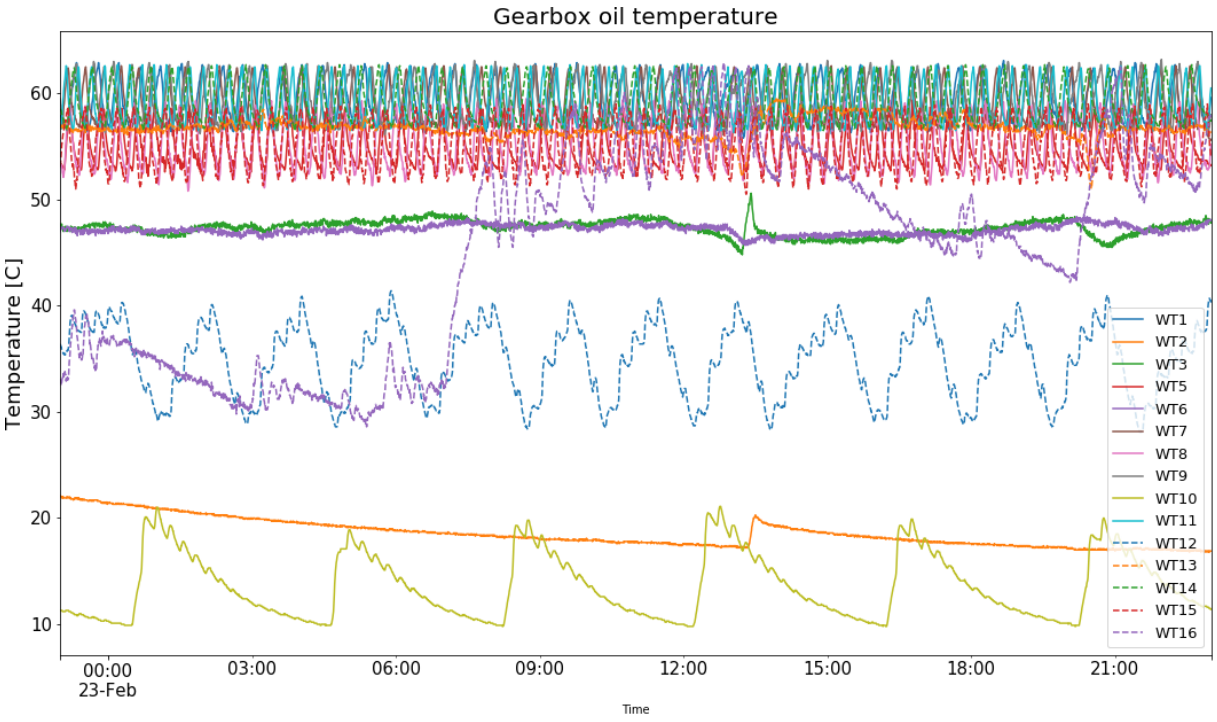


Figure 5.8: The initial time series of 'Univariate_V2' plotted during an interval of 24 hours.

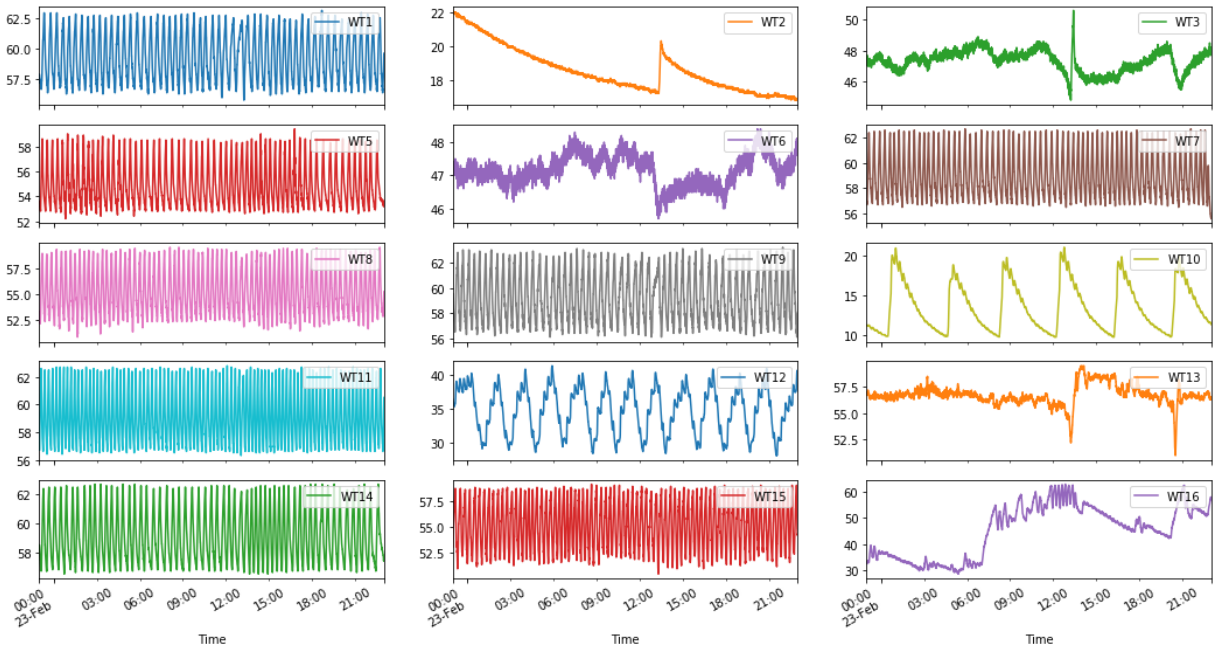


Figure 5.9: Subplots of the initial time series of 'Univariate_V2' during an interval of 24 hours.

Table 5.3: General statistics for 'Univariate_V2' data set

	WT1	WT2	WT3	WT5	WT6	WT7	WT8	WT9	WT10	WT11	WT12	WT13	WT14	WT15	WT16
count	2881	2881	2881	2881	2881	2881	2881	2881	2881	2881	2881	2881	2881	2881	2881
mean	59.3	18.6	47.3	55.0	47.2	59.0	55.3	59.3	13.6	59.2	34.8	56.737	59.2	55.2	45.3
std	2.0	1.3	0.7	1.8	0.5	1.8	2.3	1.9	3.1	1.9	3.4	1.0	1.8	2.3	9.8

Same procedure as for the 'Univariate_V1' will be followed when preprocessing this data set: Data integration (this has already been done, as the data set is constructed prior to any preprocessing), data cleaning and data transformation. The corresponding data set will be preprocessed according to these three steps.

Data cleaning

As before, the first step is to handle the missing values in the data set. Same procedure as before is followed: Forward filling is first applied followed by backward filling for the remaining missing values. Some noise can be observed in several of the time series, but no significant outliers are visually detected. As we do not wish to model the noise or have outliers present (even though no significant outliers is observed), a simple moving median filter is applied to the time series with a short length of 5. This removed some of the noise associated with some of the time series and any possible outliers which were not found visually is removed. The results of the filtering can be observed by viewing the normalised time series in Figure 5.10.

Data transformation

Before clustering the data sets, the time series needs to be transformed appropriately. The two normalisation methods introduced in Section 4.2.3 are applied to the filtered data set. For clustering the time series with the objective of similarity in time, the data set is scaled between 0 and 1 as in Equation (4.4). Scaling the data will simply just replace the y-axis values in Figure 5.8 to range between 0 and 1. Therefore, the scaled data set can simply be visualised be looking at this figure and replacing the y-axis values.

In the case of similarity in shape - as long as there is no flat but noisy time series present - the time series needs to be normalised according to Equation (4.3). Reviewing the time series, there are some time series which have relatively low standard deviation and might be in the danger zone of being too flat. However, the difference between the standard deviations of the different time series are not significant and normalisation of the time series proceeds. The normalised time series can be viewed in Figure 5.10. Normalising the time series are analogous to changing all the y-axis values of Figure 5.9 to range between 0 and 1. None of the time series are observed to be distorted because of the translation; the time series corresponding to turbine 6 had the lowest standard deviation and are the most critical to observe. However, the time series for turbine 6 does not seem to be distorted from the original observation.

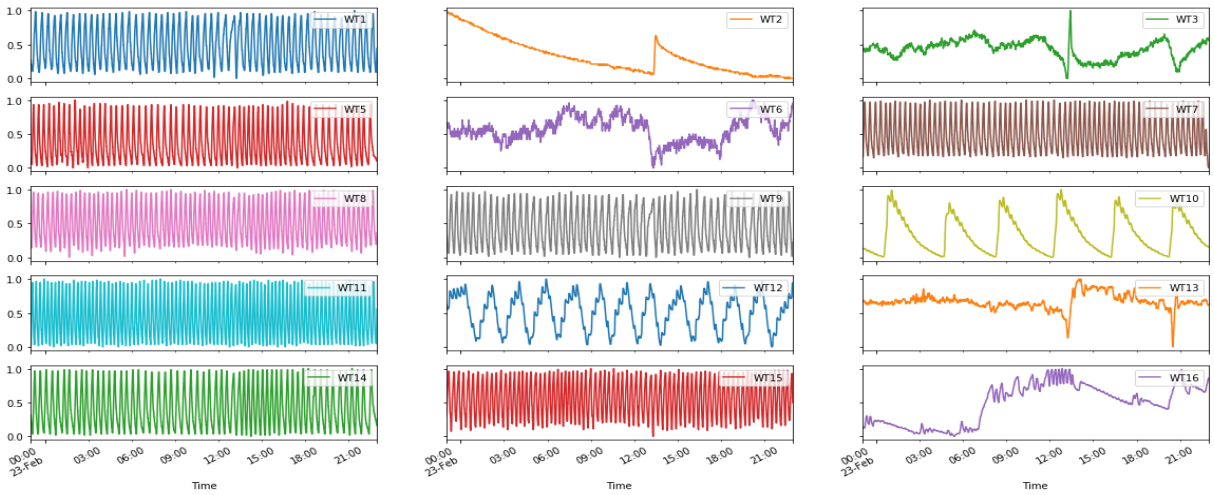


Figure 5.10: Subplots of time series of 'Univariate_V2' after normalisation. Data is scaled between 0 and 1. Note, that the subplots have shared x and y axis, in contrast to Figure 5.9.

5.2 Multivariate time series

In this section, two sets of univariate sets, which are going to be clustered in the Chapter 7, will be introduced and preprocessed. The sets of univariate time series will be collected in a 3-dimensional array whose shape is (d, k, n) where d is the dimension, k is the number of turbines, and n is the length of the time series. For example, if a multivariate data set containing three univariate data sets, each univariate data set contains 15 univariate time series (i.e. measurements on 15 different wind turbines), all of length 2000, would results in a multivariate data set of shape $(3, 15, 2000)$. All of the univariate time series will be extracted during the same interval and with the same aggregation and resampling interval. The univariate data sets within each multivariate data set are presented in Appendix A.2 and A.3.

5.2.1 Multivariate_V1

The first multivariate data set will from now on be called 'Multivariate_V1'. The data set is a collection of univariate time series presented in Table 5.4, along with the specific parameters for extraction. With the current time series, the shape of the multivariate data set is $(4, 15, 2881)$.

Table 5.4: Parameters for extraction of the univariate time series which 'Multivariate_V1' is comprised of are presented. The column for 'From' and 'To' have the following time format: [DD.MM.YYYY hh:mm:ss]

Name of data set	Name of univariate time series	Parameter of interest	From	To	Aggregation Interval [s]	Resampling Interval [s]
Multivariate_V1	Gearbox_temp	Temperature in Celsius	23.02.2019 00:00:00	24.02.2019 00:00:00	30s	30s
	Generator_speed	Rotation speed of generator [rpm]	23.02.2019 00:00:00	23.02.2019 00:00:00	30s	30s
	Wind_speed_nacelle	Wind speed measured at the nacelle [m/s]	23.02.2019 00:00:00	23.02.2019 00:00:00	30s	30s
	Positional_derivative	Direction of nacelle versus wind direction [degrees]	23.02.2019 00:00:00	23.02.2019 00:00:00	30s	30s

The time series of the gearbox temperature is the exact same time series as introduced in Section 5.1.2. Refer to this section for plots and preprocessing steps. The other time series are preprocessed as done in the univariate case. The univariate data sets are preprocessed individually. First, missing values are handled with forward filling (and backward filling if there were still missing values present). Then, outliers and noise are dealt with by an SMM filter of length 5. The only exception is the time series associated with the position deviation. One of the time series oscillated quite significantly between -360, 0 and 360 degrees. To handle this, phase unwrapping was applied before filtering the non-smooth signals for the position derivative. The generator speed, wind speed (measured at the nacelle) and the positional derivative before and after preprocessing can be seen in Appendix A.2. Each of the univariate data set are scaled or normalised prior to clustering.

5.2.2 Multivariate_V2

The multivariate second data set will from now on be called 'Multivariate_V2'. The data set is a collection of univariate time series represented in Table 5.5 along with the specific parameters for extraction. The interval for extraction is now increased to roughly 22 days, compared to 1 day for the 'Multivariate_V1' data set. Now we want to find similarities on a much larger time interval. With the current parameters for extraction, the shape of the multivariate data set is now $(6, 15, 6295)$. Finding a large enough interval where most turbines had non-missing values proved to be a difficult task. The intended interval was supposed to be a month, but the best interval after days of searching was a 22 days interval where data was available for all turbines except one (i.e. 15 different turbines had measurements on all variables).

Table 5.5: Parameters for extraction of the univariate time series which 'Multivariate_V2' is comprise of are presented. The column for 'From' and 'To' have the following time format: [DD.MM.YYYY hh:mm:ss]

Name of data set	Name of univariate time series	Parameter of interest	From	To	Aggregation Interval [s]	Resampling Interval [s]
Multivariate_V2	Power_mv2	Produced power in Watt	08.02.2019 01:00:00	01.03.2019 21:30:00	5m	5m
	GearboxTemp_mv2	Temperature in Celsius	08.02.2019 01:00:00	01.03.2019 21:30:00	5m	5m
	GeneratorSpeed_mv2	Rotation speed of generator [rpm]	08.02.2019 01:00:00	01.03.2019 21:30:00	5m	5m
	WindDir_mv2	Wind speed measured at the nacelle [m/s]	08.02.2019 01:00:00	01.03.2019 21:30:00	5m	5m
	WindSpeed_mv2	Wind speed measured at the nacelle [m/s]	08.02.2019 01:00:00	01.03.2019 21:30:00	5m	5m
	ExternalTemp_mv2	Outside temperature in Celsius	08.02.2019 01:00:00	01.03.2019 21:30:00	5m	5m

Each individual univariate time series within the data set is presented in Appendix A.3 after each time series is preprocessed appropriately. The preprocessing steps are as outlined in the previous sections. First, missing values are handled with forward filling (and backward filling if there were still missing values present). Some significant outliers were observed in the initial plots. An SMM with a window length of 100 was applied to these time series to remove the outliers and additional noise observed in the time series. No phase unwrapping was applied to the time series for the direction of the wind prior to applying the SMM filter as it were no need for it. This is because the angle can be observed to be different from zero, and would therefore not oscillate between 0 and 365 degrees as in the case for the position deviation of 'Multivariate_V1' data set. Each of the univariate data set are scaled or normalised prior to clustering.

Chapter 6

Clustering of univariate time series

Throughout this chapter whole time series clustering will be applied to analyse a set of univariate time series. The goal of this approach is to cluster similar time series in the same cluster and dissimilar time series in different clusters; similarity is measured in terms of either similarity in time or similarity in shape. Two univariate data sets ('Univariate_V1' and 'Univariate_V2') will be clustered and analysed in this chapter. The clustering results and analyses are presented in Section 6.1 and 6.2, respectively. In Section 6.3, the clustering results and analyses of both data set will be summarised and compared. The comparison will focus on strengthening (or weakening) the physical interpretation or assumptions of the other. The Python implementation of the hierarchical clustering algorithm used in conjunction with either Euclidean distance (i.e. similarity in time) or DTW distance (i.e. similarity in shape) is presented in Appendix D.1. A brief description of the Python code is included before presenting the procedure for running the code and then the code itself. The specific libraries used in the implementation can be seen in Appendix D.1.1, followed by class definitions in Appendix D.1.2, D.1.3 and D.1.4. In Appendix D.1.3, the class for clustering the time series with hierarchical clustering algorithm is included along with a procedure for how to cluster the time series with respect to either the objective of similarity in time or the objective of similarity in shape. The implementation of the K-means algorithm is presented in Appendix D.2. All computations and cluster analysis are performed on a HP EliteBook with Intel Core i7-3520M processor and 8GB of RAM.

6.1 Clustering of 'Univariate_V1'

The time series in Figure 5.3 are first visually inspected. The majority seems to have a mean of around 55 degrees Celsius and a decreasing trend towards the end of the interval; two time series are observed to have a mean of around 15 degrees Celsius and do not experience any decreasing trend towards the end of the interval. The obvious partitioning with respect to the temperature level could easily be achieved by clustering the time series with the objective of similarity in time. There are at least four different types of time series with respect to their shape and behaviour; these can be separated when similarity in shape is the objective of clustering. As the same time series seems to be dissimilar from the rest with respect to both the objective of similarity in time and shape, the first hypothesis is that both methods (or objectives) would yield similar results - at least for the quite abnormal time series observed in the plot. After reviewing the time series visually, the clustering objective for the time series clustering are:

- **Similarity in time**: When clustering the time series with this objective, the scaled data set is clustered. The scaling of the time series is done such that the clustering models and internal indexes are comparable across different data sets. The scaled time series maintain the overall shape of the time series without removing their offset or trend. Therefore, the overall shape and temperature levels will be clustered. The scaled data set can be seen in Figure 5.5.
- **Similarity in shape**: The normalised time series are clustered during this approach. As mentioned in Section 4.2, normalisation is appropriate if the objective is to cluster time series of similar shapes. The trend is not removed prior to normalisation as it is important to the overall shape of the time series and the behaviour of the turbine. Thus, time series which are similar in shape will be clustered together and absolute difference and offset, which is captured by the similarity in time objective, would be insignificant. The normalised data set can be seen in Figure 5.6.

In the following sections, the time series will be clustered with regards to both the objective of similarity in time and similarity in shape. The objectives will be clustered with an agglomerative hierarchical clustering algorithm as well as the K-means algorithm. These algorithms will be compared to each other and the most promising algorithm will be used in the consecutive sections. As stated above, clustering with the objective of similarity in time will be performed on the scaled data set with the Euclidean distance as the similarity measure. Clustering with the objective of similarity in shape will be performed on the normalised data set with DTW as the similarity measure. The standardised data set will not be analysed as standardising the time series assumes that the time series are of a Gaussian distribution. Additionally, clustering the standardised time series solves the objective of similarity in variance, which is not addressed in this thesis. For each cluster analyses, an attempt of analysing the physical interpretation of the cluster assignments will be performed; this is done to get the basis for summarising the physical implications of clustering the data set. The clustering result and analysis of similarity in time will be presented first: Both hierarchical clustering and K-means in Section 6.1.1 and 6.1.2, respectively. The analysis of the clustering results will be presented in Section 6.1.3. Then, the clustering results and analysis for similarity in shape will be presented and analysed in Section 6.1.4 and 6.1.5, respectively.

6.1.1 Similarity in time - Hierarchical clustering of the scaled data set

The objective is now to cluster the time series with regards to similarity in time. In this analysis, the scaled data set (noise, missing values and outliers have been dealt with prior to scaling) in Figure 5.5 will be analysed with the hierarchical clustering algorithm. The implementation of the hierarchical clustering algorithm along with the implementation of the internal indexes can be seen in Appendix D.1. The constructed dendrograms will be compared to one another by comparing their *Cophenetic correlation coefficient* (3.10), and the dendrogram with the largest cophenetic correlation coefficient will be chosen for further analysis. Recall that a cophenetic correlation coefficient close to 1 indicates a good fit - values above 0.75 is considered a high-quality solution - and values close to 0 indicate no correlation and a poor fit. The linkage criteria which will be compared are the *single*, *complete*, *average* and *ward* linkage. After the best dendrogram has been found, the optimal number of clusters (or cuts to the dendrogram) will be found by comparing the *Silhouette index* (3.14) and the sum of squared error (3.15) for the clusters formed. The sum of squared error for each cluster will be added together and divided by the number of clusters to find the mean sum of SSE, MSSSE (3.16), which will be used for comparison between cuts. The silhouette index range from -1 to 1: Value close to 1 indicate that the time series are well matched to its assigned cluster; value close to 0 (not zero) indicates that the object lies between two clusters; a value close to -1 indicates poor clustering and that the current object is better placed in the neighbouring cluster. Lastly, the MSSSE value is simply a measure on the within-cluster variance: A small value indicates that the objects (or time series) assigned are similar to each other (i.e. small within-cluster variance); a large value indicates that the variance within the cluster is significant. Now let us start clustering the first data set with respect to the objective of similarity in time. The summary plot and table for the internal indexes of cutting each dendrogram into *k* partitions - each with different linkage criterion - can be seen in Figure 6.1 and Table 6.1. The cophenetic correlation coefficient can be seen in the upper right corner in the aforementioned figure (Figure 6.1).

Table 6.1: Summary table with the linkage criteria: single, complete, average and ward. The data set which was clustered is 'Univariate_V1' with the objective of similarity in time. The silhouette indexes and MSSSE values for a cut K - from 2 to 13 - is presented.

Method	**Internal index**	**Number of cuts to the dendrogram**											
		K = 2	K = 3	K = 4	K = 5	K = 6	K = 7	K = 8	K = 9	K = 10	K = 11	K = 12	K = 13
Single	Silhouette	0.82	0.35	0.50	0.39	0.24	0.16	0.27	0.21	0.19	0.17	0.15	0.13
	MSSSE	15.17	12.50	6.26	5.70	4.30	3.61	1.63	1.23	0.86	0.58	0.36	0.20
Complete	Silhouette	0.82	0.49	0.37	0.42	0.39	0.38	0.27	0.21	0.19	0.17	0.15	0.13
	MSSSE	15.17	8.17	5.67	3.76	2.86	2.19	1.63	1.23	0.86	0.58	0.36	0.20
Average	Silhouette	0.82	0.49	0.50	0.35	**0.39**	0.28	0.27	0.21	0.19	0.17	0.15	0.13
	MSSSE	15.17	8.17	6.26	4.86	**2.86**	2.29	1.63	1.23	0.86	0.58	0.36	0.20
Ward	Silhouette	0.82	0.49	0.37	0.42	0.39	0.38	0.27	0.21	0.19	0.17	0.15	0.13
	MSSSE	15.17	8.17	5.67	3.76	2.86	2.19	1.63	1.23	0.86	0.58	0.36	0.20

First of all, reviewing the cophenetic correlation coefficient in Figure 6.19, it can be observed that all dendrograms experience similarly high values of either 0.984 or 0.985. The cophenetic correlation coefficient is close to 1 for all dendrogram; values close to 1 indicate that all the dendrograms represent a high-quality solution. Depending

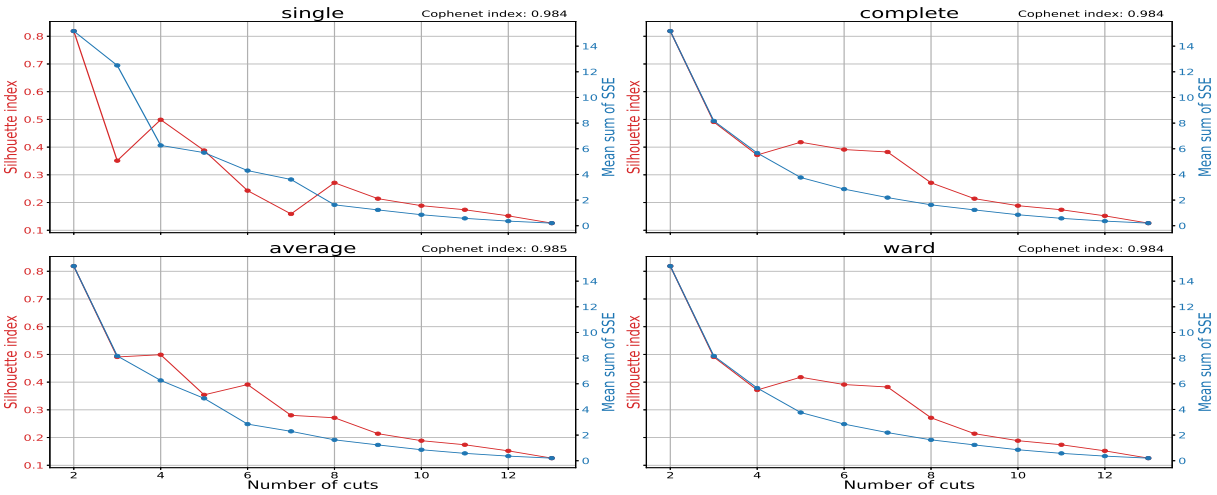


Figure 6.1: Summary plots with the linkage criteria: single, complete, average and ward. The data set which was clustered is 'Univariate_V1' with the objective of similarity in time. The silhouette indexes and MSSSE values for a cut K - from 2 to 13 - is presented along with the cophenetic correlation coefficient in the upper right corner.

on whether we want to minimise the sum of squared difference within all clusters or the separation of the different clusters, either *ward* or *average* could be used as the linkage criterion during clustering. These dendrograms have a corresponding cophenetic correlation coefficient of 0.984 and 0.985, respectively. Arguably, either of the dendrograms could be chosen for further analysis. As both of the dendrogram with average and ward as linkage criterion has similarly large cophenetic correlation coefficients, both will be interpreted. This is done in order to make a comparison between the linkage metric and the internal indexes used. Actually, all dendrograms could be chosen for further analysis, but a choice was made to further analyse these two dendrograms.

By reviewing the summary table and the corresponding summary plot, it can be observed that the silhouette index is largest for a cut of two for all linkage criteria. This indicates that a cut of two provides two clusters with the best combination of high intra-cluster similarity and large separating between them (i.e. the time series are well matched to its own cluster and poorly matched to neighbouring clusters). This can be visually verified by looking at the original plots of the time series (Section 5.1.1) where two time series are obviously different from the rest. As the physical interpretation will be analysed, deeper cuts might be helpful when doing the analysis of the cluster assignment. A number of cuts greater than two will roughly halve the silhouette index value, but the corresponding *MSSSE* decreases exponentially along with it. Note that the MSSSE value is analogous to the average within-cluster variance for all clusters formed. So when making a cut, we want the largest reduction in the within-cluster variance (i.e. the MSSSE) and the smallest reduction in the silhouette index. The large silhouette value and the large MSSSE for a cut of two indicate that the two clusters formed are well separated (large silhouette index) but the within-cluster similarity in each cluster are still quite high (i.e. a large MSSSE value). Thus, *the compromise between separation and similarity becomes an important consideration when finding the correct number of clusters*.

Now reviewing only the ward and average dendrogram. As stated above, both dendrograms result in a large silhouette index value for a cut of two. Deeper cuts will now further be analysed. First of all, when viewing the internal indexes with *average* as the linkage criterion, a cut of 4 and 6 to the dendrogram still have a relatively large silhouette index along with a significant reduction in the MSSSE value from previous cuts. A larger number of cuts than 6 to the average dendrogram does not result in a sufficient decrease in the sum of squared error or an increase or hold in the silhouette index value. Arguably, either a cut of 4 or 6 is optimal for this configuration (if we disregard a cut of 2). At a cut of 4, the silhouette index value is relatively large. However, for a cut of 6 to the dendrogram, it would approximately halve the MSSSE value while only reducing the silhouette index slightly (from 0.5 to 0.39). Arguably, a cut of 2, 4 or 6 to the dendrogram seems to be a good choice, depending on the compromise between separation and within-cluster similarity. Secondly, reviewing the indexes when *ward* is set to be the linkage criterion. A cut of 5, 6 and 7 to the dendrogram results in roughly the same silhouette index values with only a slightly decreasing in the MSSSE value across these cuts. As both - actually all dendrograms - have similar cophenetic correlation coefficients, cutting any of the dendrograms would result in a high-quality solution and roughly the same assignment. A choice has to be made between them. Depending on whether we want to minimise the sum of squared difference within all clusters or the separation of the different clusters, either *ward*

or *average* is used as the linkage criterion for the clustering. The dendrogram which will be cut is the dendrogram constructed with average as the linkage criteria because it has the largest cophenetic correlation coefficient of the two. However, both will be presented and used to make a connection between the internal indexes and the ward and average distances. These dendrograms will further be analysed in the following paragraph.

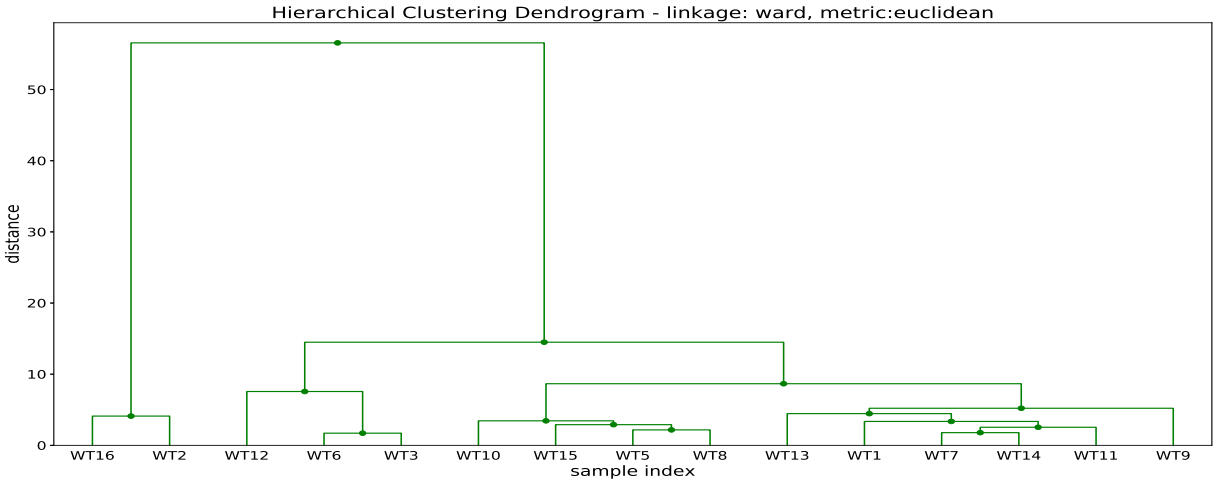


Figure 6.2: Dendrogram with *ward* as linkage criterion and distance as Euclidean distance.

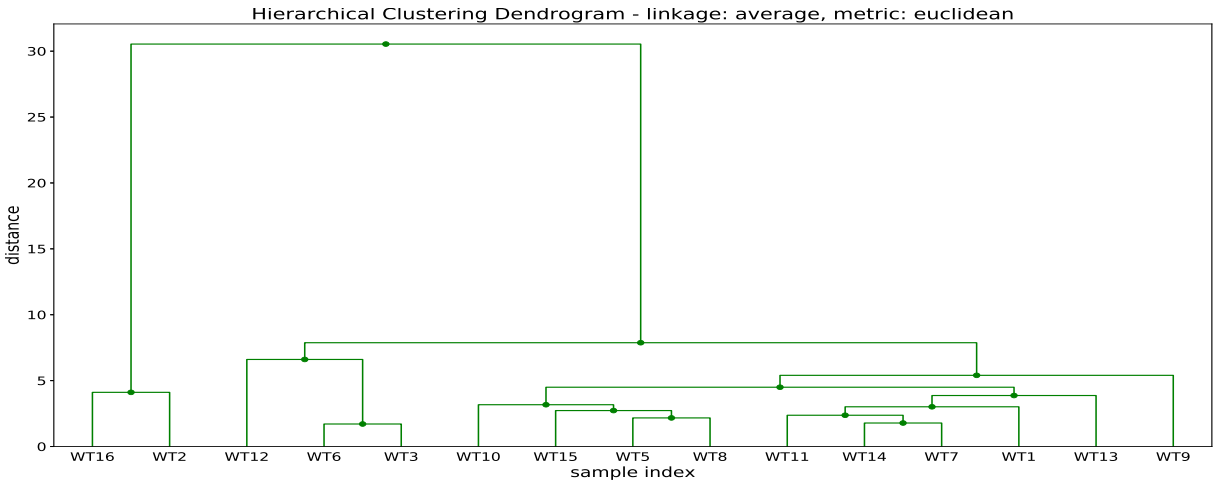


Figure 6.3: Dendrogram with *average* as linkage criterion and distance as Euclidean distance.

The dendrogram built with ward and average as the linkage criterion can be viewed in Figure 6.2 and 6.3, respectively. The entire clustering process for both of these configurations took 1.32ms. Viewing the dendrograms, the first observation to be made is that both dendrograms are very similar and make roughly the same cluster assignments. The cluster assignment is the same for both dendrograms for a cut of 2, 3 and 6 (deeper cuts than 6 is not compared). A cut of two separates the time series into two distinct clusters. This is indicated by the large reduction in the sum of squared difference within each cluster (i.e. the ward distance) and the average distance between the two clusters. This can be observed in both dendrograms by the uppermost horizontal line. The deeper the cut in the dendrogram is, the lower becomes the ward distance and similarly the average distance between the clusters. The ward distance is comparable to the MSSSE index value as both metrics reflect the within-cluster variance observed in all clusters formed (i.e. the MSSSE is highly correlation to the ward distance). The combination - or the ratio - between a large separation (the average distance between the clusters) and within-cluster variance (ward distance) is comparable to the silhouette index value. This connection will be further illustrated in the following section where the dendrogram constructed with *average* as linkage criterion will be cut. The cuts that will be further reviewed are the number of cuts equal to 2, 4 and 6. When a configuration has been found, the resulting groupings will be analysed. This is done in Section 6.1.3 where the analysis of the physical interpretation of the cluster assignment will be made.

Cutting the dendrogram to form 2 clusters

First of all, the dendrogram is cut to form two clusters, or groupings, will be reviewed. The resulting cut in the dendrogram is shown in Figure 6.4 along with colour coding for the cluster assignment (green and red). The time series belonging to both clusters are visualised in Figure 6.5. According to the Silhouette index, this is the optimal number of cuts for both dendrograms; the reason for this is the large separation between the two formed clusters. First of all, the ward distance will be reviewed. A significant reduction in the ward distance can be observed in the dendrogram (Figure 6.2) with linkage criterion as ward. The ward distance is reduced from roughly 60 (indicated by the uppermost horizontal line) to 4 and 15, respectively. The uppermost horizontal line indicates the ward distance if the two clusters were merged together to form one single cluster. This is analogous to the MSSSE value if all time series were placed in the same cluster. Similarly, the average distance in Figure 6.4 can be observed to yield significant separation between those two clusters. The large reduction in the ward distance and the clear separation between the two clusters explains - analytically - the large silhouette index value, 0.82, observed for a cut of two to both the average and ward dendrograms. The corresponding overall MSSSE value of 15.17 for this cut is highly correlation to the values read from the second and third uppermost horizontal line of each cluster formed. That is, the uppermost horizontal line (green) for the first cluster and the uppermost horizontal line (red) for the second cluster in Figure 6.4. The difference is that MSSSE distance is calculated from the cluster prototype and the ward distance is the within-cluster variance. As mentioned above, these are highly correlated: A reduction in the ward distance results in a reduction in the MSSSE value.

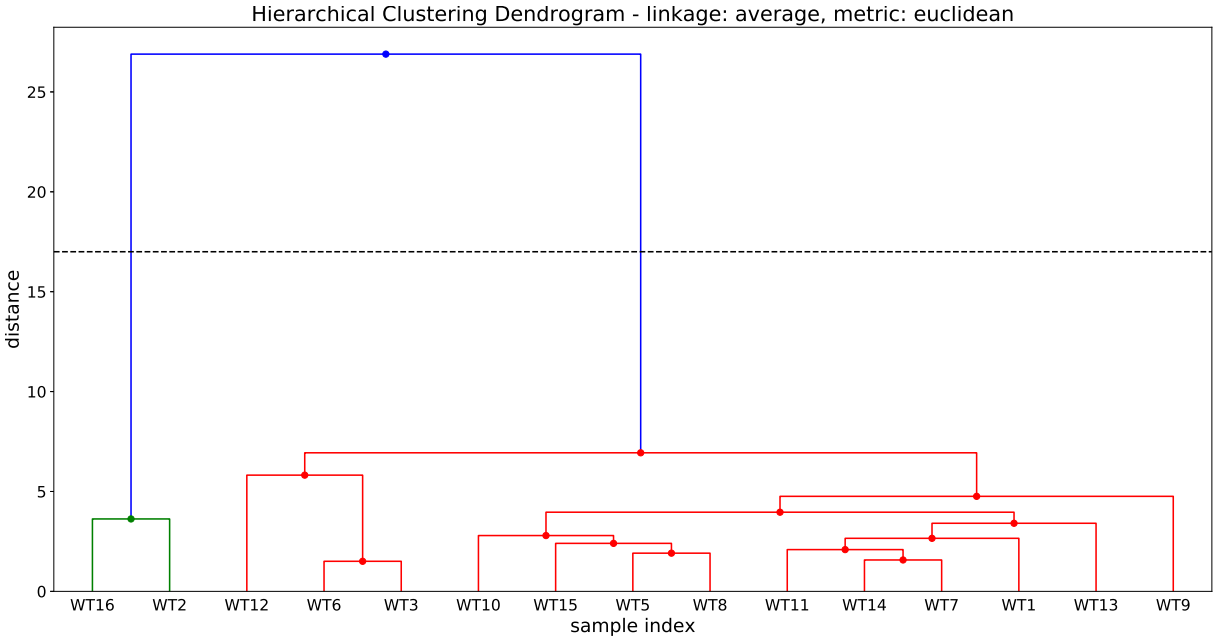
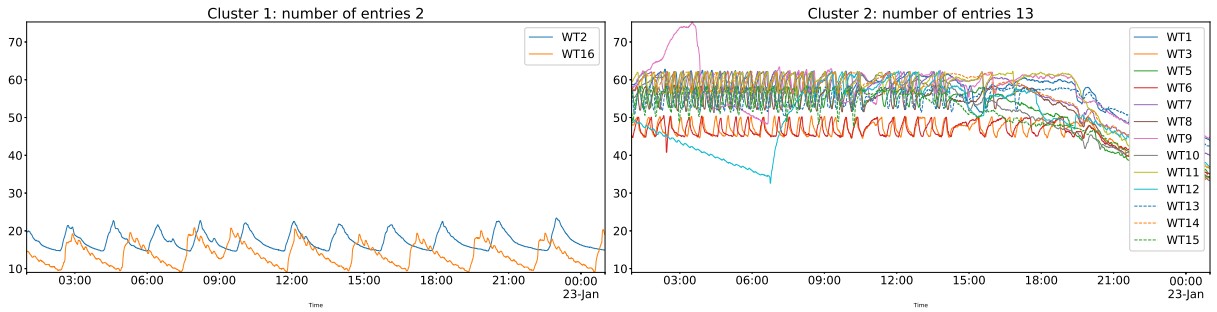


Figure 6.4: Dendrogram with average as linkage criterion and distance as Euclidean distance.



(a) **(b)**

Figure 6.5: Clusters formed by cutting the dendrogram to form two clusters.

Cutting the dendrogram to form 4 clusters

Now the average dendrogram is cut to form four clusters. The resulting cut in the dendrogram is shown in Figure 6.6, along with colour coding for the cluster assignment (green, red, dark blue and turquoise/light blue). The time series belonging to each cluster is further visualised in Figure 6.7. From viewing the average dendrogram (Figure 6.6), the three new clusters (dark blue, red and turquoise) have now an average separation which is significantly lower than that of a cut of two. The average distance between dark blue and turquoise can be read from the dendrogram where the tree merges (third uppermost horizontal line read as roughly 6). The average distance between these two clusters (dark blue and red) and the turquoise cluster can be found by reading of the distance for where the second uppermost horizontal line merges (i.e. an average distance of roughly 7). As cutting the ward dendrogram with $K = 4$ does not yield the same cluster assignment, the ward distance cannot be interpreted the same way. However, as the MSSSE is comparable to the ward distance, interpretation of the MSSSE yields similar interpretation value. From the MSSSE value, a reduction from 15.17 to 6.26 can be observed. The MSSSE value is more than halved from a cut of two. The corresponding silhouette index value suffers a significant reduction from 0.82 to 0.50 which indicates that the newly formed clusters are less separable than the assignment of a cut of two. But since the MSSSE value for the new clusters has decreased by close to a factor of two, the reduction of the silhouette value is considered to be negligible (i.e. separation between the clusters are less weighted than the within-cluster similarity). Next, the dendrogram will be cut to form six clusters.

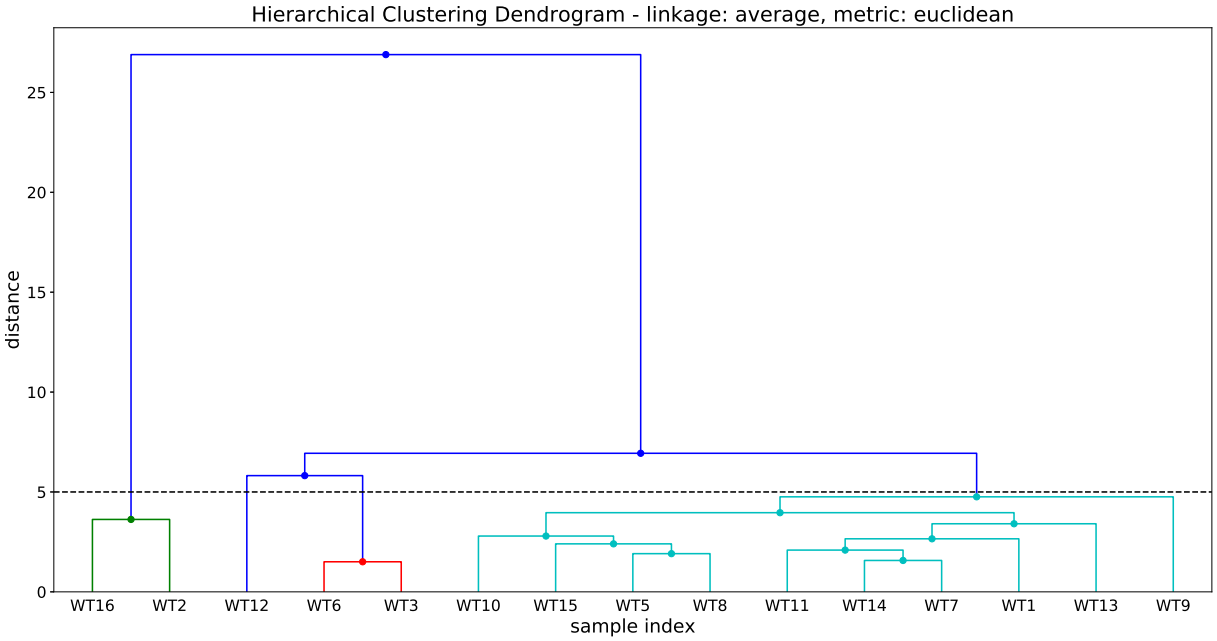


Figure 6.6: Dendrogram with average as linkage criterion and distance as Euclidean distance.

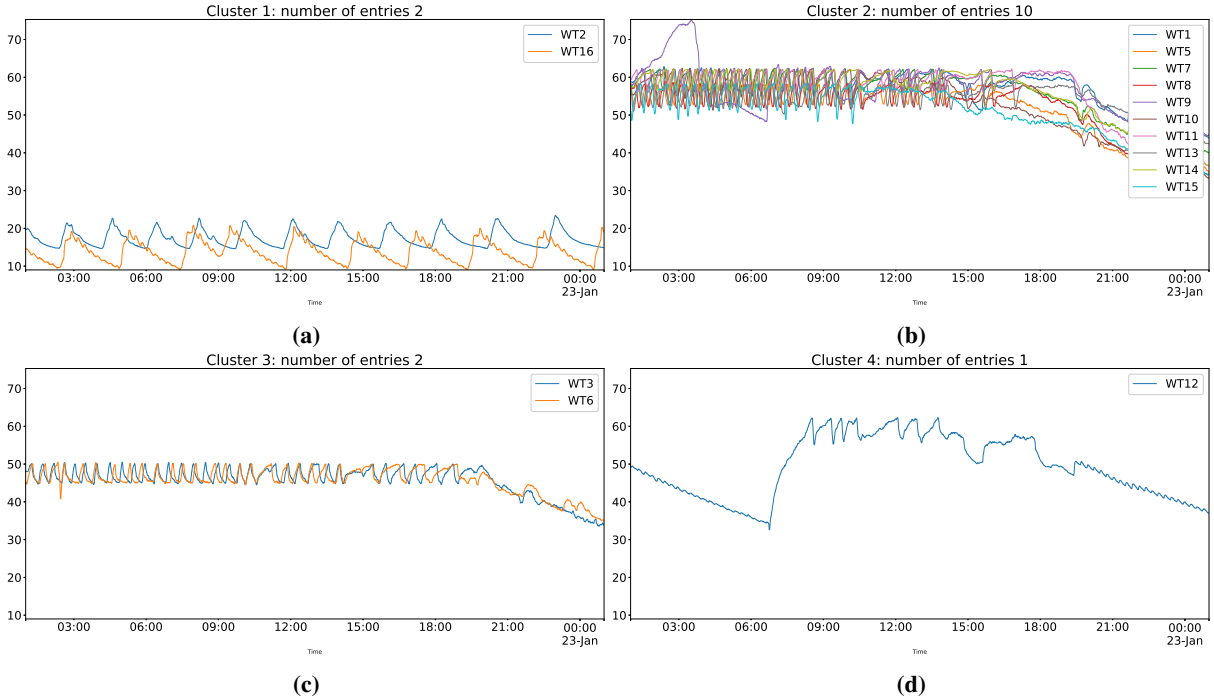


Figure 6.7: Clusters formed by cutting the dendrogram to form four clusters.

Cutting the dendrogram to form 6 clusters

Now the dendrogram is cut to form six clusters. The resulting cut in the dendrogram is shown in Figure 6.8 along with colour coding for the cluster assignment (green, dark blue (1), red, turquoise/light blue, dark blue (2), purple). The time series belonging to each cluster is further visualised in Figure 6.9. First of all, the silhouette value - compared to a cut of four - ha decreased from 0.5 to 0.39. On the other hand, the corresponding MSSSE value is reduced by a more than a factor of 2, from a value of 6.26 to a value of 2.86 for a cut of six. From viewing the corresponding dendrogram (Figure 6.2), low ward distances for each cluster formed clusters can be observed (ward distance between 1 and 5). The significant reduction in the silhouette value can be explained by a less significant separation between the newly formed clusters; this can also be seen in the average distances in Figure 6.9. Because the silhouette index and the MSSSE value are more descriptive than reviewing the ward and average distances in the dendrogram and that a hierarchical structure could be imposed by the algorithm even if such structure is not inherent to the data (Hastie et al., 2009), the choice for the number of cuts to the dendrogram will primarily be determined by evaluating the internal indexes, rather than viewing the dendrograms. Furthermore, a cut of seven to the ward dendrogram could indicate - from the silhouette index - to be beneficial. However, only a small reduction in the MSSSE value is observed. The dendrogram constructed with average as linkage criterion still has the highest cophenetic correlation coefficient (even though its just an 0.001 improvement). Looking at the corresponding silhouette and MSSSE value for that dendrogram shows that a deeper cut of seven would not yield a better compromise between separation and within-cluster similarity. The optimal configuration for the hierarchical clustering algorithm is to cut the dendrogram with average as the linkage criterion into six partitions. The following cluster assignment will be analysed in Section 6.1.3. But first, the same cluster analysis will be repeated on the K-means algorithm with the same number of clusters found in this section.

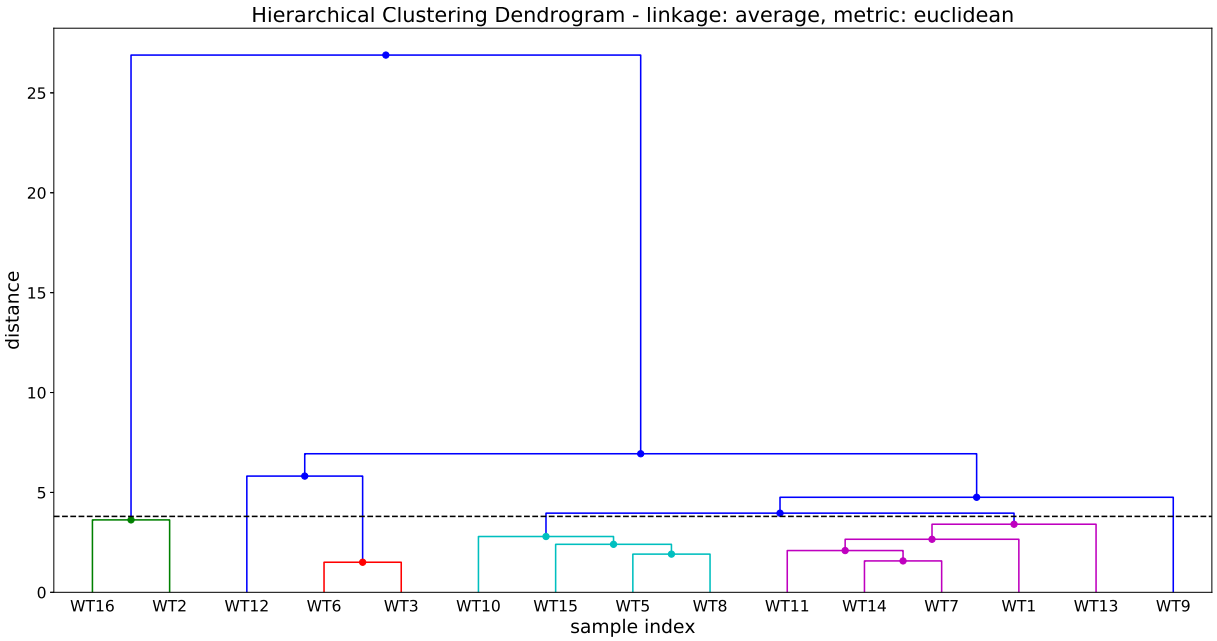


Figure 6.8: Dendrogram with average as the linkage criterion and distance as Euclidean distance.

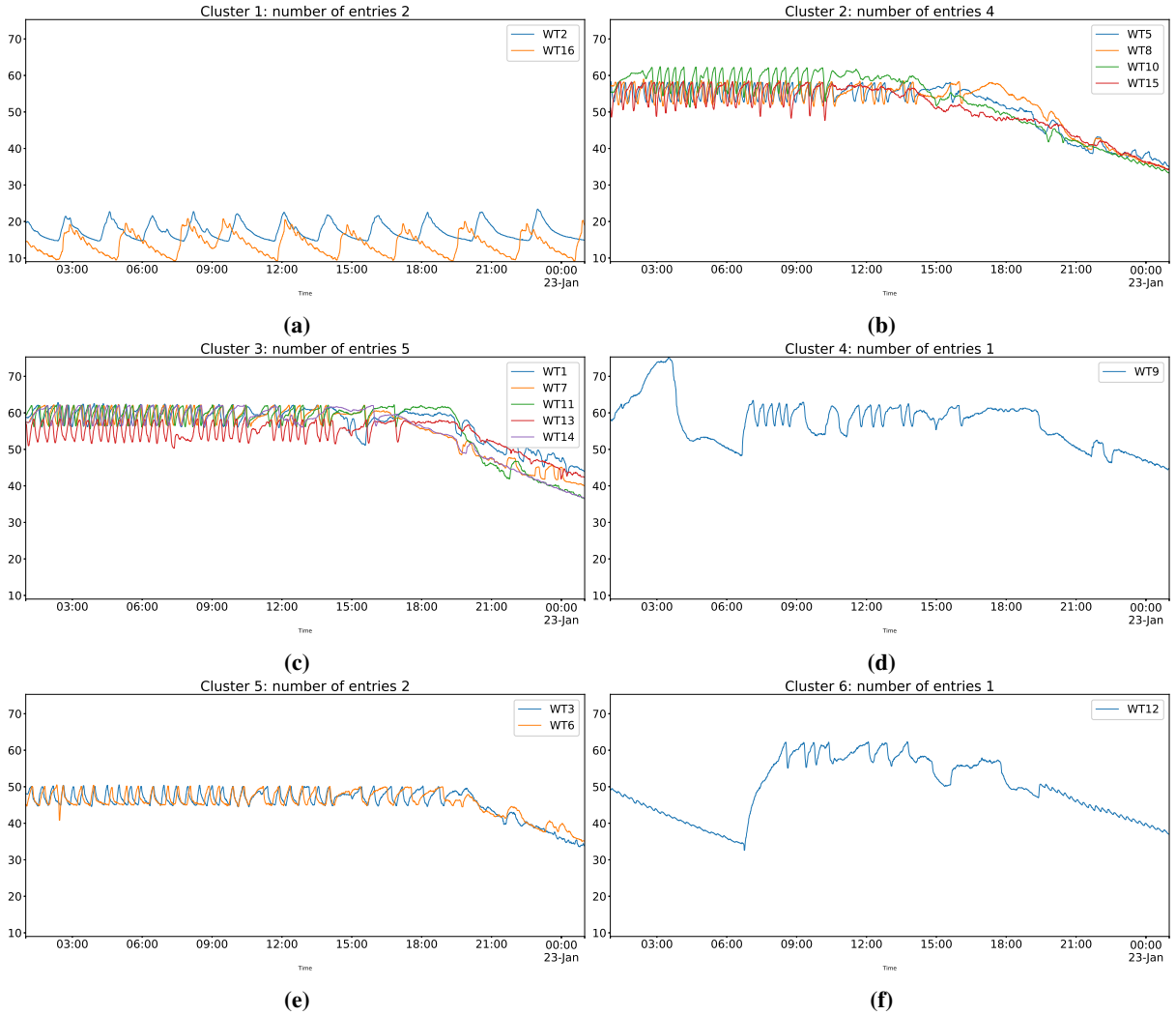


Figure 6.9: Clusters formed by cutting the dendrogram to form six clusters.

6.1.2 Similarity in time - K-means on the filtered data set

In this section K-means from Section 3.2.1 will be implemented on the 'Univariate_V1' data set. The implementation of the algorithm can be seen in Appendix D.2. The prototype for the K-means algorithm is the average sequence of the set. Thus, the cluster centres calculated are simply the average vector of all time series assigned to the same grouping. These will be presented along with the assignment of the clusters. The Euclidean distance is used as the distance measure. In the consecutive sections, K-means is implemented on the data set with the number of clusters found from reviewing the internal indexes from the hierarchical cluster analysis ($K = \{2, 4, 6\}$).

K-means implementation - K equal to 2

First of all, K-means algorithm is implemented on the data set with a number of cluster K equal to 2. The cluster centres are plotted in Figure 6.10. Similarly, the corresponding time series assigned to each cluster are plotted along with the cluster centre in Figure 6.11. The running time for the K-means algorithm with $K = 2$ was 1.11ms, which is slightly smaller than that of the hierarchical clustering algorithm. This is expected as the time complexity of K-means is smaller than the hierarchical clustering algorithm.

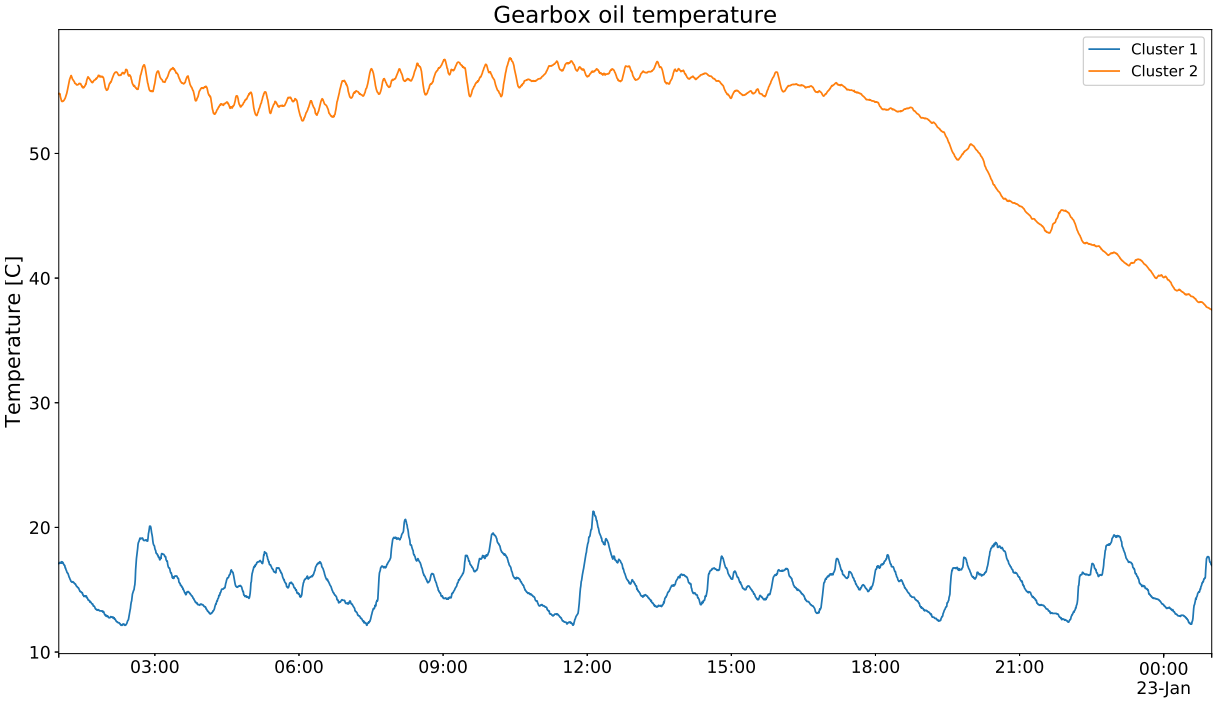
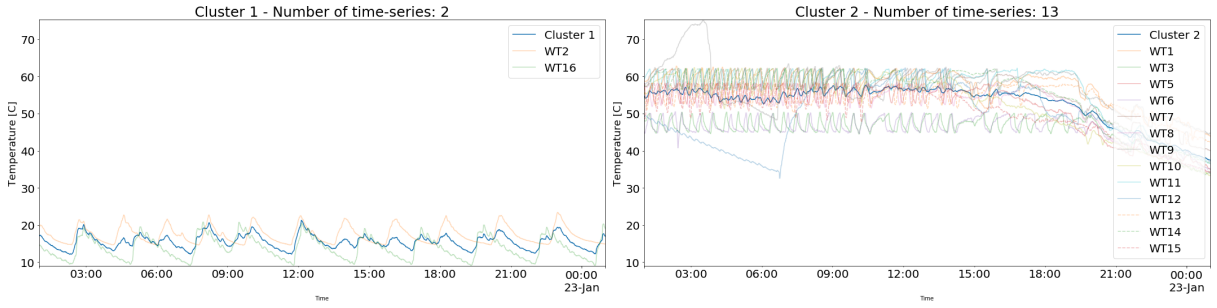


Figure 6.10: Cluster centres obtain by the K-means algorithm where $K = 2$.



(a) **(b)**

Figure 6.11: Clusters formed when $K = 2$ in the K-means algorithm.

K-means implementation - K equal to 4

Now, K-means is implemented on the data set with a number of cluster K equal to 4. The cluster centres are plotted in Figure 6.12. Similarly, the corresponding time series assigned to each cluster are plotted along with the cluster centre in Figure 6.13. The running time for the K-means algorithm with $K = 4$ was 1.25ms. It can be observed that the running time has increased from the previous cut. This is because of the increased number of prototype calculations.

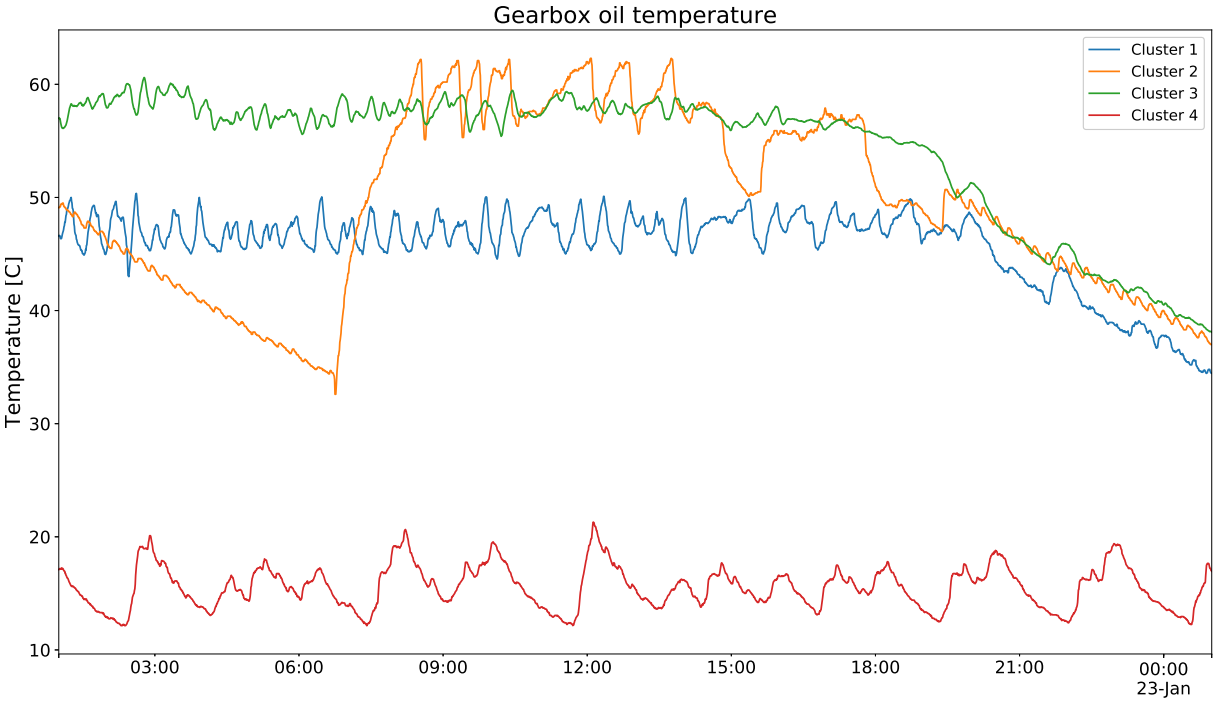


Figure 6.12: Cluster centres obtain by the K-means algorithm where $K = 4$.

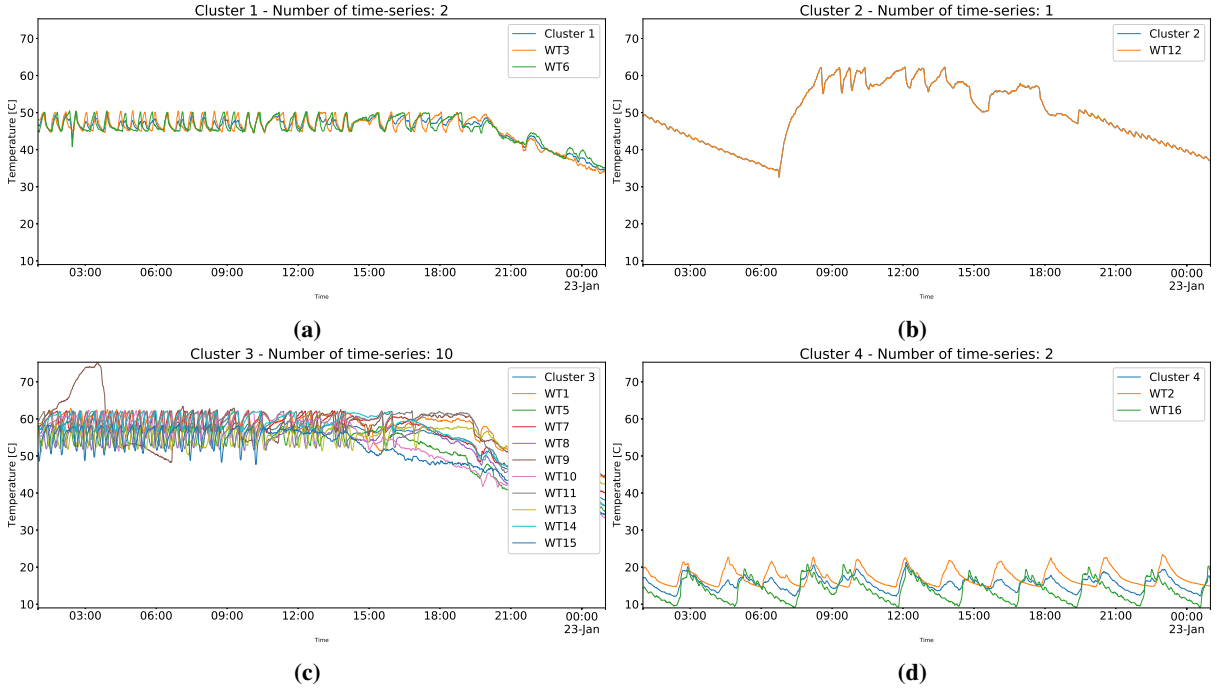


Figure 6.13: Clusters formed when $K = 4$ in the K-means algorithm.

K-means implementation - K equal to 6

K-means is implemented on the data set with a number of cluster K equal to 6. The cluster centres are plotted in Figure 6.14. Similarly, the corresponding time series assigned to each cluster are plotted along with the cluster centre in Figure 6.16. The running time for the K-means algorithm with $K = 6$ was 1.34ms. Now the running time has surpassed the running time for the hierarchical clustering algorithms. This is explained by that the K-means algorithm is dependent on the number of clusters, compared to the hierarchical clustering algorithm, which is not. K-means would surpass the hierarchical clustering algorithm in running time if the number of objects clustered is increased (linear vs quadratic complexity). Running the K-means algorithm several times, with the same parameters, provided different cluster assignment almost every time. An example of the cluster centres from a different run can be seen in Figure 6.15. This will be addressed in more detailed in the following section.

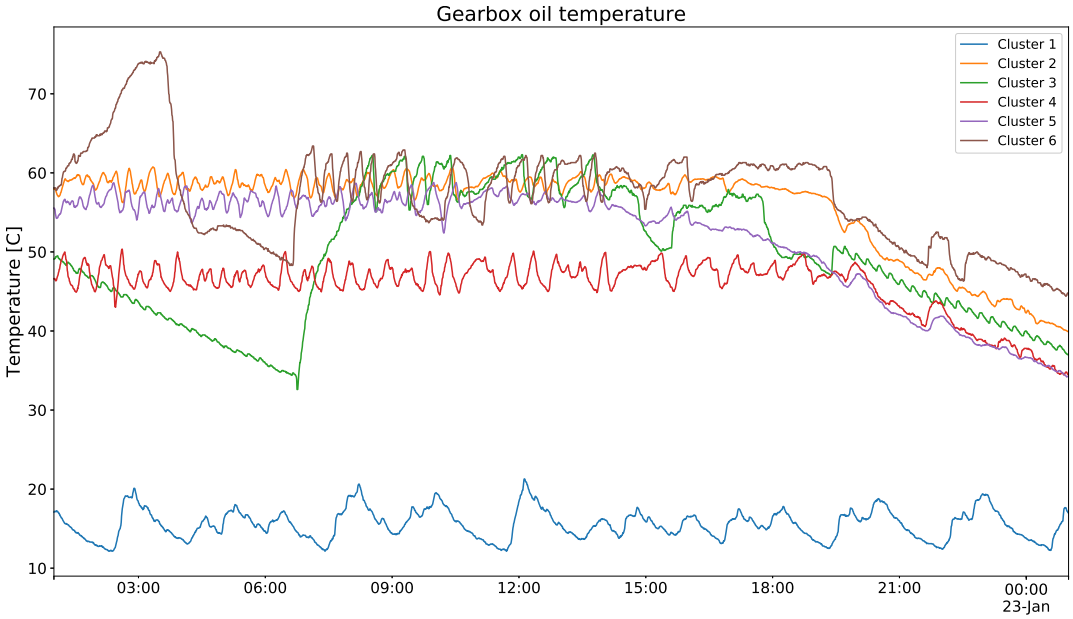


Figure 6.14: Cluster centres obtain by the K-means algorithm where $K = 6$.

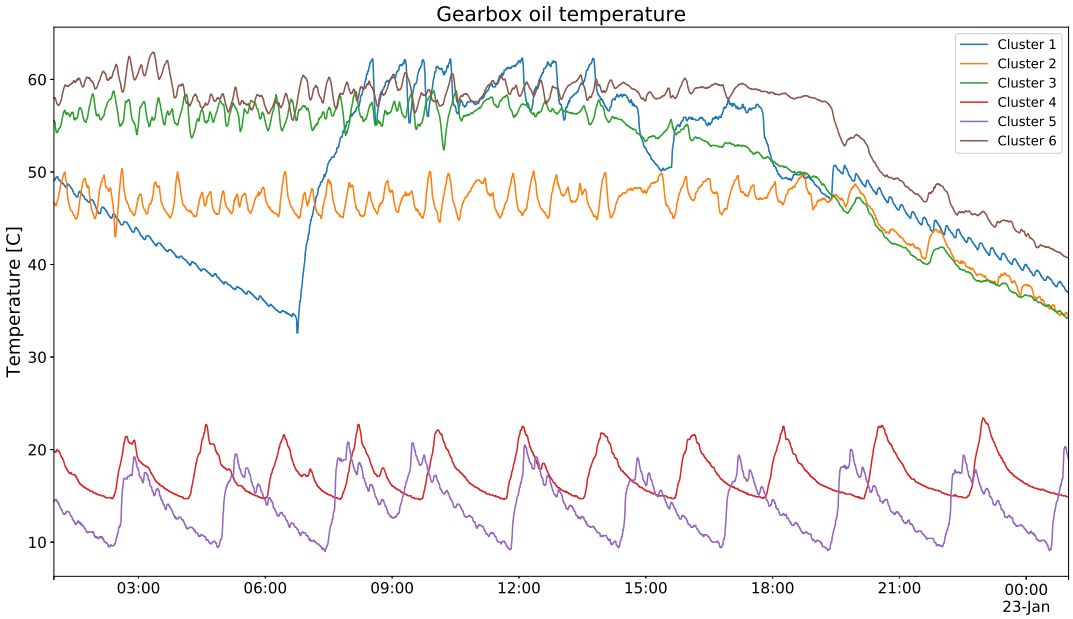


Figure 6.15: Cluster centres obtain by the K-means algorithm where $K = 6$. K-means algorithm ran a second time with the same input as the original run in figure 6.14

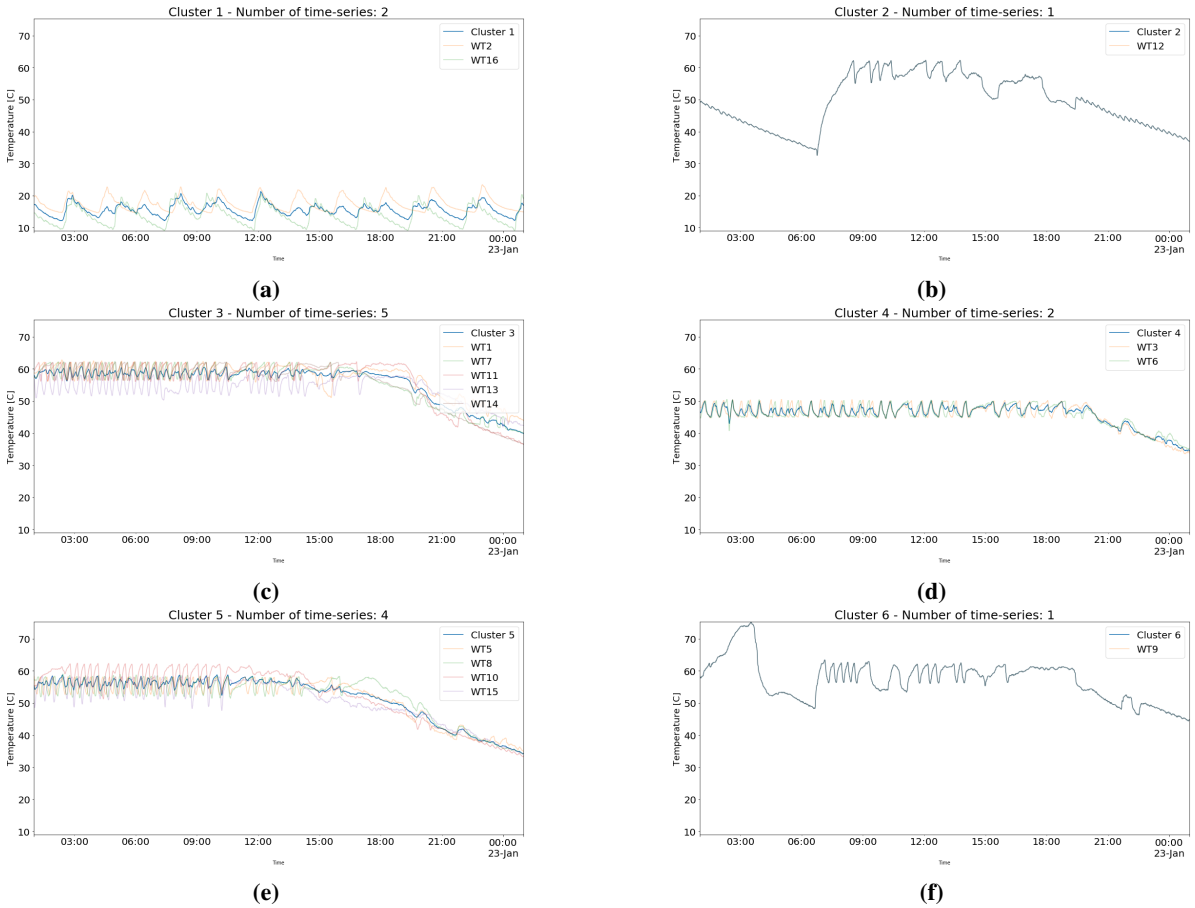


Figure 6.16: Clusters formed when $K = 6$ in the K-means algorithm. Based on the results from Figure 6.14

Summary of similarity in time with K-means

Both the hierarchical clustering algorithm and the K-means algorithm resulted in the same cluster assignments for a number of clusters set to two. However, for a cut of four and six, the assignments of the time series were slightly different for each run. An example was shown for a cut of six which can be seen in Figure 6.14 and 6.15 where the cluster prototypes are slightly different because the time series were assigned differently. For a cut of four, the cluster assignment presented occurred roughly 25% of the time, whereas, for the cut of six, the assignment presented in Figure 6.14 occurred rarely. Each of the configuration for the algorithm was run for $5,000$ iterations, with random seeding, where the best assignment was chosen. The best run was determined by choosing the run which had the smallest sum of all distances to its closest cluster centre. The results for a cut of 4 and 6 are inconsistent and proves hard to verify even with random initiation, increasing the number of iterations, or adjusting the termination criteria for the optimisation problem. In other words, the clustering results from K-means are not deterministic as for hierarchical clustering. Additionally, the benefit of linear time complexity for the K-means algorithm - compared to the quadratic time complexity of the hierarchical clustering algorithm - is not prominent in the current analysis as the number of turbines (or objects) within each data set are only 15. For such small data sets, the benefit from using K-means over hierarchical clustering algorithm is nonexistent. Because of these remarks, clustering the time series with the K-means algorithm will not be used for analysing the other data sets introduced in this section. However, for the scaled-up version where the number of wind turbines could potentially be several hundred turbines, the K-means algorithm should be considered.

6.1.3 Analysing the clustering results from similarity in time

In this section, the results from clustering the 'Univariate_V1' data set in Section 6.1 will be analysed. The corresponding dendrogram with the objective of clustering the time series with respect to similarity in time can be seen in Figure 6.17. The corresponding groupings or cluster assignment are identical for both K-means (for the first run) and hierarchical clustering. The cluster analysis is based on the results from the hierarchical clustering algorithm, rather than K-means algorithm due to the inconsistency of the K-means algorithm. The dendrogram was cut into six partitions and the resulting assignment will be analysed.

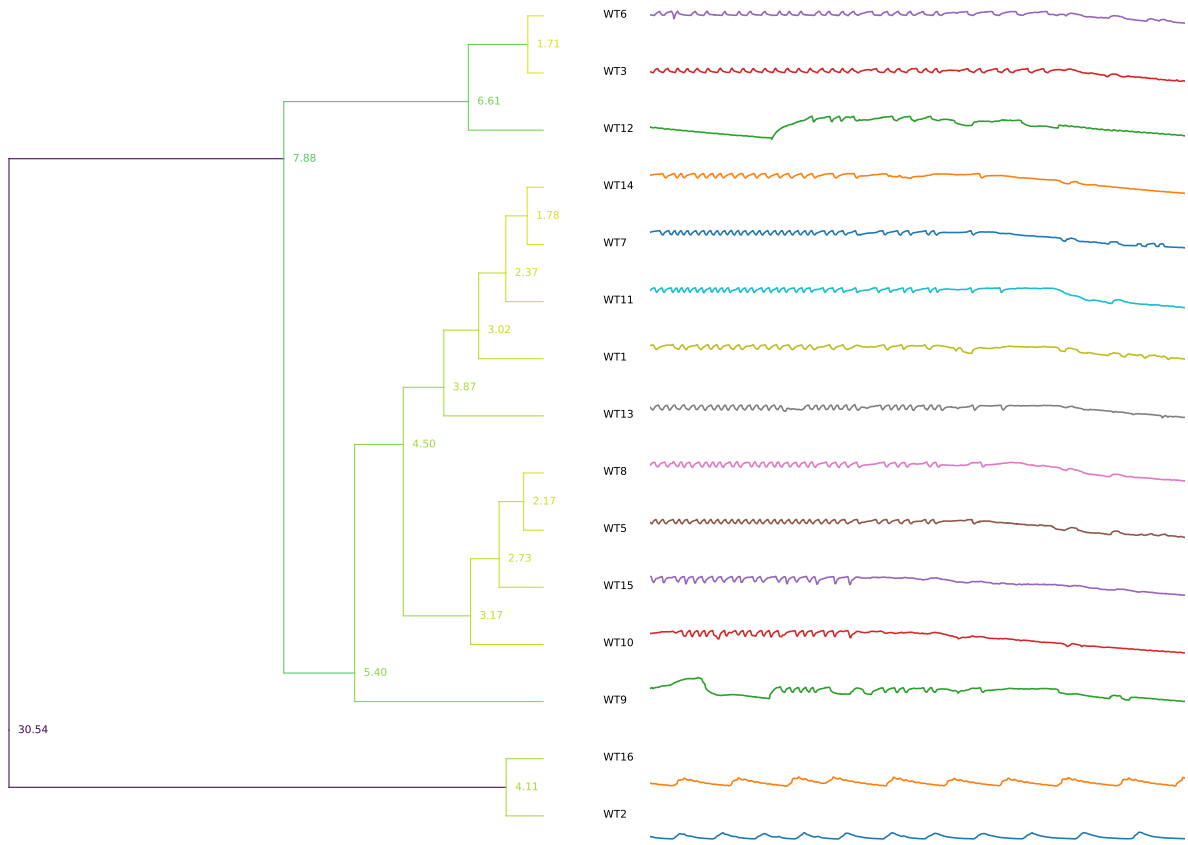


Figure 6.17: Summary of hierarchical clustering with Euclidean as similarity measure and the linkage criterion set to 'average'. The Python script for constructing this figure is presented in Appendix D.3.2.

Summary of the gearbox temperature [degrees Celsius], rotor speed [rpm], generator speed [rpm] and position deviation [degrees] (difference in angle between the direction of the wind and the direction of the nacelle) can be viewed in Table 6.2. The corresponding time series used to extract the content of the table can be seen in Appendix A.1 (Figure A.1.1 to A.1.2). The exception is the rotor speed which is presented later in the current section in Figure 6.18. Additionally, the time series for the wind speed measured at the nacelle for each turbine is presented in Appendix A.1, in Figure A.1.3. Based on these results, an attempt of the physical interpretation will be made based on the cluster assignment. The mean vector of the gearbox temperature or the cluster centre of each cluster are better visualised in Figure 6.14 which shows the cluster centres for the K-means implementation. As can be seen from the plot, cluster centre for group A, B, C, D, E and F are separated by an obvious difference in the overall temperature level (mean temperature) of the different time series involved. In other words, the turbines are separated into six different clusters with respect to their overall temperature level. The two obviously outlying time series associated with turbine 2 and 16 are partitioned into its own separate cluster, regardless of the number of cuts chosen. Larger partitions than two to the dendrogram manages to separate the more abnormal time series (associated with turbine 3, 6, 9 and 12) from the majority. Time series for turbine 3 and 6 are separated by an obvious difference in the intercept - although not that significant compared to the time series associated with turbine 2 and 16. The remaining time series associated with turbine 9 and 12 have similar intercept and trends, but experience very dissimilar behaviour during the first one-third of the interval. The assignment of each of the time

series will now be analysed and interpreted to acquire physical meaning. To help with the interpretation, statistical properties of the aforementioned parameters are provided in Figure 6.2.

Table 6.2: Summary of the properties of each time series within its respective cluster assignment (Group A, B, C, D, E and F) for a cut of 6. Numbers within parentheses (*) refers to the estimated mean of the time series, without including the trend.

Groups	Wind turbine number	Gearbox temp [Celsius]	Rotor speed [rpm]	Generator speed [rpm]	Position deviation [*]
Group A	WT2	Low (20)	Low ($\sim$0)	Low ($\sim$0)	0
	WT16	Low (15)	Low ($\sim$0)	NaN	0
Group B	WT3	Medium-High (48)	High (17)	High (1.2K)	0
	WT6	Medium-High (48)	High (17)	High (1.2K)	0
Group C	WT12	Varying High (60) Medium (30)	Medium (zero in beginning) (17)	Medium (zero in beginning) (1.2K)	0
Group D	WT5	High (55)	High (17)	High (1.2K)	0
	WT8	High (55)	High (17)	High (1.2K)	0
	WT10	High (55)	NaN	High (1.2K)	0
	WT15	High (55)	High (17)	High (1.2K)	0
Group E	WT1	High (60)	High (17)	High (1.2K)	0
	WT7	High (60)	High (17)	High (1.2K)	0
	WT11	High (60)	High (17)	High (1.2K)	0
	WT13	High (60)	High (17) Slightly varying	High (1.2K) Slightly varying	0
	WT14	High (60)	NaN	High (1.2K)	0
Group F	WT9	High (60) Varying	High (17) Varying	High (1.2K) Varying	0

Group A: From the dendrogram, a cut of 2, 4 and 6 would all separate the time series associated with group A into its separate cluster. These time series are two obviously different time series with respect to their intercept and nonexistence of a trend towards the end. The difference between group A and the rest can be visually observed by the clear difference in the mean vector of the time series involved. Time series from turbine 2 and 16 have a significant separation between the rest of the time series in the data set, suggesting a significant difference in those two turbines. It can also be observed that the two outliers have a significantly lower frequency of the oscillations and have a constant mean throughout the whole interval. The first impression of this separation is that all the wind turbines, except turbine 2 and 16, are actively rotating. This explains the lower temperature and the nonexistence of the drop toward the end of the interval. This assumption can be verified by viewing the rotor speed for each individual turbine (see Figure 6.18). The rotor speed of wind turbine 10 and 14 was not available as the database contains a lot of areas where the sensors malfunctions and does not produce any sensor data. However, the generator speed measurement was available and reflects the exact same property. It can be observed that the rotor/generator speed for both wind turbine 2 and 16 are both zero (or close to zero), where the rest of the turbines are rotating at a significantly higher speed. The assumption that the turbines are not rotating is confirmed and the clustering manages efficiently to separates these two time series from the rest. It can also be noted that doing the same clustering procedure - only with a multivariate case where the rotor speed is included - these would most likely result in the same cluster assignment as for the univariate case. However, the missing rotor speed values of wind turbine 10 and 14 have to be addressed prior to this; preferably substituted by the generator speed data.

Group B: The mean of the gearbox temperature within these time series are significantly lower than that of the remaining wind turbines in group C, D, E, and F. The first assumption for why these turbines have lower temperatures than the rest is that they are rotating more slowly. However, this was not the case as they have identical generator speeds as the others with slightly larger gearbox temperatures. Another hypothesis is that the lowered average temperature levels might indicate that the components of the gearbox are less worn out or better lubricated than the others. This is not transparent in the rotor speed data (or any of the other parameters) and the assumptions prove hard to verify. The last hypothesis is that the wind turbines are from a different model and have

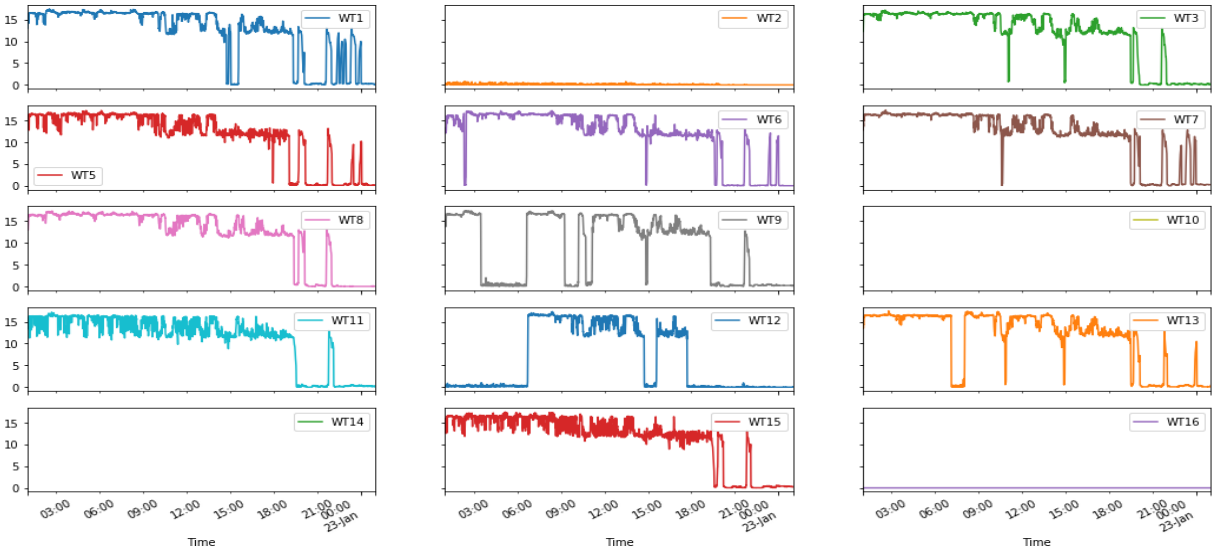


Figure 6.18: Plot of the rotor speed [rpm] during the same period as the extracted data in Section 5.1.1

a higher/lower maximum power consumption. By reviewing the maximum power - which will be different if the turbine model were different from the rest - it was found that the turbines of group B have similar maximum power outputs as the majority and therefore of the same turbine model (this can be seen in Figure A.3.2 in Appendix A.3).

Group C: This group contains only one wind turbine and has one of the most drastically changing gearbox temperatures in the data set. Its varying temperature in the gearbox is justified by that it experience similar behaviour in rotor/generator speed. However, the wind speed does not experience similar behaviour. The first assumption for the turbine is that it was subjected to high wind conditions and therefore had its brakes engaged. But, from reviewing the wind speed in Figure A.1.3, this was not the case. Nonetheless, the algorithm manages to separate it from the rest as its experience quite different behaviour in the first quarter of the interval, compared to the rest.

Group D: Turbines within this group are characterised by a high gearbox temperature, high generator/rotor speed, and all are directed against the wind. The turbines in this group experience similar behaviour as those of group B, only the gearbox temperature is slightly elevated. Same interpretation as for group B holds, only reversed: The elevated gearbox temperature might indicate that the components are more worn out or poorer lubricated than group B.

Group E: Turbines within this group are characterised by a high gearbox temperature, high generator/rotor speed, and are all directed against the wind. The turbines within this group - compared to those of group D - have slightly elevated average gearbox temperature levels, despite having the same rotational speed. Same interpretations as for the turbines in group D applies for these: Turbines of group E might be more worn out or poorer lubricated than those of group D, and even more so, compared to the turbines in group B.

Group F: Contains only one time series. The properties for this turbine are similar to the turbines associated with group C. The turbine experience varying gearbox temperatures which can be explained by an equally varying generator/rotor speed. However, the sudden halt in the rotation is not explained by either the wind speed or the positional derivation. Nonetheless, the algorithm manages to separate it from the rest as its experience quite different behaviour in the first quarter of the interval, compared to the rest.

6.1.4 Similarity in shape - Hierarchical clustering of the normalised data set

The objective is now to cluster the time series with similarity in shape and not the absolute difference and offset. Therefore, the cluster analysis is performed on the normalised data set. The trend is not removed prior to the normalisation as its important to the overall shape and we wish to capture this behaviour in the cluster analysis. The time series which will be clustered with regards to the objective of similarity in shape can be viewed in Figure 5.6. For similarity in shape, the dynamic time warping (DTW) distance (3.7) is used instead of the Euclidean distance. As DTW is not supported as a similarity measure for the hierarchical clustering algorithm, the DTW distance between each time series are calculated first and then stored in a dissimilarity matrix (DM) (as in Equation (3.3)). The condensed form of the dissimilarity matrix is furthermore fed to the hierarchical clustering algorithm. The DTW distance can be calculated by numerous libraries in Python. A comparison between the libraries is made in Appendix C.2. The library chosen for calculation of the condensed distance matrix between all time series was the library called *dtaidistance* with the function call '*dtw.distance_matrix*'. The DTW algorithm is implemented with a global constraint called the *Sakeo-Chiba band* restriction. The width of the band is set to 10% of the length of the time series in the data set. Justification for the choice of the width and need for the global restriction (Sakeo-Chiba band) has been made in Appendix C.1. In brief, an accurate DTW calculation is dependent on the implementation of the global constraint and how narrow the width of the band is. Furthermore, DTW has much higher time complexity then Euclidean distance and the calculation of the condensed distance matrix took 413 seconds compared to Euclidean distance which took only 1.32 ms. In order to find the appropriate linkage criterion and the optimal number of clusters, the internal indexes are analysed. The best method for hierarchical clustering will be found by comparing the *Cophenetic correlation coefficient* and the number of cuts to the dendrogram will be determined by viewing the internal performance indexes: *Silhouette index* and *MSSSE*. More information about these specific indexes can be found in Chapter 3. The implementation of the internal indexes as well as the implementation of the hierarchical clustering algorithm with DTW as the similarity measure can be seen in Appendix D.1. The summary plot and table for the internal indexes of cutting each dendrogram into *k* partitions - each with different linkage criterion - can be seen in Figure 6.19 and Table 6.3. The cophenetic correlation coefficient can be seen in the upper right corner in the aforementioned figure (Figure 6.19).

Table 6.3: Summary table with the linkage criteria: single, complete, average and ward. The data set which was clustered is 'Univariate_V1' with the objective of similarity in shape. The silhouette indexes and MSSSE values for a cut K - from 2 to 13 - is presented.

Method	Internal index	Number of cuts to the dendrogram											
		K = 2	K = 3	K = 4	K = 5	K = 6	K = 7	K = 8	K = 9	K = 10	K = 11	K = 12	K = 13
Single	Silhouette	0.66	0.63	0.61	0.56	0.06	0.05	0.08	0.16	0.11	0.14	0.13	0.07
	MSSSE	3.75	2.77	2.04	1.58	1.36	1.14	0.93	0.64	0.49	0.36	0.24	0.16
Complete	Silhouette	0.66	0.63	0.61	0.56	0.22	0.23	0.23	0.20	0.16	0.14	0.13	0.07
	MSSSE	3.75	2.77	2.04	1.58	1.22	1.02	0.80	0.65	0.51	0.36	0.24	0.16
Average	Silhouette	0.66	0.63	0.61	**0.56**	0.22	0.23	0.23	0.20	0.17	0.14	0.13	0.07
	MSSSE	3.75	2.77	2.04	**1.58**	1.26	1.02	0.80	0.65	0.51	0.36	0.24	0.16
Ward	Silhouette	0.67	0.63	0.61	0.56	0.22	0.23	0.23	0.20	0.16	0.14	0.13	0.07
	MSSSE	3.73	2.77	2.04	1.58	1.26	1.02	0.80	0.65	0.51	0.36	0.24	0.16

First of all, reviewing the cophenetic correlation coefficient in Figure 6.19, it can be seen that the largest coefficient is observed for the dendrogram constructed with *average* as the linkage criterion. However, the others still have relatively high values which are close to 1, indicating that all the dendrograms represent a high-quality solution. The dendrogram built with linkage criterion set to *average* will be cut and analysed. Secondly, reviewing the summary table and the corresponding summary plot, it can be observed that the silhouette index is largest for a cut of two for the average dendrogram. Actually, this can be observed for all dendrograms, as the cophenetic correlation coefficient are all very large and quite similar. Only a small reduction in the silhouette index value can be observed by going from a cut of two to a cut of five; where afterwards the silhouette index suddenly plunges. The silhouette index does not decrease exponentially for a lower number of cuts as it did with the objective of similarity in time. This is because the intercept is somewhat ignored and the time series are more similar in terms of their overall shape, rather than their similarity in time. Furthermore, it can be observed that the corresponding MSSSE value decreases exponentially along with deeper cuts to the dendrogram. The values of the MSSSE values can be observed to be significantly smaller during this analysis, than in the cluster analysis with similarity in time as the objective. The reasoning for this is because the intercept was included in the analysis of similarity in time and the objective is a much stricter measure of similarity, compared to the objective of similarity in shape. The silhouette

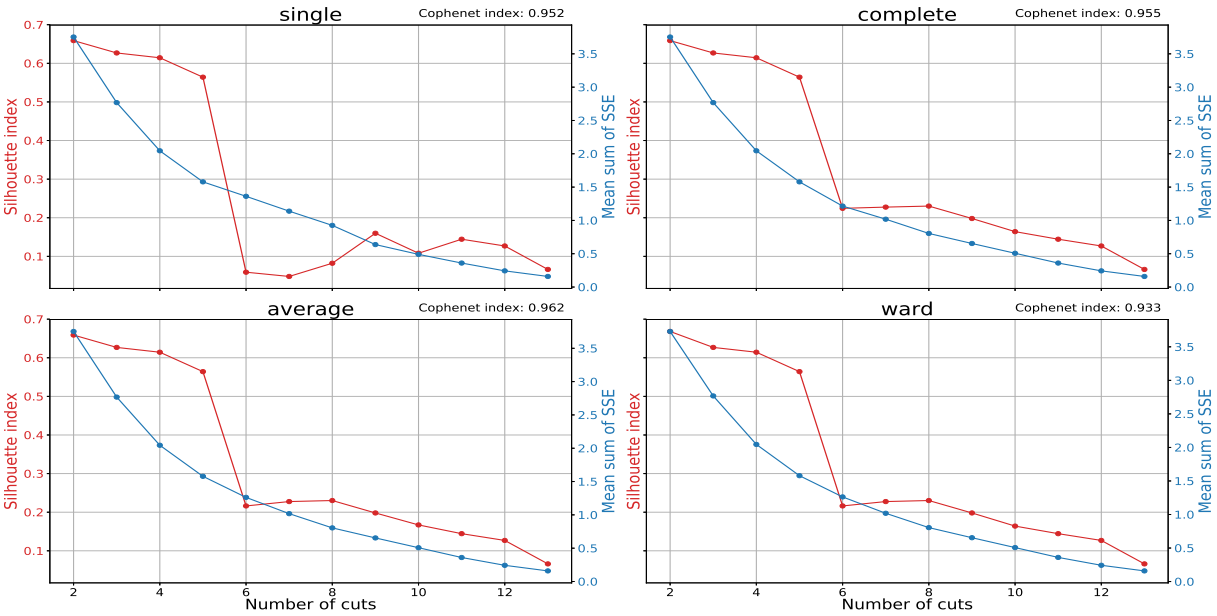


Figure 6.19: Summary plots with the linkage criteria: single, complete, average and ward. The data set which was clustered is 'Univariate_V1' with the objective of similarity in shape. The silhouette indexes and MSSSE values for a cut K - from 2 to 13 - is presented along with the cophenetic correlation coefficient in the upper right corner.

index and the MSSSE value - during this analysis - indicates that a cut of five is the optimal number of cuts to the dendrogram. For a cut of five, a relatively large silhouette index and a significant reduction of the MSSSE value from previous cuts can be observed. The best compromise between separation and similarity is, therefore, a cut of five to the dendrogram built with average as the linkage criterion. Deeper cuts to the dendrogram significantly reduce the silhouette index value and an insignificant reduction in the MSSSE value. The optimal configuration for the current data set is chosen to be *average* as the linkage criterion and cutting the dendrogram such that it constructs five partitions. Deeper insight will be obtained by reviewing the corresponding dendrogram along with the assigned clusters. The resulting dendrogram - with *average* as the linkage criterion - can be viewed in Figure 6.20. Viewing the dendrogram and the corresponding time series associated with each partition, a deeper cut of five to the dendrogram would actually result in a significant reduction in the average distance between each cluster. This strengthens the choice of the optimal number of cuts based on the internal indexes. Therefore, the physical interpretation of a cut of five will be examined in greater detail in the consecutive section.

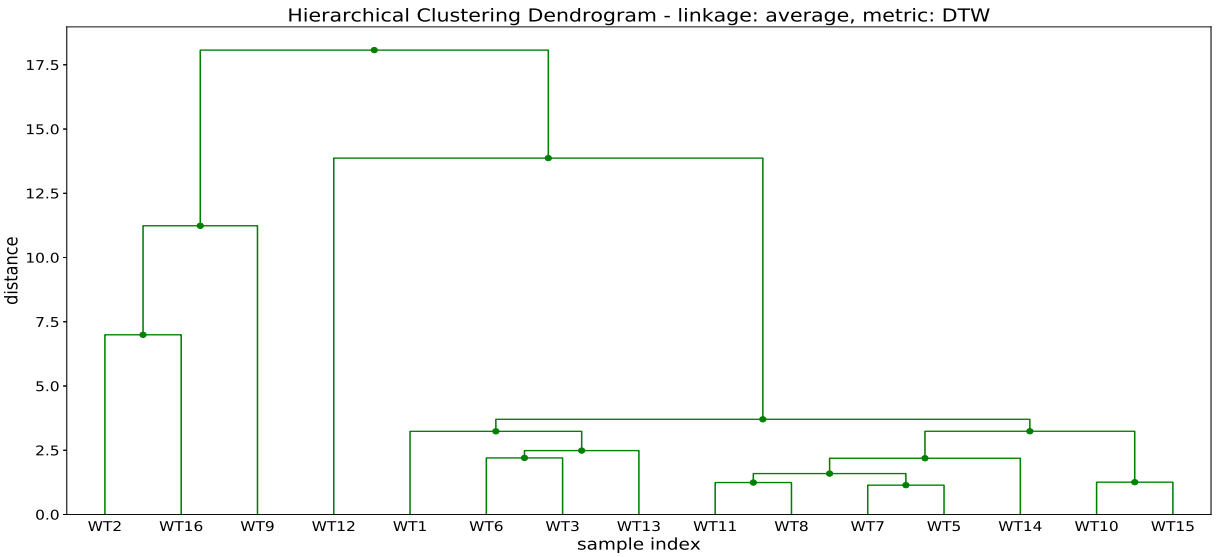


Figure 6.20: Hierarchical clustering with DTW as similarity measures.

6.1.5 Analysing the clustering results from similarity in shape

The corresponding dendrogram with the objective of clustering the time series with respect to similarity in shape can be seen in Figure 6.21. The absolute difference and the offset will be ignored to a certain degree, as normalising the data set will remove the offset and scale the data between 0 and 1. Clustering with the objective of similarity in shape manages to separate turbines with different behaviour pattern and manages to group time series subjected to shifts in time. For instance, a cosine wave and a sinus wave are pretty similar when using the DTW distance, but are drastically different when using Euclidean distance. In other words, cosine and sinus wave are similar in shape but not in time.

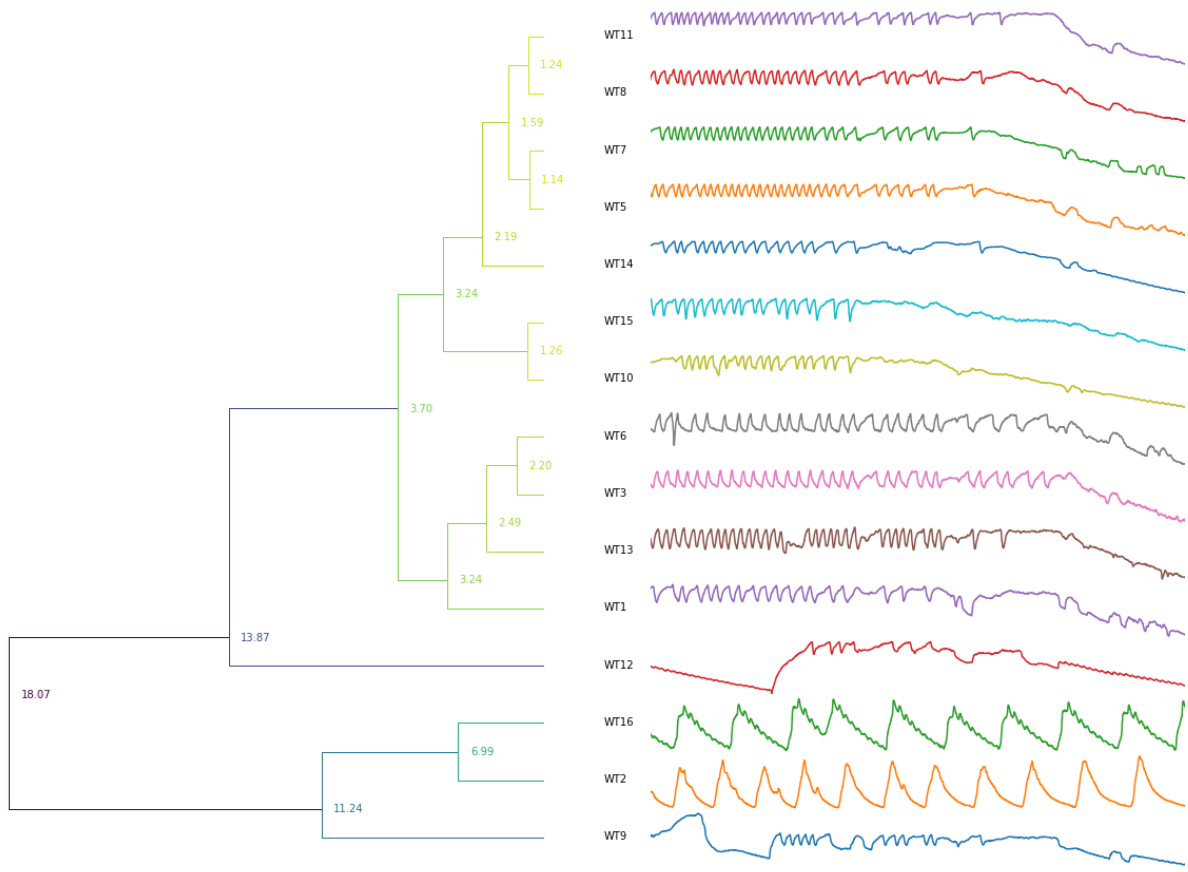


Figure 6.21: Summary of hierarchical clustering with DTW as similarity measure and the linkage criterion set to 'average'

Initially, we can observe that using DTW as the distance measure to the clustering algorithm manages to group the most similar time series in shape and separate the most dissimilar ones. A cut of five separates the four most abnormal time series - time series corresponding to wind turbine 2, 9, 12 and 16 - into its own separate clusters (from now on called group A, B, C and D, respectively). The last cluster contains the majority of the time series and is hereby called group E. Note that the size of the branches is not scaled properly, which is one of the limitations of the *dtaidistance* library. The number in the dendrogram should be observed instead; the numbers indicate the average distance when the clusters are merged together. For example, turbine 16 and 2 are merged together at a distance of 6.99 but shows a smaller distance visually compared group E, which are merged together at a distance of 3.70. Oscillations can be observed in all time series for all clusters. These oscillations are the result of the cooling pumps enabling and disabling when hitting their corresponding threshold values; the pump actives when hitting the upper threshold value and deactivates when it reaches the lower threshold value. Analysis of the physical interpretation of the clustering assignment will be performed in the consecutive paragraphs.

Group A: Contains only the time series associated with wind turbine 2. The time series is characterised by medium frequency oscillations and constant mean throughout its entire length. The cooling pump enables when reaching the maximum values observed in the plot and cools it down to its minimum value. The time series does not experience a decreasing trend towards the end, which the wind speed has. A reduction in the wind should cause

a reduction in the rotation speed for the turbine, which furthermore, should lower the gearbox temperature for the associated wind turbine. Since it does not follow the same trend as the local wind, it is a strong indication that the turbines operate unaffected by the wind. The only possible explanation for this is that the turbine is not rotating. This can be verified by reviewing the generator speed from the previous section (cluster analysis of similarity in time).

Group B: Contains only the time series associated with wind turbine 9. The time series is characterised by quite abnormal behaviour in the initial 5 hours (02:00 - 07:00) of the interval, but has similar behaviour as the majority, after this interval. As with the objective of similarity in time, the explanation for this is that the turbine experiences a period at the beginning where the generator/rotor is not rotating. The physical interpretation of this is not transparent in the wind speed or direction. One would expect that the wind during the initial period experiencing a raise and then a decreased, such that the temperature in the gearbox increases and then decreases, but this was not the case. Regardless, the algorithm effectively manages to separate it from the rest. From viewing the dendrogram, turbine 9 can be observed to be more similar to turbine 2 and 16, than it is to any of the others. The explanation for this is that the time series are normalised, and the elevated values in the initial hours of the interval are used for normalising the time series. In other words, maximum observable values are larger than the majority and the time series are then normalised appropriately. This is expected as we want to compare the shape of the time series, rather than its variance. If the variance was of greater interest, standardisation of the time series would solve this issue, making it more similar to the majority.

Group C: Contains only the time series associated with wind turbine 12. The time series is characterised by quite a abnormal behaviour during the initial 5 hours (02:00 - 07:00) of the interval, but pretty similar behaviour as the majority, after the initial 5 hours. As with the objective of similarity in time, the explanation for this is that the turbine experience period at the beginning where the generator/rotor is not rotating. The same physical interpretation of turbine 9 in group B holds for this turbine.

Group D: Contains only the time series associated with wind turbine 16. The time series is characterised by oscillations with a frequency close to half of the frequency of the time series in group A - the oscillations also has lower amplitude than the time series in group A. This might indicate that the settings for the cooling pump are different, especially with regards to the threshold values where the pump enables and disables. This explains the slightly elevated temperature levels, but proves hard to verify as such information is not available. Other than that, the same interpretation as for turbine in group A holds: The turbine is not rotating as it does not follow the trend of its local wind conditions.

Group E: Contains the rest of the turbines. These turbines are characterised by relatively high-frequency oscillation, roughly the same amplitude and a decreasing trend towards the end. These turbines are also highly correlated to the behaviour of the local wind conditions. The physical interpretation for these turbines is that they are actively rotating and could be further interpreted as turbines which are operating under normal conditions. This is verified by viewing the generator speed for all turbines.

6.2 Clustering of 'Univariate_V2'

In this section, the 'Univariate_V2' data set will be clustered and the clustering results will be analysed as done in the previous section. First of all, the time series in Figure 5.8 are visually inspected. At first glance, there are four to five different groups of time series with respect to its temperature levels where the majority of the time series are in the group which has roughly a mean of 57 degrees Celsius. Majorly different time series in shape can also be observed in the plot. The objectives for clustering of the 'Univariate_V2' data set - which are of interest - are both similarity in time and similarity in shape. Similar procedure as in the previous section (Section 6.1) will be repeated in order to enhance the physical interpretation of its results. As K-means did not provide consistent results in the previous section, the analysis will not be based on the results of the K-means algorithm but rather the hierarchical clustering algorithm. First of all, hierarchical clustering with Euclidean distance as the similarity measure will be implemented on the scaled data set in Section 6.2.1 followed by the analysis in Section 6.2.2. This will solve the clustering problem with respect to the objective of similarity in time. Then, the same clustering algorithm will be used in conjunction with DTW as the distance function on the normalised data set to find similar time series with respect to their shape. The resulting cluster assignment will be presented in Section 6.2.3 followed by an analysis in Section 6.2.4 in order to obtain physical insight into the partition. The Python implementation is the same as for the previous data set, only some minor changes to the script in Appendix D.1.5. For running the cluster analysis on the 'Univariate_V2', comment out line number 11 to 29 and uncomment line 33 to 51, and follow the procedure outlined in Appendix D.1.

6.2.1 Similarity in time - Hierarchical clustering on the scaled data set

The objective is now to cluster the time series with regards to the objective of similarity in time. In this analysis, the scaled data set will be analysed with the hierarchical clustering algorithm. The Python implementation of the hierarchical clustering algorithm along with the implementation of the internal indexes can be seen in Appendix D. Similar to the clustering of the 'Univariate_V1' data set in Section 6.1, the dendrograms will be compared to one another by comparing their *cophenetic correlation coefficient* (3.10). After the best dendrogram has been found, the optimal number of clusters (or cuts to the dendrogram) will be found by comparing the *Silhouette index* (3.14) and the *MSSSE* (3.16) value for each cut, K. The summary plot and table for the internal indexes of cutting each dendrogram into *k* partitions - each with different linkage criterion - can be seen in Figure 6.22 and Table 6.4.

Table 6.4: Summary table with the linkage criteria: single, complete, average and ward. The data set which was clustered is 'Univariate_V2' with the objective of similarity in time. The silhouette indexes and MSSSE values for a cut K - from 2 to 13 - is presented.

Method	Internal index	Number of cuts to the dendrogram											
		K = 2	K = 3	K = 4	K = 5	K = 6	K = 7	K = 8	K = 9	K = 10	K = 11	K = 12	K = 13
Single	Silhouette	0.73	0.57	0.51	**0.59**	0.50	0.36	0.26	0.14	0.14	0.14	0.13	0.13
	MSSSE	55.67	29.88	17.16	**5.29**	4.04	1.75	1.46	1.20	0.94	0.69	0.46	0.21
Complete	Silhouette	0.73	0.62	0.59	0.59	0.50	0.36	0.19	0.19	0.14	0.14	0.14	0.13
	MSSSE	55.67	18.05	9.91	5.29	4.04	1.75	1.48	1.20	0.94	0.69	0.44	0.21
Average	Silhouette	0.70	0.57	0.59	0.59	0.50	0.36	0.26	0.19	0.14	0.14	0.14	0.13
	MSSSE	46.51	29.88	9.91	5.29	4.04	1.75	1.46	1.20	0.94	0.69	0.44	0.21
Ward	Silhouette	0.73	0.62	0.59	0.59	0.45	0.36	0.26	0.14	0.14	0.14	0.14	0.13
	MSSSE	55.67	18.05	9.91	5.29	2.99	1.75	1.46	1.20	0.94	0.69	0.44	0.21

First of all, reviewing the cophenetic correlation coefficient in Figure 6.22, it can be seen that all dendrograms experience similarly high values, all above 0.899. Both single and ward dendrograms have cophenetic correlation coefficients which are greater than 0.925. These cophenetic correlation coefficients are close to 1 for all dendrogram (i.e. greater than 0.75); values close to 1 indicate that all the dendrograms represent a high-quality solution. The dendrogram with the highest cophenetic index is the dendrogram constructed with linkage criterion as *single*. Therefore, the dendrogram built with linkage criterion as *single* will be cut and its assignment will be analysed. Secondly, reviewing the summary table and the corresponding summary plot, it can be observed that the silhouette index is largest for a cut of two ($SI = 0.73$). This indicates that a cut of two provides two clusters with a good combination of high intra-cluster similarity and large separating between them (i.e. the time series are well matched to its own cluster and poorly matched to any other clusters). Observing the MSSSE value, the high silhouette index is mainly caused by a significant separation as the MSSSE value is still relatively large. This indicates that

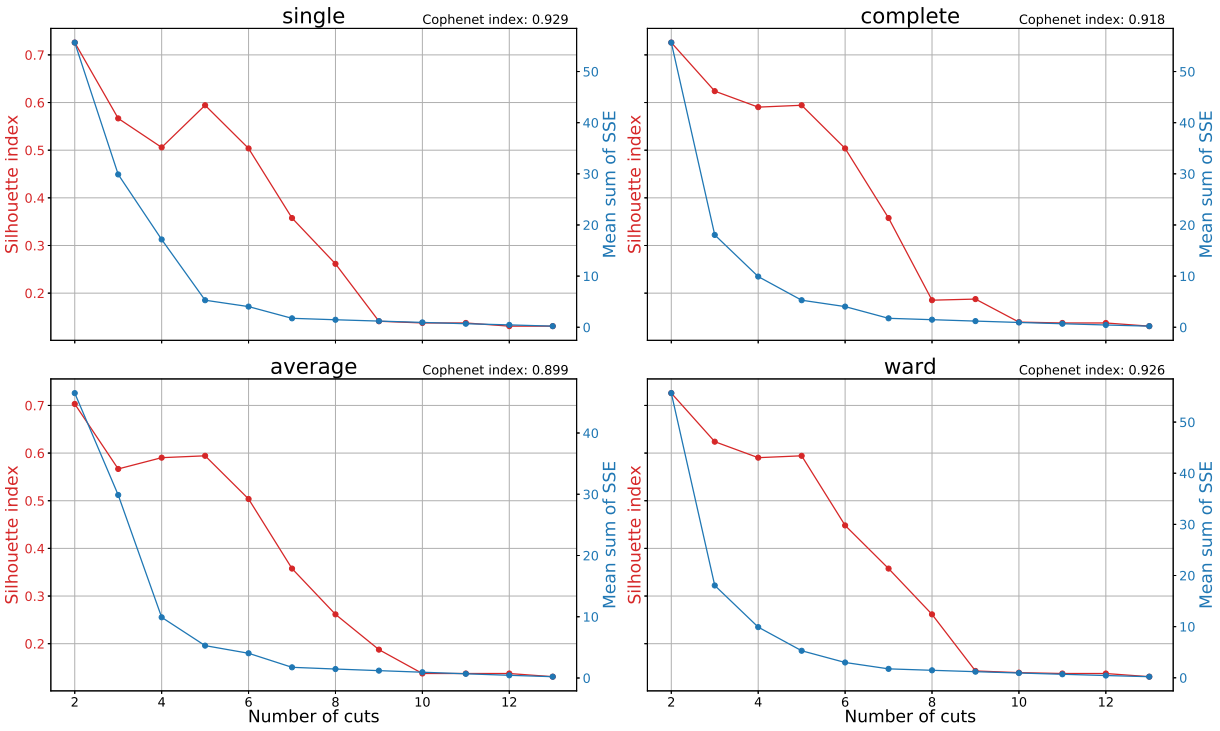


Figure 6.22: Summary plots with the linkage criteria: single, complete, average and ward. The data set which was clustered is 'Univariate_V2' with the objective of similarity in time. The silhouette indexes and MSSSE values for a cut K - from 2 to 13 - is presented along with the cophenetic correlation coefficient in the upper right corner for each dendrogram.

a deeper cut to the dendrogram might be more suitable. The second biggest silhouette index value of 0.59 occurs when the dendrogram is cut into five partitions. The corresponding MSSSE value decreases exponentially during a cut of 2 to a cut of 5; it decreases from 55.67 to 5.29 which is reduction by a factor of more than 10. After a cut of five, the corresponding MSSSE value starts to flatten and no significant reduction is observed in the internal index. The best compromise between a small MSSSE value and a high silhouette index value is achieved for a cut of five to the dendrogram. A cut of five to the dendrogram with *single* as linkage criterion is therefore chosen for further analysis. The corresponding dendrogram - with *single* as linkage criterion - can be seen in Figure 6.23. The resulting assignment for a cut of two and a cut of five will further be visualised in the consecutive sections; this is mainly for illustrative purposes.

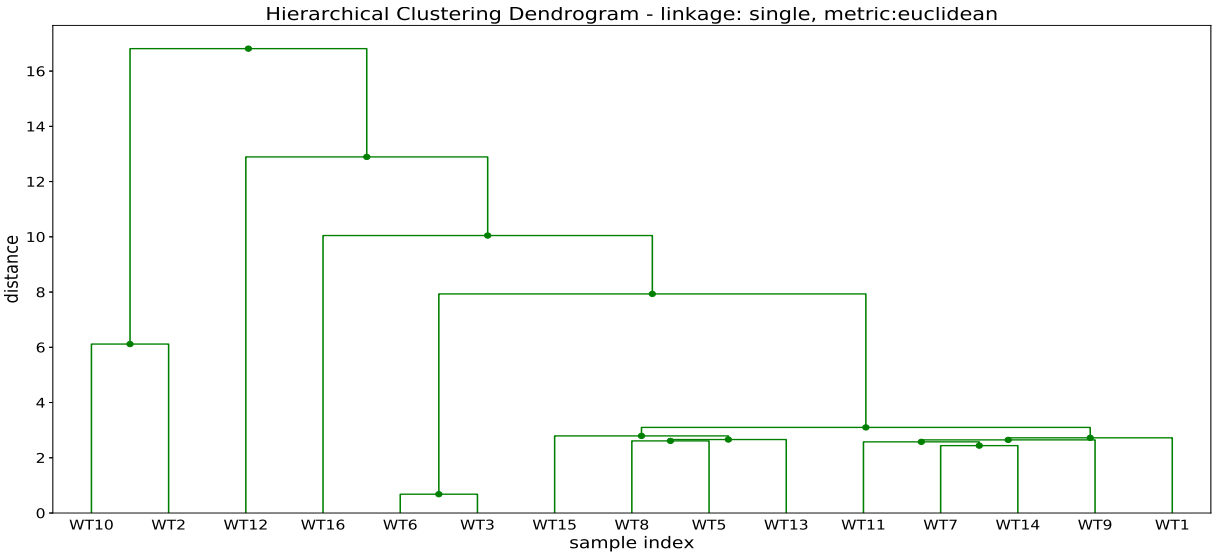


Figure 6.23: Dendrogram with single as the linkage criterion and distance as the Euclidean distance implemented on 'Univariate_V2' data set. The entire clustering process took 1.31ms which is basically the same as for the previous data set.

Cutting the dendrogram to form 2 clusters

First of all, the dendrogram is cut to form two clusters. The resulting cut in the dendrogram is shown in Figure 6.24 along with colour coding for the cluster assignment (green and red). The shortest distance between the two clusters is given by the distance of the uppermost horizontal line, which is roughly 14. The large separation between the clusters is responsible for the large silhouette index observed in the summary table and summary plot. However, the corresponding MSSSE value of 55.67 is relatively large, indicating a large within-cluster variance. Deeper cut can be observed to reduce the MSSSE value quite substantially.

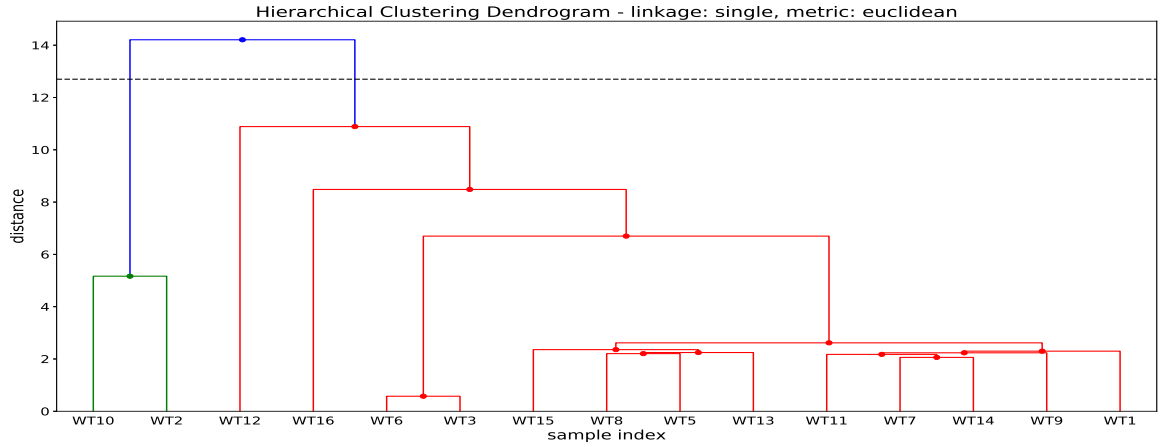


Figure 6.24: Cutting the dendrogram - with single as linkage criterion and distance as Euclidean distance - into 2 partitions.

Cutting the dendrogram to form 5 clusters

Now the dendrogram is cut to form five clusters. The resulting cut in the dendrogram are shown in Figure 6.25 along with colour coding for the cluster assignment (green, dark blue(1), dark blue(2), red, turquoise/light blue). Note that WT12 and WT16 are both dark blue, but is not in the same cluster as they are in its own individual cluster containing only itself. The silhouette index value for a cut of five is still pretty high, with a value of 0.59. However, now the corresponding MSSSE value is reduced by a factor of more than 10 ($MSSSE = 5.29$). A deeper cut in the dendrogram would result in a significant reduction in the silhouette index value and only a small reduction in the MSSSE. A cut of five to the dendrogram with linkage criterion as *single* is the optimal configuration which has the best compromise between separation and within-cluster similarity. A cut of six would separate the green

cluster into two and would still result in a significant separation between the two. However, the corresponding silhouette index has reduced quite a bit with a relatively small reduction in the MSSSE value. Arguably, a cut of six could also be of interest, but will not be performed as the internal indexes strongly indicate a cut of five to the dendrogram.

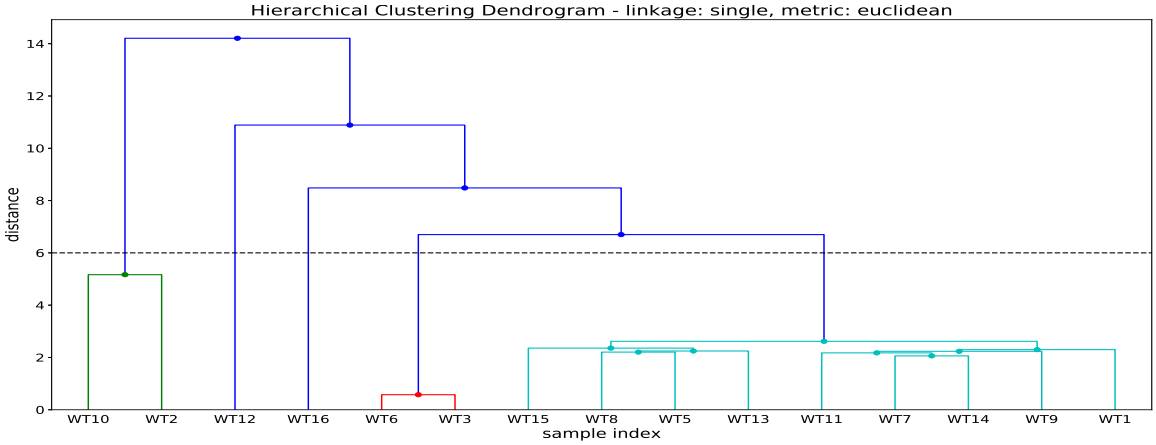


Figure 6.25: Cutting the dendrogram - with single as linkage criterion and distance as Euclidean distance - into 5 partitions.

6.2.2 Analysing the clustering results from similarity in time

In this section, the results from clustering the 'Univariate_V2' data set in Section 6.2 will be analysed. The corresponding dendrogram with the objective of clustering the time series with respect to similarity in time can be seen in Figure 6.26.

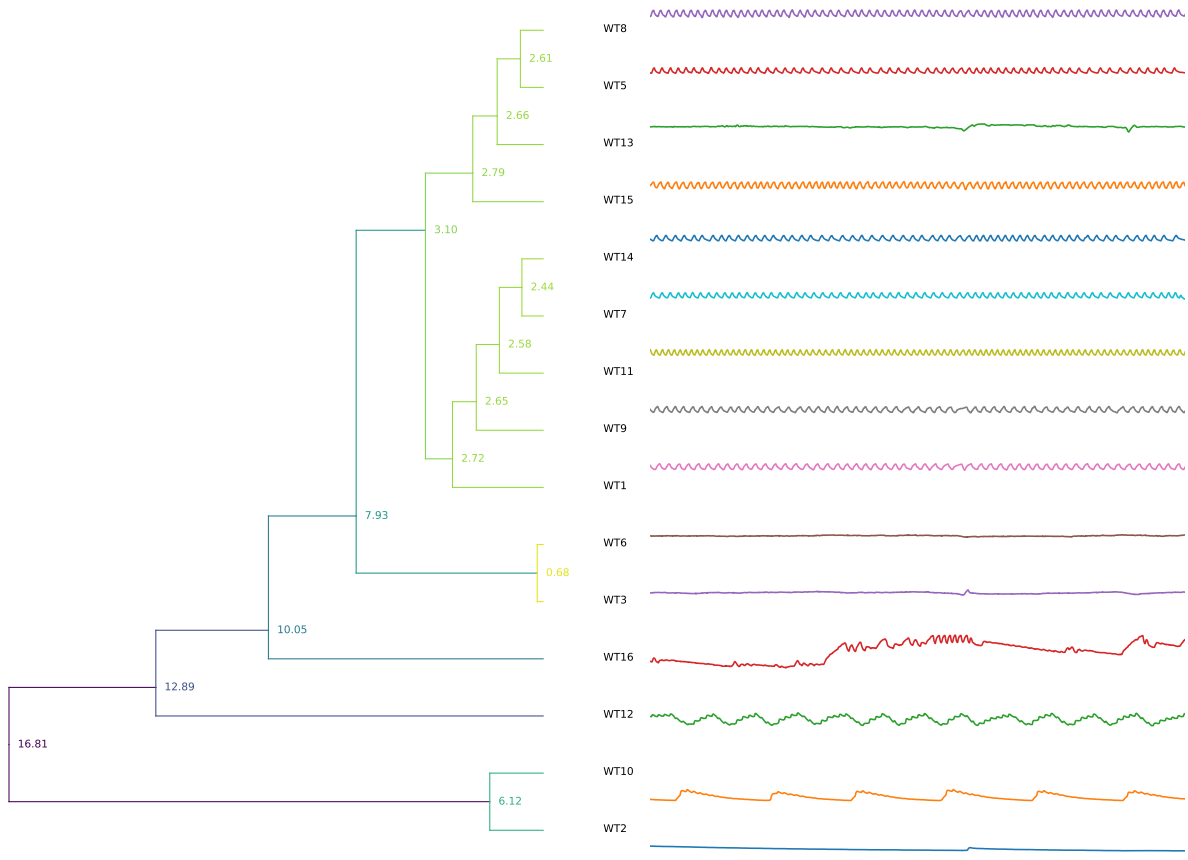


Figure 6.26: Summary of hierarchical clustering with Euclidean distance as similarity measure and linkage criterion set to 'single'. Note that the scaling of the tree is wrong; refer to the distances (i.e. the numbers in the figure) at each merging instead.

Summary of the gearbox temperature [degrees Celsius], rotor speed [rpm], generator speed [rpm] and position deviation [degrees] (difference in angle between the direction of the wind and the direction of the nacelle) can be viewed in Table 6.5. The gearbox temperature corresponds to the time series presented in Section 5.1.2. The other corresponding time series used to extract the content of the table can be seen in Appendix A.2, where the data set called 'Multivariate_V1' is presented. The univariate time series corresponding to the rotor speed, generator speed and position deviation are presented in Figure A.2.5, A.2.6 and A.2.7, respectively. The time series for the wind speed measured at the nacelle for each turbine is also presented within Appendix in Figure A.2.8 followed by descriptive statistics for the wind speed in Table A.2.1. Furthermore, the summary table, in this section in Table 6.5, is constructed for a cut of five, dividing the wind turbines into five groups: Group A, B, C, D and E. The assignment of individual time series can be seen in the corresponding table. Based on these results, physical interpretation can be made from the cluster assignment for a cut of five. Initially, the hierarchical clustering algorithm efficiently manages to separate the time series experiencing different temperature levels. The separated time series can also be observed to be quite different in terms of shape. Based on that, the clustering results from for clustering the data set with the objective of similarity in shape (presented in the next section) can be assumed to be similar to the clustering results from this section; the time series are both similar in time and shape. Now, the physical interpretation of this partition will be conducted in the following paragraphs.

Group A: The wind turbines in this group have in common that they have a relatively low average gearbox temperature compared to the others. The initial assumption is that these wind turbines are rotating slower than the rest or not rotating at all. Reviewing the rotor speed it is clear that wind turbine 2 has significantly lower rotor

Table 6.5: Summary of the properties of each time series within its respective cluster assignment (Group A, B, C, D and E) for a cut of five. Numbers within parentheses (*) refers to the estimated mean of the time series, without including the trend.

Groups	Wind turbine number	Gearbox temp [Celsius]	Rotor speed [rpm]	Generator speed [rpm]	Position deviation [*]
Group A	WT2	Low (20)	Low ($\sim$0)	Low ($\sim$0)	0
	WT10	Low (15)	NaN	Low ($\sim$0)	100-150 (132)
Group B	WT12	Low-Medium (35)	Low ($\sim$0)	Low ($\sim$0)	80-100 (90)
Group C	WT3	Medium-High (48)	High (17)	High (1.2K)	0
	WT6	Medium-High (48)	High (17)	High (1.2K)	0
Group D	WT16	Vary between high (60) and low-medium (35)	Vary between high (17) and low ($\sim$0)	Vary between high (1.2K) and low ($\sim$0)	0
Group E	WT1	High (60)	High (17)	High (1.2K)	0
	WT5	High (60)	High (17)	High (1.2K)	0
	WT7	High (60)	High (17)	High (1.2K)	0
	WT8	High (60)	High (17)	High (1.2K)	0
	WT9	High (60)	High (17)	High (1.2K)	0
	WT11	High (60)	High (17)	High (1.2K)	0
	WT13	High (60)	High (17)	High (1.2K)	0
	WT14	High (60)	NaN	High (1.2K)	0
	WT15	High (60)	High (17)	High (1.2K)	0

speed than the others. Because the rotor speed data was not available for turbine 10 and the generator speed data can be interpreted instead. The generator speed for turbine 10 indicates the same physical interpretation as with wind turbine 2: These wind turbines are not rotating. Furthermore, the positional deviation for turbine 10 has values between 100-150 indicating that the wind turbine is not optimally directed against the wind, which justifies the low rotation speed for the turbine. Turbine 2 is directed against the wind and might indicate that the brakes are activated or the angle for the blades are straightened. After reviewing the wind speed (presented in Appendix A.2, Figure A.2.8) it becomes clear that turbine 10 experiences elevated wind speeds and the brakes are activated due to these extreme local wind conditions. Turbine 2 does not experience similar extreme local wind conditions, but are possibly experiencing maintenance since it was not rotating now or in the previous interval (1 month earlier); this was confirmed by talking to the supervisors at Kongsberg Digital AS.

Group B: Wind turbine 12 is assigned to group B and has low to medium gearbox temperature. The position deviation of the turbine range between 80 and 100 degrees, having a mean of roughly 85-90 degrees. This confirms the low rotor speeds observed in the time series. The effect of the wind will diminish as it gets closer to 90 degrees and at 90 degrees the turbine will not be rotating regardless of wind speed. Furthermore, analysing the wind speed (Figure A.2.8) and summary table presented in Appendix A.2 in Table A.2.1) it is clear that the turbine was subjected to much higher wind (indicates extreme wind conditions) than the rest. Extreme wind conditions would automatically engage the brakes and stop the turbine. However, this does not explain the slightly elevated temperature level observed, compared to the temperature levels of the turbines in group A. Since the turbine has a low rotor speed (basically not rotating at all), the gearbox temperature should be similar to that of group A (WT2 and WT10). That is why the turbine is not clustered along with the remaining turbines in group A. This relatively high temperature in the gearbox might indicate faulty conditions for the corresponding turbine or scaling issues for the temperature sensor. Regardless, additional attention or maintenance might be required.

Group C: Turbines within this group are characterised by medium-to-high gearbox temperatures, high rotor/generator speed and is directed against the wind. These turbines have lower temperature levels than those of group E, despite having the same rotational speed. The lowered gearbox temperature could indicate that the turbines are less worn out, better lubricated, or of a different turbine model. The latter was clarified when analysing the first data set ('Univariate_V1') in Section 6.1.3: These turbines are all of the same models as indicated by their maximum power production value. The hypothesis is therefore that these turbines have parts which are less worn out or better lubricated than the other turbines in group E. However, it proves hard to verify and therefore remains a hypothesis.

Group D: This group contains only one time series and belongs to turbine 16. Turbine 16 is very different from all the rest. The turbine has quite varying gearbox temperature which can be explained by the equally varying rotor/generator speed (presented in Appendix A.2, in Figure A.2.6). One interpretation of this is that the turbine is subjected to highly varying wind conditions, especially compared to the other turbines analysed. However, after comparing the wind conditions in Figure A.2.8) it becomes clear that the local wind conditions are roughly identical to the others and does not explain the deviations from the rest. Another hypothesis is that the braking system is malfunctioning or that controller controlling the pitch angle of the blades is malfunctioning. However, either of these hypothesis proves hard to verify with the information acquired.

Group E: Common for all turbines within this group are high gearbox temperature, high rotor/generator speed and a corresponding position deviation of zero (turbines are directed against the wind). Time series experiencing elevated temperature levels are due to the fact that the turbine is actively rotating and producing heat because of friction in the gears/bearings. However, these turbines have elevated temperature level and might indicate that these turbines are more worn out or less lubricated than those of group C with lower gearbox temperature levels. This is not transparent in the rotor speed data (or any of the other parameters analysed) and the assumptions prove hard to verify.

6.2.3 Similarity in shape - Hierarchical clustering of the normalised data set

The objective is now to cluster the time series with the objective of similarity in shape and not the absolute difference and offset. Therefore, the cluster analysis is performed on the normalised data set. The trend is not removed prior to the normalisation as its important to the overall shape and we wish to capture this behaviour in the cluster analysis. The time series which will be clustered with regards to the objective of similarity in shape can be viewed in Figure 5.10. For similarity in shape, the DTW distance (3.7) is used instead of the Euclidean distance. Same constraint as in Section 6.1.4 is applied to the DTW algorithm. DTW has much higher complexity then Euclidean distance and the calculation of the distance matrix took 408 seconds compared to Euclidean distance which took only 1.31 ms. The condensed distance matrix is furthermore fed to the hierarchical clustering algorithm. In order to find the appropriate linkage criterion and the optimal number of clusters, the internal indexes are analysed. The best method for hierarchical clustering will be found by comparing the *Cophenetic correlation coefficient* and the number of cuts to the dendrogram will be determined by viewing the internal performance indexes: *Silhouette index* and *MSSSE*. More information about these specific indexes can be found in Chapter 3. The summary plot and table for the internal indexes of cutting each dendrogram - each with different linkage criterion - into k cuts can be seen in Figure 6.27 and Table 6.6. The cophenetic correlation coefficient can be seen in the upper right corner in the aforementioned figure (Figure 6.27).

Table 6.6: Summary table for each linkage criteria supported by the hierarchical clustering algorithm: single, complete, average and ward. The silhouette index and mean sum of the sum of squared error is plotted for each cut in the dendrogram.

Method	Internal index	Number of cuts to the dendrogram											
		K = 2	K = 3	K = 4	K = 5	K = 6	K = 7	K = 8	K = 9	K = 10	K = 11	K = 12	K = 13
Single	Silhouette	0.32	0.37	0.36	0.34	0.33	0.33	0.22	0.17	0.13	0.18	0.15	0.13
	MSSSE	8.12	5.96	5.18	4.36	3.69	3.08	2.47	2.08	1.42	0.88	0.59	0.33
Complete	Silhouette	0.32	0.37	0.34	0.34	0.33	0.21	0.21	0.19	0.21	0.16	0.15	0.13
	MSSSE	8.12	5.96	5.37	4.38	3.69	2.89	2.29	1.82	1.28	0.90	0.59	0.33
Average	Silhouette	0.32	0.37	0.36	0.34	0.33	**0.33**	0.22	0.20	0.15	0.18	0.15	0.13
	MSSSE	8.12	5.96	5.18	4.36	3.69	**3.08**	2.47	1.70	1.32	0.88	0.59	0.33
Ward	Silhouette	0.35	0.37	0.36	0.34	0.33	0.20	0.21	0.20	0.23	0.18	0.15	0.13
	MSSSE	6.89	5.96	5.18	4.36	3.69	2.82	2.21	1.70	1.26	0.88	0.59	0.33

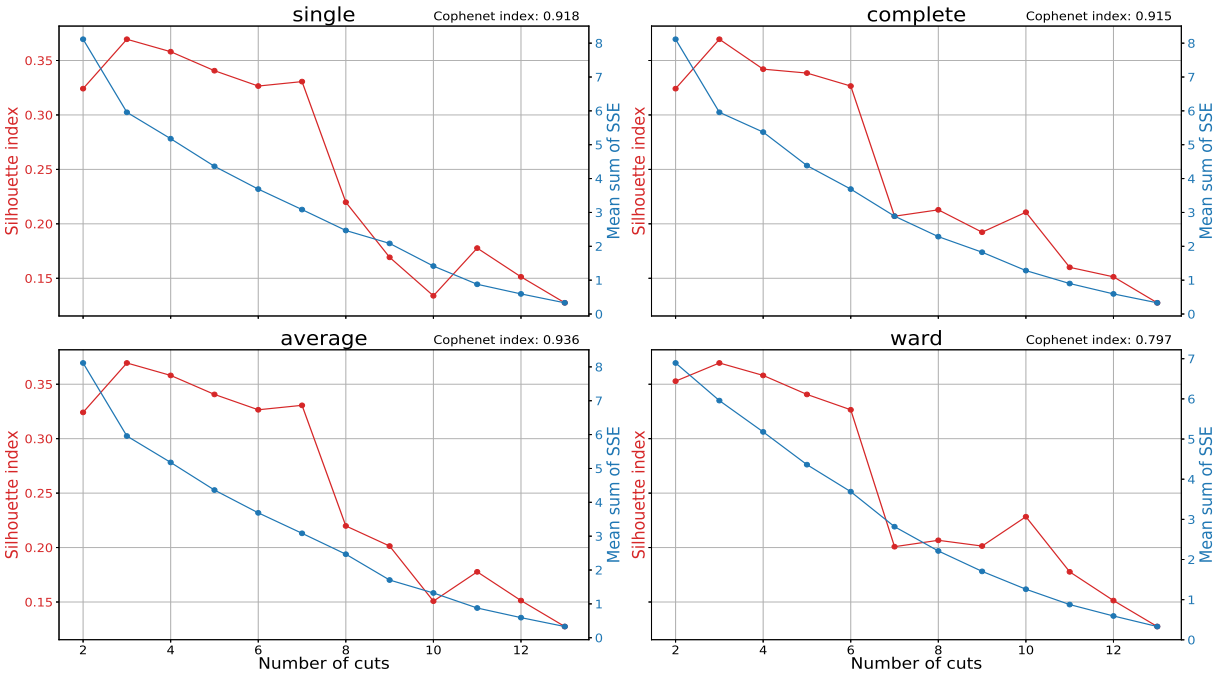


Figure 6.27: Summary plot for each linkage criteria supported by the hierarchical clustering algorithm: single, complete, average and ward. The silhouette index and mean sum of the sum of squared error (MSSSE) is plotted for each cut in the dendrogram.

First of all, reviewing the cophenetic correlation coefficient in Figure 6.22, it can be observed that all dendrograms have relatively high coefficients close to 1 (all over 0.75). The dendrogram constructed with linkage criterion set to *average* have the largest correlation coefficient which represents the dendrogram with a high-quality solution. The dendrogram built with average as the linkage criterion will, therefore, be cut and its assignments will be analysed. Secondly, reviewing the summary table and the corresponding summary plot, it can be observed that the silhouette index is largest for a cut of three, with a silhouette index value of 0.37. From here it slowly decreases until after a cut of seven where it suddenly plummets. The silhouette index value for a cut of seven is 0.33, which is only a 0.04 reduction from a cut of three. The corresponding MSSSE value decreases from 5.96 to 3.08, from a cut of three to a cut of seven. The reduction is significant for the MSSSE index and only a small reduction in the silhouette index value is observed during this interval. Arguably, a cut of three, four, five, six or seven could be chosen as the optimal number of cuts. The difference between the different cuts is simply the ratio between separation and intra-cluster similarity: A cut of three has the largest separation between the clusters and a relatively large intra-cluster similarity; a cut of seven has the smallest separation between the clusters but has also a relatively small within-cluster variance. The latter partitions the time series based on less significant differences and would be more interesting to analyse in terms of acquiring physical interpretations and finding time series which are behaving slightly different from the rest. Therefore, a cut of seven to the dendrogram is chosen as the best compromise between separation and intra-cluster similarity. The optimal configuration for the data set is to cluster the data set with *average* as the linkage criterion and cutting the dendrogram to form *seven clusters*. The dendrogram - with average as linkage criterion - along with visualisation of the corresponding time series can be seen in Figure 6.28 and 6.29 (in the next section), respectively.

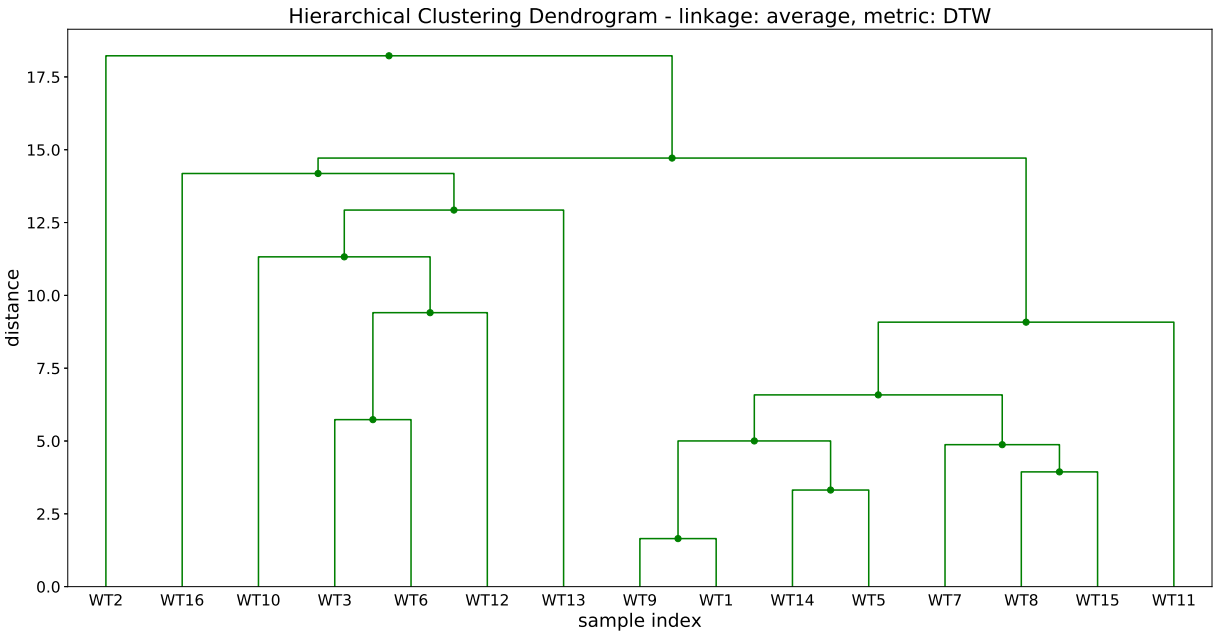


Figure 6.28: Dendrogram with average as the linkage criterion and distance as the DTW distance implemented on 'Univariate_V2' data set.

6.2.4 Analysing the clustering results from similarity in shape

In this section, the clustering results from clustering 'Univariate_V2' data set in Section 6.2.3 will be analysed. The corresponding dendrogram with the objective of clustering the time series with respect to similarity in shape can be seen in Figure 6.28 or 6.29. The absolute difference and the offset will be ignored to a certain degree, as normalising the data set will remove the offset and scale the data between 0 and 1. From analysing the internal indexes in the previous section, best cut to the dendrogram was found to be a cut of seven.

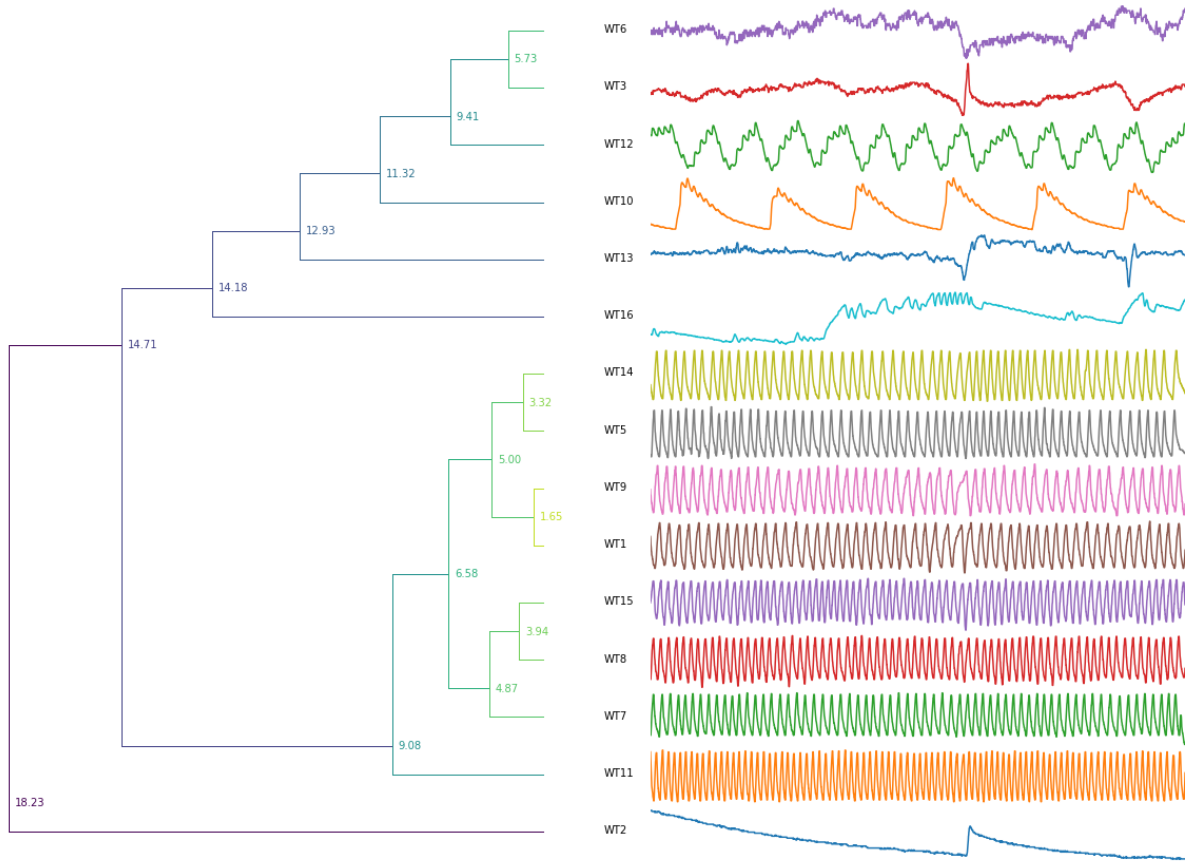


Figure 6.29: Summary of hierarchical clustering with DTW as similarity measure. Average is chosen as the linkage criterion.

Initially, we can observe that when using DTW as the distance measure for calculating the dissimilarity matrix it manages to group the most similar time series in shape and separate the most dissimilar ones from the rest. This is especially true for a cut of three to dendrogram with 'average' as the linkage criterion. This clearly separates the time series experiencing rather stationary periodically oscillations of higher frequency to the one experiencing quite different behaviour. A cut of two only separated the time series associated with wind turbine 2. This is justified by that it is very different from all the rest. A much more detailed separation which results in only similar time series within each cluster is achieved by cutting the dendrogram into seven partitions, as justified in the previous section. A cut of seven separates turbine 2, 10, 12, 13 and 16 into its own cluster containing only itself. The grouping - from top to bottom in Figure 6.29 - are now the following: Group A: {WT6, WT3}; Group B: {WT12}; Group C: {WT10}; Group D: {WT13}; Group E: {WT16}; Group F: {WT14, WT5, WT9, WT1, WT15, WT8, WT7, WT11}; Group G: {WT2}. The plots for the generator speed, position deviation and the wind speed were presented in Section 5.2.1 (which refers to Appendix A.2 for the plots). The summary table for all turbines with descriptive statistics was presented in the analysis of similarity in time in the previous section (Table 6.5). These parameters will be used to interpret the physical meaning behind the resulting partitioning. The groupings will now be analysed and interpreted in the following paragraphs.

Group A: Contains time series from turbine 3 and 6. These are characterised by low to no oscillations with small fluctuation in the signal. The cooling pump - which is causing the oscillations - does not seem to be engaged during the entire interval. The temperature remains relatively constant through the interval, with the exception of

one bump after the midpoint. The bump can be explained by viewing the time series for the wind speed in Figure A.2.8 which experience a decrease and right after an increase in wind speed (right after the midpoint). The lack of oscillatory behaviour could be explained by that the threshold values for the cooling pump are very small or that the gearbox temperature reaches its maximum value (i.e. equilibrium), which is lower than the upper threshold for the cooling pump (i.e. when the cooling pump should be activated). The latter could be further explained by that the parts within the gearbox are less worn out or are better lubricated, such that the gearbox temperature will not exceed this threshold value for the pump regardless of rotor/generator speed. The former hypothesis is most likely not correct as the overall variance is quite large. The latter hypothesis - as before - proves hard to verify.

Group B: Contains only the time series corresponding to turbine 12. The time series is characterised by lower frequency oscillations. The signal also contains a higher frequency signal component with much lower amplitude. The cooling pump enables when it reaches the upper value and then cools the gearbox down until it reaches the minimum value (both values is set by the parameters for the cooling pump). The wind experiences sudden decrease and then an increase right after the midpoint; the time series does not indicate any changes after the midpoint due to this. This could indicate that the time series of the gearbox temperature is not dependent on the wind speed. The only case in which this is true is when the turbine is not rotating. As with the objective of similarity in time, the wind speed was found to be of critically high level, indicating that the brakes must be engaged.

Group C: Contains only the time series corresponding to turbine 10. Its characteristics are similar to that of the time series assigned to group B but the frequency for which it oscillates with, has decreased by a factor of three. Same interpretation as for group B applies to this turbine: The temperature does not follow the same trend as the wind speed. However, after reviewing the wind speed, it is clear that the wind speed experience critically high levels, indicating that the brakes must be engaged.

Group D: Contains only the time series corresponding to turbine 13. This turbine is characterised by low to no oscillations with small fluctuation in the gearbox temperature. The cooling pump - which is causing the oscillations - does not seem to be engaged during the entire interval. The temperature remains relatively constant through the interval, with the exception of one bump after the midpoint. The bump can be explained by viewing the time series for the wind speed in Figure A.2.8 which experience a decrease and right after an increase in wind speed (right after the midpoint). Same physical interpretation as for those in group A holds for this turbine. The reason that this turbine is not clustered with the others is simply that they are different in terms of shape. The time series associated with turbine 13 have relatively large values in the first half of the time series and even greater values in the second. This is not the same shape as the two other time series in group A - they have a reduction in the temperature level after the midpoint.

Group E: Contains only the time series corresponding to turbine 16. The turbine has quite varying gearbox temperature and experiences a small number of oscillations. Intervals where the gearbox temperature slightly decreasing in a 'straight' line indicate intervals where the turbine is not rotating. Same interpretation as for analysis of similarity in time applies: The temperature varies with the generator speed. However, the generator does not seem to stop because of the lack of wind or a position deviation different from zero but stops for different reasons. Reasons could be the malfunctioning of the brakes or the malfunctioning of the pitch controller of the blades or something completely else. These assumptions prove hard to verify and additional maintenance might be required. Regardless, the turbine is effectively separated from the rest.

Group F: Turbines associated with this group contains the majority of the turbines and could indicate normal operational behaviour for rotating turbines. The assigned wind turbines within this group are actually the same assignment - with the exception of turbine 13 - formed from clustering with the objective of similarity in time. Turbine 13 was not found to be of any interest for the analysis of similarity in time. But during this analysis, it can be seen that the shape is quite different; it does not experience the same periodic oscillations as the turbines in this group (group F). The faster the turbines rotate the more heat they produce which requires more "aggressive" cooling. The upper threshold for the cooling pump is reached at the peaks and the cooling pump enables until it reaches the lower threshold value. This explains the higher frequency oscillations observed for the temperatures. The fact that it reaches the upper threshold very often suggest that the somewhere in the gearbox additional heat is produced because of poor lubrication or more worn out parts. Viewing the dendrogram, a deeper cut would separate turbine 11 from this group (group F). The time series of the gearbox temperature corresponding to this turbine has larger frequency oscillations than the rest of group F, despite having the same rotational speed and wind

conditions. A possible explanation for this behaviour is that the component in the gearbox of turbine 11 is even more worn out or worse lubrication than the remaining turbines in this group. If this is the case, the temperature increases faster along with an appropriate reaction of the cooling pumps (i.e. faster cooling).

Group G: Contains only the time series corresponding to turbine 2. No periodic oscillations are observed, but time series is affected by the change in the wind slightly after the midpoint. Reviewing the generator speed it becomes clear that during that period it experiences a peak value where the turbine is actually rotating. This explains the sudden increase in temperature levels observed in the gearbox. During the remainder of the interval the turbine is not rotating or increasing its temperature (rather trying to reach its equilibrium value with the temperature outside). No oscillatory behaviour is present which might indicate that the thresholds values for the cooling pump are not exceeded.

6.3 Summary and comparison of the two data sets

In order to strengthen the physical interpretation of the clustering results from the first data set analysed, the experiment was repeated on a different set of measurements (same turbines and sensors sampled at a different interval). First of all, the physical interpretation of the results from the objective of similarity in time - for both data sets - will be compared. Then, the same procedure will followed for the objective of similarity in shape.

6.3.1 Similarity in time

First of all, clustering the time series based on the temperature in the gearbox results in a clear separation of the rotating and non-rotating wind turbines for both data sets analysed. This is especially true for the first data set analysed, where two non-rotating turbines were present. Regardless of the number of cuts made, the clustering algorithms manage to separate the non-rotating turbines from the rotating turbines and place the non-rotating turbines into its separate cluster. The second data set contains a collection of more diverse time series and proves harder to find the correct number of clusters to get this separation. Initially, it manages to separate two turbines which were not rotating into the same cluster. However, one turbine - which is not rotating nor clustered in the same group - experiences elevated gearbox temperature despite it not rotating. It was therefore not clustered together with the other non-rotating turbines but was placed in a separate cluster containing only itself; this is expected as it shows abnormal temperature levels despite not rotating. Therefore, for the second data set, two clusters were formed: The first cluster contains the turbines which are not rotating and has temperature levels which comply with that; the second cluster contained the one turbine which is not rotating but experienced significantly higher temperature levels than the others which did not rotate. Clustering time series based on the temperature of the gearbox successfully manages to separate turbines of different rotational speeds (rotating and non-rotating). For all turbine, the low gearbox temperatures observed is justified by an equally low generator speed which is for most turbines a result of extreme local weather conditions. The exception of this is turbine 2 for both data sets which was subjected to maintenance during both intervals and was manually deactivated from the grid. This becomes more transparent when reviewing the time series with respect to their behaviour or shape.

Secondly, a larger number of cuts to the dendrogram or larger predetermined clusters to the K-means algorithms was shown to separate turbines with respect to smaller dissimilarities. For example, turbines which have possibly more worn out parts in the gearbox or poorer lubrication are well separated. The fact that these turbines are poorer lubricated or have parts which are more worn out still remains a hypothesis and proves hard to verify with the information acquired. Nonetheless, clustering with the objective of similarity in time still manages to separate the time series behaving differently than others. The practical implication for clustering the turbines with respect to their temperature levels can be extended to automatic classification of turbines experiencing slightly different temperature levels, and not only rotating and non-rotating turbines. The practical application for the classification of the turbines depends on the depth of where the dendrograms are cut (i.e. how low the cut is in the dendrogram). This will be addressed in the next paragraph.

Deeper cuts to the dendrogram in many cases result in a reduction in the silhouette index and a definitive reduction in the MSSSE value. The intra-cluster similarities within each cluster formed becomes more similar with the increasing number of cuts (i.e. the time series within each cluster are more similar to each other), but the separation between the clusters formed becomes less defined. By keeping this in mind, the user can customise the specific objective to the particular application at hand. If a large number of cuts is used, the algorithm manages to separate time series of smaller difference which might be of interest when interpreting the physical meaning of the groupings or looking for outliers. The practical applications can vary between the classification of turbines experiencing high and low temperature levels or slightly reduced temperature levels which could indicate turbines which are more worn out or poorer lubricated. In other words, the compromise between separation and similarity becomes an important consideration for the specific application and classification problem solved.

Lastly, it was shown during the analysis of the first data set that K-means produces inconsistent results. Additionally, the benefit of linear time complexity for the K-means algorithm - compared to the quadratic time complexity of the hierarchical clustering algorithm - is not prominent in the current analysis as the number of turbines (or objects) within each data set are only 15. For such small data sets, the benefit from using K-means over hierarchical clustering algorithm is nonexistent. Because of these remarks, K-means was not applied to the second data set and will not be applied in the multivariate clustering problem in the next chapter either. The hierarchical clustering algorithm is much more advantageous over K-means when it comes to visualisation and consistency. This is really important in the multivariate case, where the clustering assignment tends to be much harder to interpret. Additionally, as the number of turbines within each wind park still remains relatively low, K-means does not benefit from having lower time and space complexity than the hierarchical clustering algorithm. For the multivariate case, the

same number of turbines will be clustered, only with including additional parameters (or dimensions).

6.3.2 Similarity in shape

When clustering with the objective of similarity in shape, both of the data sets analysed showed that it effectively manages to group the time series which were most similar in shape and separate the most dissimilar time series into their own individual clusters. Time series which does not have any oscillatory behaviour is separated into either individual clusters or clusters with time series alike. Time series with no oscillation might indicate faulty conditions for the cooling pump or that the threshold values are not within its optimal range. As addressed, this assumption cannot be verified with the information acquired and still remains a hypothesis. Furthermore, turbines experiencing sudden halts in the rotation of the rotors due to elevated wind conditions are also effectively captured by the algorithms. The time series experiencing sudden halts in the rotation will automatically result in non-stationary behaviour of the gearbox temperature; reduction in the rotor speeds results in a reduction in the gearbox temperature, and vice versa. Because of the nature for some of the time series (the values can go from a minimum value to a maximum value in a matter of few time steps), care must be taken when normalising such time series. This will be addressed in detailed in Section 7.1, where typical normalisation cannot be applied to one of the multivariate data sets.

In Appendix B.1 and B.2, cluster analyses with the objective of similarity in shape have been performed without constraining the DTW algorithm (Sakeo-Chiba band not implemented). First of all, let us review the similarity in shape analysis of the 'Univariate_v1' data set without constraining the DTW algorithm - results are presented in Appendix B.1 - and compare it to the one in Section 6.1.4 which has implemented the DTW distance with a global constraint. Several differences and similarities can be observed for both of the analysis. First of all, the dendrograms (in Figure 6.20 and B.1.2) can be observed to be quite similar to each other with the exception of a couple of turbines. The major differences are in the turbines which originally can be observed to be quite different to the others; time series which are relatively similar to each other is not affected by whether or not the DTW algorithm is constrained. This was shown in Appendix C.1, where the warped path for similar time series did not deviate much from the diagonal (this can be observed in both Figure C.1.3 and C.1.4). On the other hand, the warped path between time series which are obviously different experiences a warped path which deviates quite a lot from the diagonal and maps a large number of points to a few numbers of points (this can be observed in both Figure C.1.5 and C.1.6). This is an unwanted scenario as it forces similarity between two time series where it should not be. This can also be observed for the second univariate data set analysed in Section 6.2.3. Comparing the dendrograms between the constrained and non-constrained case (Figure 6.28 and B.2.2 in Appendix B.2, respectively), show very similar results, but the separation between the abnormal looking time series can be observed to be more significant for the constrained case. This is indicated by the large average distance between the dissimilar time series. The similarities between the dendrograms and the cluster assignments for both the constrained and unconstrained case are quite small, but in a more realistic case where the data set may contain several hundreds of turbines, the results can be severe. The DTW should, therefore, be implemented with the Sakeo-Chiba band.

The overall similarities in shape are captured to a great extent by the clustering algorithm when DTW is used as the similarity measure. However, it comes at a cost. The constrained DTW has a pretty high running time compared to Euclidean distance (over 400 seconds for the DTW distance compared to 1.31ms for Euclidean distance) and is not suitable for very large data sets or applicable for real-time applications.The running time between different implementation of the DTW algorithm can be observed in Appendix C.2. For the unconstrained case, the running time of the algorithm was reduced by a factor of nearly 10, where the fast implementation ('*dtw.distance_matrix_fast*') was applied, instead of the normal implementation ('*dtw.distance_matrix*'). Sadly, the fast implementation did not work with the implementation of the *Sakeo-Chiba band* and the other, slower function had to be used. The comparison between the running times of the '*dtw.distance_matrix*' function - with and without Sakeo-Chiba band implemented - show a reduction in the running speed by a factor of 5 for the constrained case. Already now, we can say with certainty that the DTW algorithm (especially the implementation of the DTW in the python library, *dtaidistance*) are not suitable for larger data sets. However, because the DTW algorithm manages to capture the similarities in shape between time series very well, it will further be utilised on a multivariate case, despite the poor scalability of the similarity measure.

Chapter 7

Clustering of multivariate time series

In this chapter, the multivariate data sets introduced in Section 5.2, will be clustered and the clustering results will be analysed. The clustering results and analysis of the first data set ('Multivariate_V1') and second data set ('Multivariate_V2') will be presented in Section 7.1 and 7.2, respectively. In Section 7.3, the clustering results and analyses of both data set will be summarised and compared. The only difference between doing a cluster analysis on a multivariate data set, versus a univariate data set, is how the dissimilarity matrix is constructed. A multivariate data set can simply be thought of as a set of univariate data sets; each of these univariate data sets can be clustered as done previously, namely by constructing a dissimilarity matrix. Now, in order to cluster multivariate time series, the individual dissimilarity matrices of each individual univariate data set - each associated with a weighting value (alpha) - are added together. This is illustrated in Equation (3.2). The weighting vector (alpha) contains the weighting value for each individual data set and can be adjusted according to the specific application and preference. The Python implementation of the multivariate cluster analysis is the same as for the univariate data set, only some minor changes to the script in Appendix D.1.5 is required. For running the cluster analysis on either of the multivariate data sets, uncomment the data set which the analysis will be run on and comment out the remaining code specific to the other data sets. Examples of how this is done are already provided in the implementation of the univariate data sets.

7.1 Clustering of 'Multivariate_V1'

The first multivariate data set which is going to be clustered is the 'Multivariate_V1' data set introduced in Section 5.2.1. This data set is an extension of the 'Univariate_V2' presented in Section 6.2, which only contains time series of the gearbox temperature. The multivariate data set is comprised of the univariate data set from 'Univariate_V2' data set (i.e. the gearbox temperature) in addition to three new dimensions: generator speed, wind speed and position derivative. By clustering this combination of univariate data sets, it is expected to get similar (and hopefully better) results as for clustering only the gearbox temperature. When clustering the multivariate data set, the results and analysis will be presented in a similar fashion as for the univariate case. In Section 7.1.1, the data set will be clustered with respect to the objective of similarity in time and the results will be presented, followed by an analysis of the results in the consecutive section, Section 7.1.2. Then, the same layout will be used when clustering the data set with the objective of similarity in shape. The results is presented in Section 7.1.3 followed by an analysis of the results in Section 7.1.4. Lastly, a comparison between the clustering results of the univariate data set - where only the gearbox temperature was included in the cluster analysis - and the clustering results of the multivariate data set will be made in Section 7.1.5.

7.1.1 Similarity in time - Results from hierarchical clustering of the scaled data set

The filtered data set contains features which are of different units of measurements. This makes adding together dissimilarity measures inconvenient. Before applying any kind of clustering, the time series have to be scaled. For convenience, the univariate data sets are scaled (4.4) individually so that the minimum and maximum values for each univariate data set are 0 and 1, respectively. This is different from normalisation used in the similarity in shape analysis as we now scale the entire data set and not individual time series within that data set. As all the univariate data sets - which the multivariate data set is comprised of - are individually scaled, clustering of the

data set can begin. Similar to Chapter 6, the best linkage criterion for the hierarchical clustering algorithm will be found by comparing the *Cophenetic correlation coefficient* (3.10) and the number of cuts to the dendrogram will be determined by inspecting the internal performance indexes: *Silhouette index* (3.14) and *MSSSE* (3.17). More information about these specific indexes can be found in Chapter 3. The summary plot and table for the internal indexes of cutting each dendrogram into k partitions - each with different linkage criterion - can be seen in Figure 7.1 and Table 7.1. The cophenetic correlation coefficient can be seen in the upper right corner in the aforementioned figure (Figure 7.1). In the current run, the alphas are all equal to one, which weight the different univariate time series equally.

Table 7.1: Summary table with the linkage criteria: single, complete, average and ward. The data set which was clustered is 'Multivariate_V1' with the objective of similarity in time. The silhouette indexes and MSSSE values for a cut K - from 2 to 13 - is presented.

Method	**Internal index**	**Number of cuts to the dendrogram**											
		K = 2	K = 3	K = 4	K = 5	K = 6	K = 7	K = 8	K = 9	K = 10	K = 11	K = 12	K = 13
Single	Silhouette	0.73	0.63	0.60	0.57	0.25	0.12	0.12	0.12	0.14	0.12	0.12	0.10
	MSSSE	57.04	25.97	12.99	6.99	3.87	3.44	2.86	1.91	1.18	0.90	0.52	0.33
Complete	Silhouette	0.73	0.63	0.60	0.57	0.25	0.16	0.14	0.13	0.14	0.12	0.12	0.10
	MSSSE	57.04	25.97	12.99	6.99	3.87	2.92	2.35	1.62	1.18	0.80	0.52	0.33
Average	Silhouette	0.73	0.63	0.60	**0.57**	0.25	0.16	0.14	0.12	0.14	0.12	0.12	0.10
	MSSSE	57.04	25.97	12.99	**6.99**	3.87	2.92	2.35	1.91	1.18	0.80	0.52	0.33
Ward	Silhouette	0.70	0.66	0.60	0.57	0.25	0.15	0.15	0.13	0.14	0.12	0.12	0.10
	MSSSE	47.90	25.14	12.99	6.99	3.87	2.90	2.26	1.62	1.18	0.80	0.52	0.33

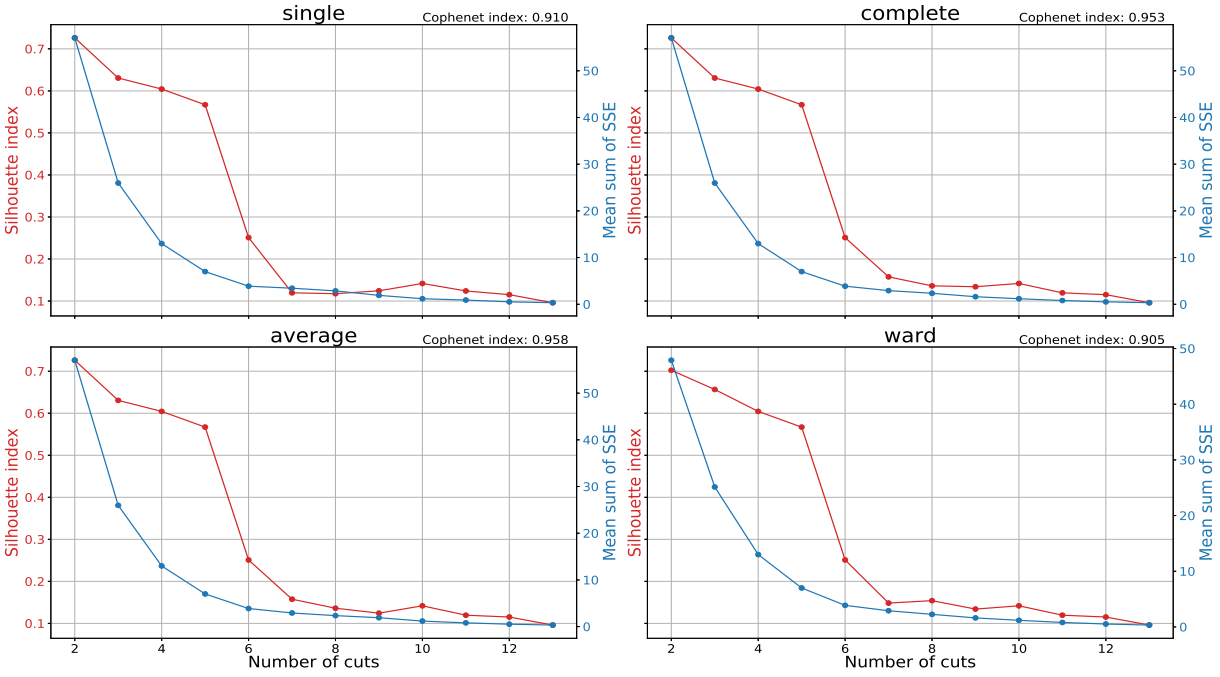


Figure 7.1: Summary plots with the linkage criteria: single, complete, average and ward. The data set which was clustered is 'Multivariate_V1' with the objective of similarity in time. The silhouette indexes and MSSSE values for a cut K - from 2 to 13 - is presented along with the cophenetic correlation coefficient in the upper right corner.

First of all, reviewing the cophenetic correlation coefficient in Figure 7.1, it can be seen that all of the dendrograms experience similar high values; all are above 0.9 and two of them experience values above 0.95. The cophenetic correlation coefficient is close to 1 for all dendrograms; values close to 1 indicate that the dendrograms represent a high-quality solution. The dendrogram with the highest cophenetic index is the dendrogram constructed with linkage criterion as *average*. Therefore, the dendrogram built with linkage criterion as *average* will be cut and its assignment will be analysed. Secondly, reviewing the summary table and the corresponding summary plot, it can be observed that the silhouette index is largest for a cut of two and slowly decrease in its magnitude until after

a cut of five where it suddenly drops. The corresponding MSSSE value during this interval decreases exponentially from 57.04 at a cut of two to 6.99 at a cut of five. A cut of five reduces the MSSSE values substantially where the silhouette index - from a cut of three and four - remains roughly constant. Therefore, a cut of five seems to be the optimal configuration where the compromise between separation and similarity is at its best. A cut of five will, therefore, be further analysed in the consecutive section. The corresponding dendrogram - with *average* as the linkage criterion - can be seen in Figure 7.2.

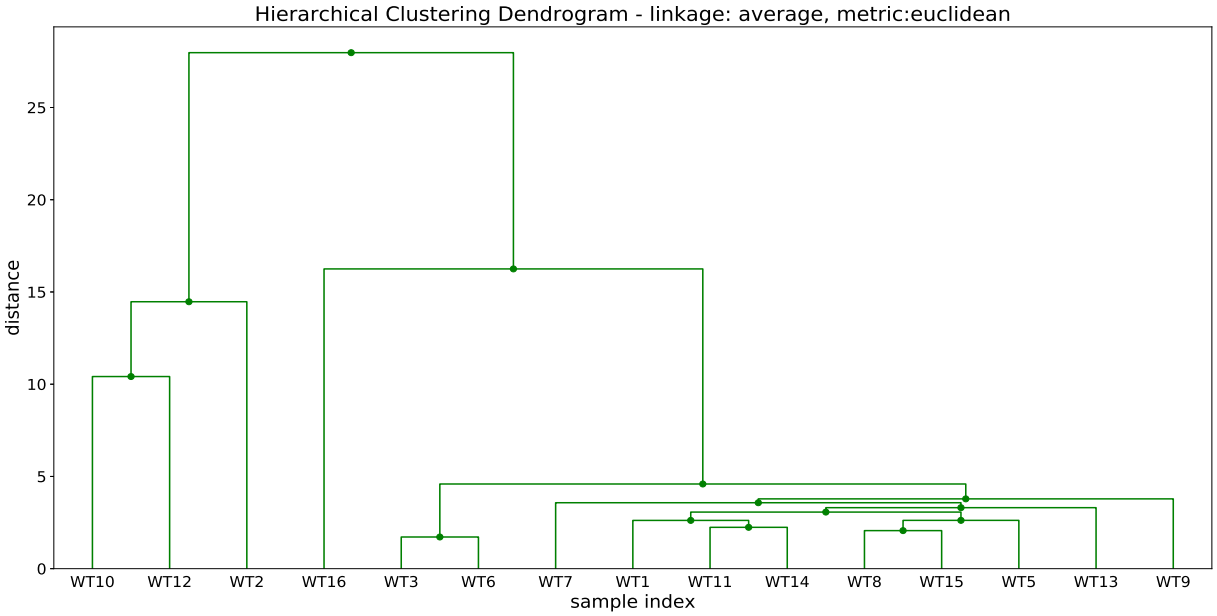


Figure 7.2: Dendrogram with *average* as linkage criterion and distance as Euclidean distance implemented on 'Multivariate_V1' data set. The weighting vector α is set to 1 for all elements.

7.1.2 Analysing the clustering results from similarity in time

In this section, the clustering results from clustering 'Multivariate_V1' data set in Section 7.1.1 will be analysed. The corresponding dendrogram which will be cut can be seen in Figure 7.2. In Figure 7.3 the cluster assignment and the time series are visualised for a cut of five. As before, the summary of the gearbox temperature, generator speed, wind speed and position deviation (difference in angle between the direction of the wind and the direction of the nacelle) is presented and can be seen in Table 7.2.

Initially, it can be observed from reviewing the dendrogram in Figure 7.2, that a cut of two to the dendrogram partitions turbine 2, 10 and 12 into its own separate cluster, which separates the non-rotating from the rotating turbines. From revisiting the internal indexes, this can be reflected by a large silhouette index value to be a good idea. But reviewing the within-cluster variance, the second cluster (i.e. rotating turbines) contains a few dissimilar turbines which results in a relatively large within-cluster variance. That is why the internal indexes recommenda a cut of five to the dendrogram. From observing the assignment, for a cut of five, in Figure 7.3, the first observation to be made is that there are four clusters which only contain one time series each; all of them deviates from cluster four, but have drastically different behaviour. Therefore, none of these individual time series are placed within the same cluster. The algorithm manages to separate turbines which behave differently from the majority (i.e. normal operations), which might indicate faulty turbines or at least turbines which requires some additional analysis. Secondly, it can be observed that the univariate data sets which stands for much of the variance between clusters are the univariate data set associated with the gearbox temperature, generator speed and the position deviation; these time series have the largest variety and the time series for the gearbox temperature is unique for each cluster; the generator speed and position derivative time series are not necessarily unique for each cluster. The wind speed for all the different clusters are quite similar; only cluster 1 and cluster 2 seems to deviate a bit from the rest. Thirdly, cluster 4 can be observed to have two turbines which are quite different in terms of the gearbox temperature. These were effectively separated in the univariate analysis where only the gearbox temperature was clustered, but not now in the multivariate case. This is because the wind speed, generator speed and the position deviation are very similar to the rest of the turbines associated with cluster 4 - three out of the four univariate time series state that

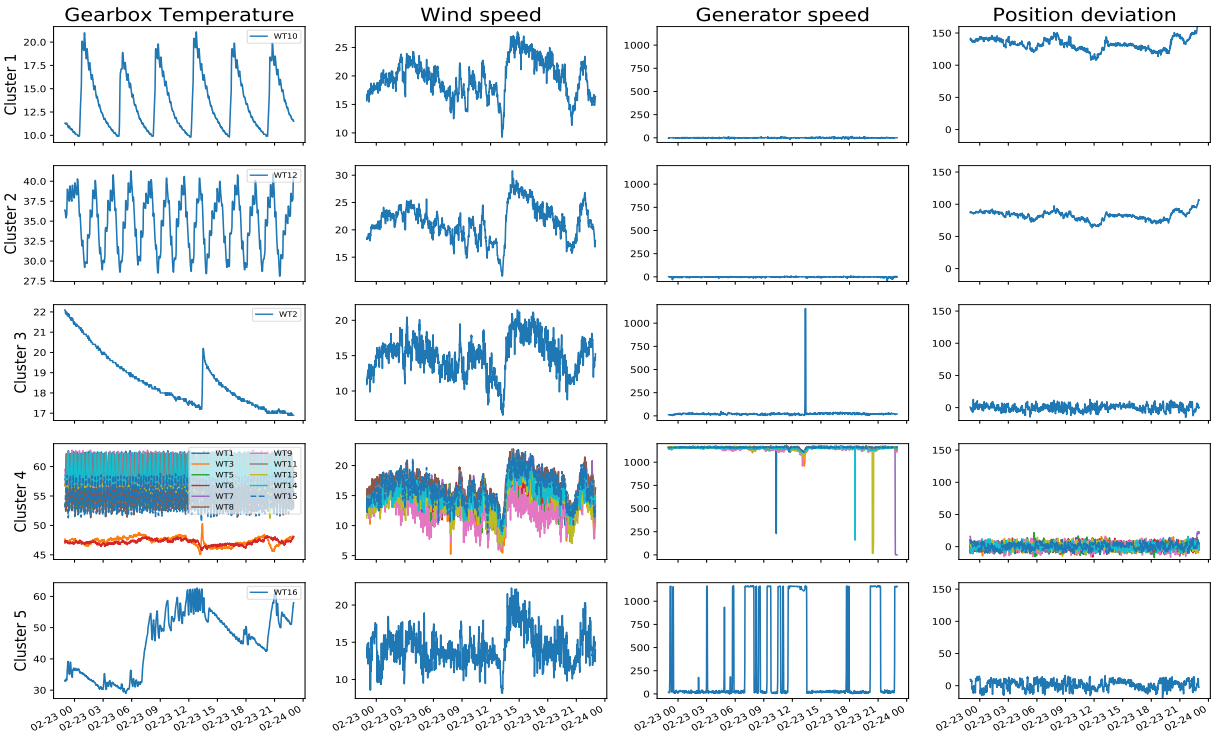


Figure 7.3: Clustering results from the 'Multivariate_V1' data set with Euclidean distance. Summary plot of a cut of five to the dendrogram with the linkage criterion set as 'average'.

Table 7.2: Summary of the properties of each time series within its respective cluster assignment (Group A, B and C) for a cut of 3. Numbers within parentheses (*) refers to the mean of the time series. The associated time series can be viewed in Figure 7.3

Cluster	Wind turbine number	Gearbox temp [Celsius]	Generator speed [RPM]	Wind speed [m/s]	Position deviation [degrees]
Cluster 1	WT10	Low (15)	Low (∼0)	19.6	100-150 (132)
Cluster 2	WT12	Low-Medium (35)	Low (∼0)	21.4	75-90 (80)
Cluster 3	WT2	Low (20)	Low (∼0)	14.9	0
Cluster 4	WT3	Medium-High (48)	High (1.2K)	14.4	0
	WT6	Medium-High (48)	High (1.2K)	14.6	0
	WT1	High (60)	High (1.2K)	14.2	0
	WT5	High (60)	High (1.2K)	14.6	0
	WT7	High (60)	High (1.2K)	15.8	0
	WT8	High (60)	High (1.2K)	16.7	0
	WT9	High (60)	High (1.2K)	12.4	0
	WT11	High (60)	High (1.2K)	15.6	0
	WT13	High (60)	High (1.2K)	14.1	0
	WT14	High (60)	High (1.2K)	14.9	0
	WT15	High (60)	High (1.2K)	16.1	0
Cluster 5	WT16	Vary between high (60) and low-medium (35)	Vary between high (1.2K) and low (∼0)	14.3	0

these turbines are very similar to each other. A deeper cut to the dendrogram would actually separate these from cluster 4 but is not recommended by the internal indexes. With the corresponding configuration, the algorithm effectively manages to separate abnormal behaving wind turbines from the rest. The physical interpretation of the underlying turbines within each cluster will be done in a similar manner as performed in Section 6.2.2: Gearbox

temperature, generator speed, wind speed and position deviation will be compared between and within clusters.

Cluster 1: This cluster contains only one turbine and it belongs to turbine 10. This turbine experiences a relatively low gearbox temperature in compliance with a non-rotating generator. It has position deviations not equal to zero and experience elevated wind speed level. These high wind speed conditions indicate that the brakes might be engaged and therefore the nacelle is not directed against the wind and the generator is not rotating.

Cluster 2: This cluster contains only one turbine and it belongs to turbine 12. This turbine experiences elevated gearbox temperature, despite not rotating. As the turbine is not rotating it should maintain similar temperature levels in the gearbox as for the turbine in cluster 1 or 3. Same physical interpretation as for the univariate cluster analysis holds: The temperature in the gearbox might indicate faulty conditions for the corresponding turbine or scaling issues for the temperature sensor.

Cluster 3: This cluster contains only one turbine and it belongs to turbine 2. The turbine is characterised by low gearbox temperature and equally low generator speed. The temperature level is comparable to that of cluster 1 with generator speed equal to zero. Nonetheless, the position deviation remains roughly around zero for the entire interval. Same physical interpretation as for the univariate case in Section 6.2.2 holds: Because the turbine is directed against the wind and is not rotating, it might indicate that the blades of the turbine are straightened and/or the brakes are engaged.

Cluster 4: This cluster contains the majority of the turbines. Common for these turbines is that they experience high gearbox temperatures (with the exception of turbine 3 and turbine 6), high generator speed, roughly the same wind speeds (no elevated wind conditions is observed) and are all directed against the wind. These turbines can be assumed to be working under normal conditions. Any deviations from this assignment are considered abnormal behaviour and should be further analysed.

Cluster 5: This group only contain one turbine and is the one turbine which experiences abnormal behaviour with quite varying temperature and generator speed. One hypothesis of this is that the turbine is subjected to highly varying wind conditions, especially compared to the other turbines analysed. However, reviewing the wind conditions in Figure 7.3, it can be seen that the wind has an average mean and shape similar to the others in cluster 3 and 4, and is quite constant throughout the entire interval. Another hypothesis is that the braking system or pitch controller for the blades are malfunctioning. However, this proves hard to verify with the information acquired. Regardless, it separates it from the rest, indicating that its different from the rest and requires additional attention.

7.1.3 Similarity in shape - Results from hierarchical clustering of the normalised data set

Before the clustering can initiate, some remarks to the preprocessing of the time series have to be made. The time series for the generator (or rotor) speed cannot be normalised as the others. This is because caution must be made when normalising flat but noisy time series. The noise will distort the average value and/or standard deviation of the time series which leads to improper translation. Clustering time series subjected to improper translation is meaningless. However, by scaling with the maximum and minimum value observed for all time series in the univariate data set, this can be avoided. This can be done because, throughout the interval, the maximum and minimum possible value for the generator speed is observed. The difference can be observed in Figure 7.4. Three time series (WT5: red line; WT10: yellow line; WT11: turquoise line) is present in the plot which experiences flat but noisy signals (i.e. does not experience any peaks as the others). Significant distortion to these time series - when subjected to normalisation - can be observed and clustering of such time series would result in clustering those turbines with respect to their noise level rather than their overall (flat) shape.

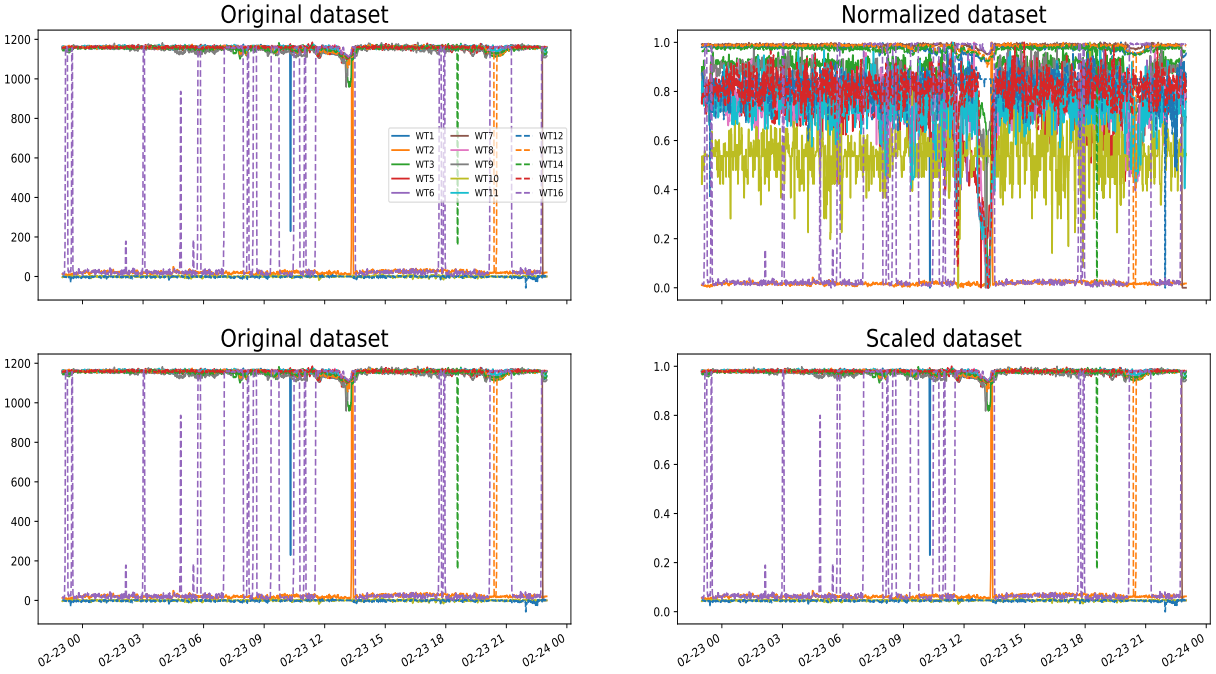


Figure 7.4: Difference between scaling and normalisation. If flat but noisy signals are normalised, the noise will distort the normalised signal (see turbine 5, 10 or 11).

Because of this, the time series - for the generator speed - are instead scaled, as done when analysing the data set with respect to the objective of similarity in time. This effectively suppressed the noise for flat signals. Essentially, these time series behave similarly to a digital signal: They do not experience intermediate values and the signal have only two levels, maximum and minimum. For time series such as this, it is important to scale them instead of normalising. The rest of the time series are normalised as described in Section 4.2.3.

The objective is now to cluster the time series with similarity in shape and not the absolute difference and offset. Therefore, the clustering analysis is performed on the normalised (and partly scaled) data set. For similarity in shape, the DTW distance (3.7) measure is used instead of the Euclidean distance. Same constraint as in Section 6.1.4 and 6.2.3 is applied to the DTW algorithm. DTW has much higher complexity then Euclidean distance and the calculation of the condensed distance matrix takes d additional time - d refers to the dimension of the multivariate data set. The calculation of the dissimilarity matrices now takes 1752 seconds (831 seconds parallelization with two cores). The condensed distance matrix is then fed to the hierarchical clustering algorithm. The best linkage criterion for the hierarchical clustering algorithm and the optimal number of clusters will be found with the same internal indexes used so far: Cophenetic correlation coefficient, Silhouette index and MSSSE. More information about these specific indexes can be found in Chapter 3. In this run, the alphas are all equal to one, which weight the different univariate time series equally. The summary plot and table for the internal indexes of cutting each dendrogram - each with different linkage criterion - into k cuts can be seen in Figure 7.5 and Table 7.3. The cophenetic correlation coefficient can be seen in the upper right corner in Figure 7.5.

Table 7.3: Summary table with the linkage criteria: single, complete, average and ward. The data set which was clustered is 'Multivariate_V1' with the objective of similarity in shape. The silhouette indexes and MSSSE values for a cut K - from 2 to 13 - is presented. Alpha for all univariate data sets is equal to one.

Method	Internal index	Number of cuts to the dendrogram											
		K = 2	K = 3	K = 4	K = 5	K = 6	K = 7	K = 8	K = 9	K = 10	K = 11	K = 12	K = 13
Single	Silhouette	0.65	0.61	0.58	0.25	0.26	0.18	0.11	0.09	0.05	0.04	0.08	0.05
	MSSSE	4.55	3.69	3.19	2.74	2.25	1.97	1.71	1.42	1.18	0.88	0.60	0.39
Complete	Silhouette	0.65	0.61	0.58	0.27	0.26	0.18	0.20	0.19	0.12	0.12	0.08	0.05
	MSSSE	4.55	3.69	3.19	2.61	2.25	1.97	1.63	1.35	1.09	0.84	0.60	0.39
Average	Silhouette	0.65	0.61	**0.58**	0.27	0.26	0.18	0.20	0.13	0.11	0.12	0.08	0.05
	MSSSE	4.55	3.69	**3.19**	2.61	2.25	1.97	1.63	1.37	1.12	0.84	0.60	0.39
Ward	Silhouette	0.65	0.61	0.31	0.27	0.26	0.18	0.20	0.19	0.18	0.12	0.08	0.05
	MSSSE	4.55	3.69	3.11	2.61	2.25	1.91	1.63	1.35	1.10	0.84	0.60	0.39

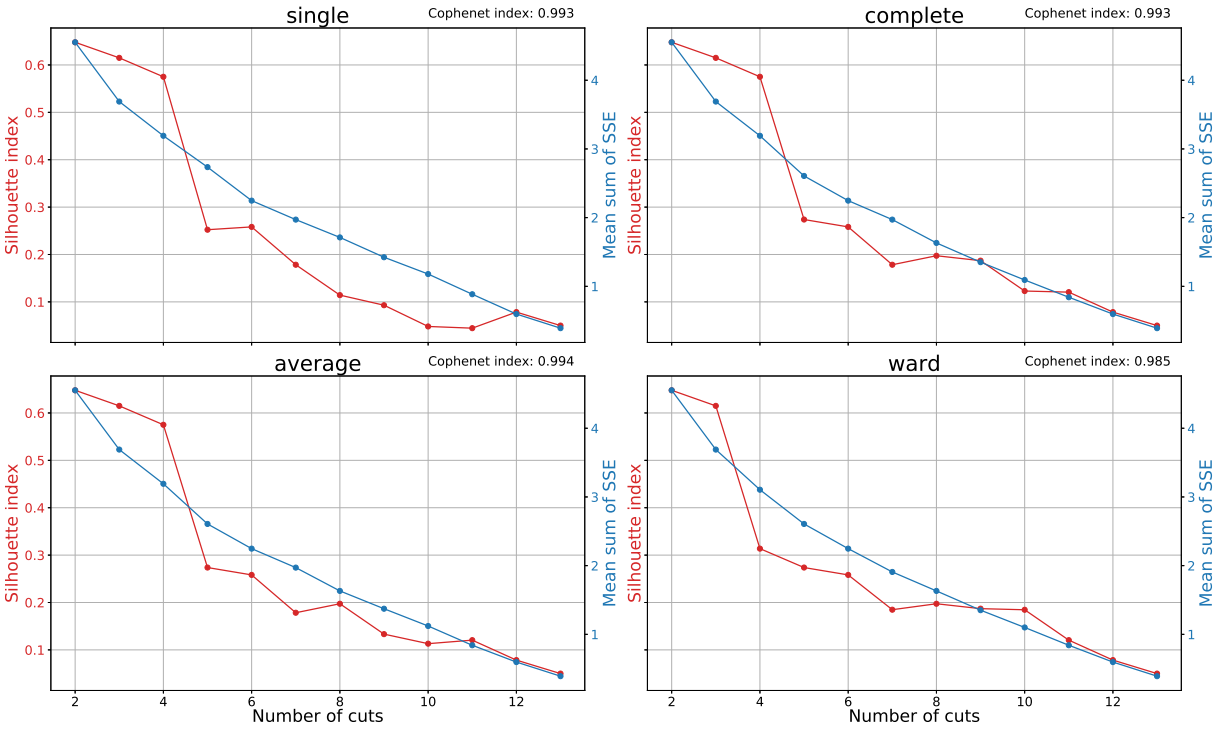


Figure 7.5: Summary plots with the linkage criteria: single, complete, average and ward. The data set which was clustered is 'Multivariate_V1' with the objective of similarity in shape. The silhouette indexes and MSSSE values for a cut K - from 2 to 13 - is presented along with the cophenetic correlation coefficient in the upper right corner.

First of all, reviewing the cophenetic correlation coefficient in Figure 7.5, it can be observed that all of the dendrograms experiences very large values for the cophenetic correlation coefficient. These values are very close to 1 and indicate a high-quality solution. The dendrogram with the highest cophenetic index is the dendrogram constructed with the linkage criterion set as *average*. Therefore, the dendrogram built with linkage criterion as *average* will be cut and its assignment will be analysed. Secondly, reviewing the summary table and the corresponding summary plot, it can be observed that the silhouette index is largest for a cut of two and decreases quite slowly up to a cut of four where it suddenly decreases quite rapidly; the corresponding MSSSE value during this periods decreases relatively steep. A fair reduction in the MSSSE value can be observed by cutting the dendrogram into four partitions, where the corresponding silhouette value still remains relatively high. Deeper number of cuts decreases the silhouette index value too much. Therefore, a cut of four seems to be the optimal configuration where the compromise between separation and similarity is at its best. A cut of four will be further analysed in the consecutive section. The corresponding dendrogram - with *average* as linkage criterion - can be seen in Figure 7.6.

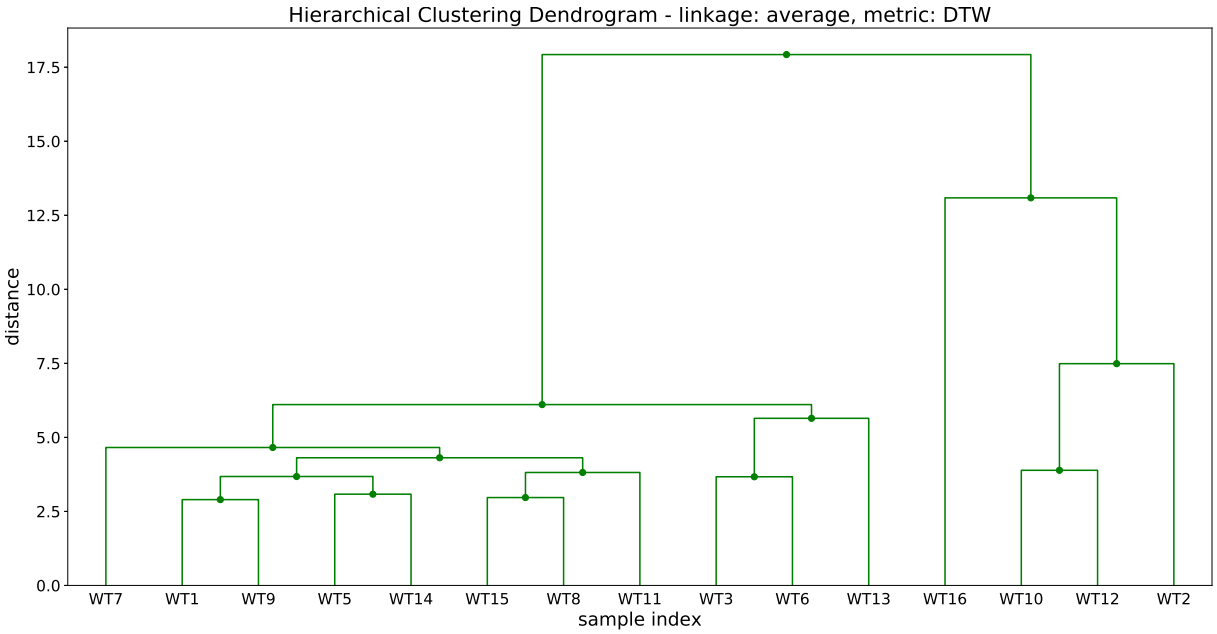


Figure 7.6: Dendrogram with average as linkage criterion and distance as DTW distance implemented on 'Multivariate_V1' data set. The weighting vector α is set to 1 for all elements.

7.1.4 Analysing the clustering results from similarity in shape

In this section, the clustering results from clustering 'Multivariate_V1' data set in Section 7.1.3 - with respect to the objective of similarity in shape - will be analysed. The corresponding dendrogram which will be cut can be seen in Figure 7.6. In Figure 7.7, the cluster assignment and the time series are visualised for a cut of four to the dendrogram. The summary of the gearbox temperature, generator speed, wind speed and position deviation (difference in angle between the direction of the wind and the direction of the nacelle) can be seen in Table 7.2 in the previous cluster analysis in Section 7.1.2.

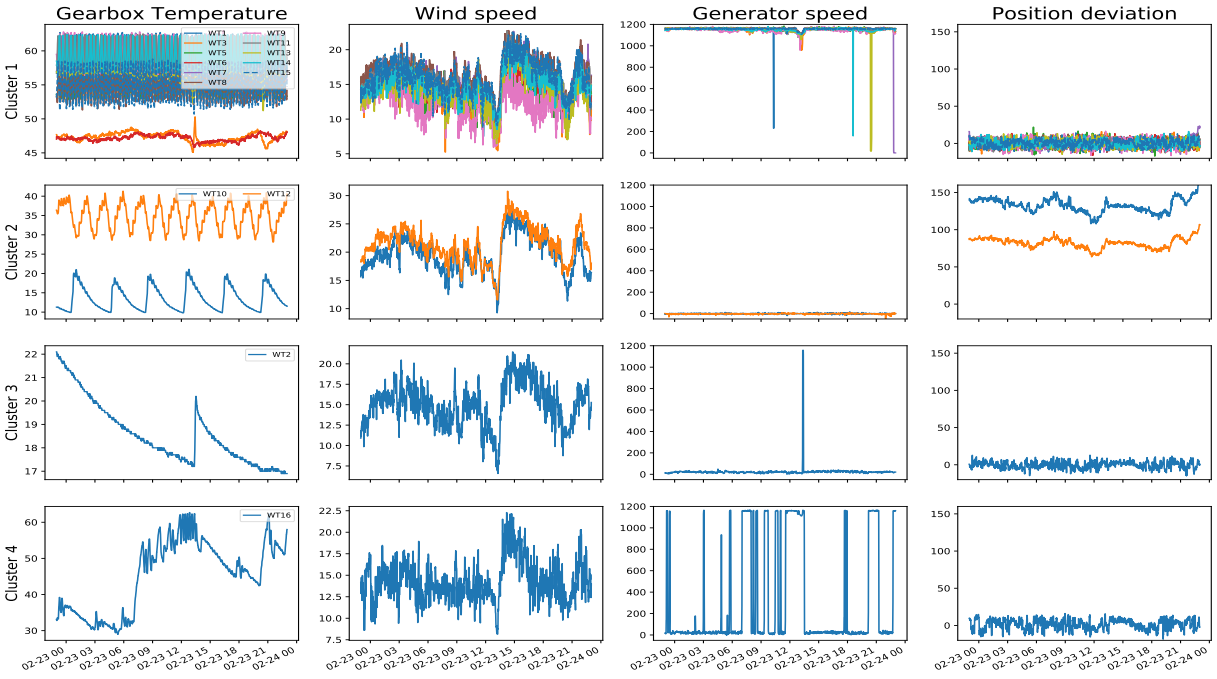


Figure 7.7: Clustering results from the 'Multivariate_V1' data set with DTW distance. Summary plot of a cut of four to the dendrogram with the linkage criterion set as 'average'.

Initially, one can observe that the turbines are separated well with respect to their overall similarity in shape. As with the objective of similarity in time, most of the dissimilarity between the clusters can be observed to be in the time series for the gearbox temperature. Cluster 1 can be observed to contain three (WT3, WT6, WT13) time series which are quite different in shape, compared to the rest. Same explanation for similarity in time holds here: The turbines have very similar shapes when reviewing the wind speed, generator speed and position deviation. Actually, cutting the current dendrogram a cut deeper (i.e. number of cuts equal to five) would separate the three turbines within cluster 1 and place them into its own separate cluster. However, this is not indicated by the silhouette index which suffers a quite significant reduction from a cut of four to a cut of five. More interestingly, the alpha values for weighting the different univariate data sets - which the multivariate data set is comprised of - can be modified. If we want stricter clustering of the gearbox temperature one could increase the corresponding alpha value associated with it. This is illustrated in Appendix B.3.3 and shows an optimal configuration which cuts the dendrogram into five partitions; this splits turbine 3, 6 and 13 from cluster 1 to an own separate cluster as discussed above. In other words, the cluster analysis becomes more similar to the univariate case where only the gearbox temperature is clustered. Now, the physical interpretation of the underlying turbines within each cluster - for the current cluster assignment seen in Figure 7.7 - will be further analysed in the next paragraphs. Much of the physical interpretation is similar to that of the univariate analysis in Section 6.2.4.

Cluster 1: Most of the turbines have a gearbox temperature which oscillates quite heavily with much higher frequency than the observed in the other clusters. These time series are characterised by high generator speeds and a position deviation of zero. As the majority of the clustered turbines is assigned to this cluster - and their behaviour is expected - these turbines are assumed to be operating under normal conditions. However, there are three outliers which can be observed in the plots for the gearbox temperature: Turbine 3, 6, and 13. These are very similar to the others with respect to all traits but the gearbox temperature; the time series for the gearbox does not experience similar oscillations. Oscillations are caused by the activation and deactivation of the cooling pump. The lack of oscillations might indicate some fault in the cooling system. A deeper cut to the dendrogram would separate these three from the current cluster, as mentioned in the previous paragraph.

Cluster 2: This cluster contains both turbine 10 and 12. These turbines are characterised by a gearbox temperature with a much lower frequency than those of cluster 1. The position deviation is nonzero - which means that the turbine is not directed against the wind - and the time series associated with the generator speed shows that the turbine is not rotating; further inspection shows that the generator is not rotating because of the extreme local wind conditions it experiences (and of course its alignment which is not directed against the wind direction).

Cluster 3: Contains only turbine 2 because the time series for the gearbox temperature is very different from the rest. Wind speed, generator speed and position deviation have similar behaviours. Reviewing the generator speed it becomes clear that this turbine is not rotating. Note that this turbine is under maintenance as mentioned in Section 6.2.4. Same physical interpretation as for the univariate case holds for the multivariate case.

Cluster 4: Contains only turbine 16 because the time series for the gearbox temperature and the generator speed is very different compared to the other turbines. The gearbox speed is varying quite heavily between the minimum and the maximum values; this results in an equally varying gearbox temperature. It can also be observed that the oscillations for the gearbox temperature occur only when the turbine is rotating. For further depth to the analysis, revisit the univariate cluster analysis, Section 6.2.4.

7.1.5 Comparison between the univariate and the multivariate data set

In this section, the clustering results from the univariate case (Section 6.2) will be compared to those of the multivariate case in the previous section. The univariate data set contains only the time series associated with the gearbox temperature; the multivariate data set is a combination of univariate data sets - where the univariate data set in Section 6.2 is one of the univariate data set which the multivariate data set is comprised of. The multivariate data set is also comprised of the univariate data sets: wind speed, generator speed and the position derivative. The measurements are extracted during the same interval and with the same parameters for extraction as for the 'Univariate_V2' data set.

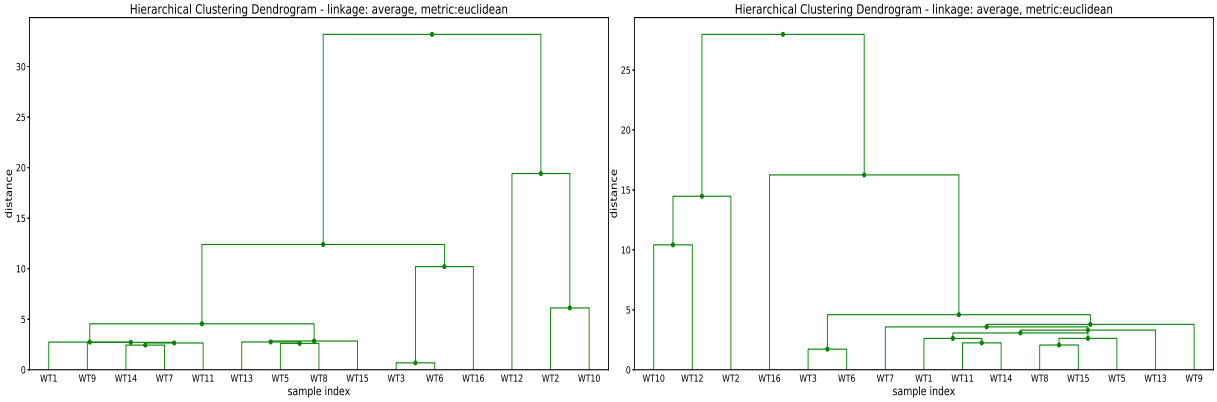
Similarity in time

The first observation to be made is that the internal indexes for both univariate and the multivariate case results in different linkage criterion, but recommend the same number of cuts to the associated dendrogram. The small difference in the internal indexes is expected as the univariate case is only dependent on the temperature in the gearbox, whereas the multivariate case, the clustering also needs to take into account the wind speed, the position deviation and the generator speed; a result of this is the assignment of turbine 3 and 6. In the univariate case, these two turbines are separated into its own separate cluster, but in the multivariate case, these are clustered together with the majority observed in cluster 4 in Figure 7.3. This is because the wind speed, generator speed and the position deviation are very similar to the other turbines in cluster 4; three out of the four univariate data sets indicate strongly that these turbines are similar to each other with respect to the objective of similarity in time. And for the univariate case, the gearbox temperature alone indicates that the turbines are quite dissimilar to the others. Furthermore, when comparing the dendrogram for the univariate analysis (Figure 6.23) and the multivariate analysis (Figure 7.2), a deeper cut of six to both dendrograms would result in the exact same assignment; turbine 3 and 6 are separated in the multivariate case and turbine 2 and 12 are separated in the univariate case.

The second observation to be made is that the multivariate case manages to separate the non-rotating turbines slightly better than in the univariate case (especially for a cut of two to the dendrograms). This is because the gearbox temperature is highly correlated to the generator speed; relatively low gearbox temperature along with a low generator speed is a strong indication of the first cluster formed in the multivariate case. Therefore, for a cut of two to the dendrogram, wind turbine 2, 10 and 12, are assigned to the same cluster. This is because the generator speed for these turbines is very similar. Because many of the time series associated with a dynamic process is highly correlated to each other, the separation between turbines will be more prominent if such time series are included. For instance, if the power production - which is highly correlated to the generator speed and gearbox temperature - is also included in the current multivariate analysis, the separation between rotating and non-rotating turbines would be more distinct. That is, if two turbines have similar gearbox temperatures, these two turbines would also have similar generator speed and power production (as long as there are no fault in the system). Approximately the same effect, as from including the power production time series, can be achieved by doubling the weighting value for either the gearbox temperature or the generator speed. This is, to some extent, analogous to including two time series (one additional) of the generator speed or gearbox temperature in the analysis. The weighting of the univariate data sets allows the user to focus on a specific trait. For instance, let us say for the gearbox temperature, it is much more critical to separate those experiencing abnormal temperature levels rather than those experiencing different wind speeds. Then, a high weighting value would, in this case, be given to gearbox temperature and a lower weighting value would be given to the wind speed parameter. Thus, the advantage of multivariate clustering, compared to univariate clustering, is that the significance of different parameters can be set according to preference and application. Of course, this comes at a cost. If more dimensions are included in the multivariate cluster analysis, the running time will increase along with it. However, this would not be the case if the corresponding alpha values are adjusted instead; the running time will remain the same.

The third observations can be made by reviewing the dendrograms for the univariate and the multivariate analysis. Both of the dendrograms - with average as the linkage criterion - for the univariate and the multivariate analysis is presented in Figure 7.8. The assignment of the turbines can be observed to be quite similar for both dendrograms. There are some minor differences between the assignment of turbine 2, 10, 12 and 16: For the univariate case, turbine 2 and 10 are more similar to each other than turbine 12 is to either of them; in the multivariate case, turbine 10 and 12 are more similar to each other than turbine 2 is to either of them. This can be justified by reviewing the time series for the wind speed, generator speed and position deviation in Figure 7.7. These three parameters - in a multivariate case - for turbine 10 and 12 are more similar to each other than that of turbine 2 is to either. For the univariate case, the similarity is only measured between the time series of the gearbox temperature. In that case, turbine 2 are more similar to turbine 10 than turbine 12 is to turbine 10. The assumption made in

the previous paragraph - that the multivariate case separates the rotating and non-rotating turbines better - can be verified by viewing the distance at where the three non-rotating turbines merge; the non-rotating turbines are more similar to each other in the multivariate, than in the univariate case (indicated by a smaller distance for the dendrogram in the multivariate case).



(a) Univariate analysis (Univariate_V2) **(b)** Multivariate analysis (Multivariate_V1)

Figure 7.8: Comparison between the dendrograms for similarity in time for both univariate and multivariate data set.

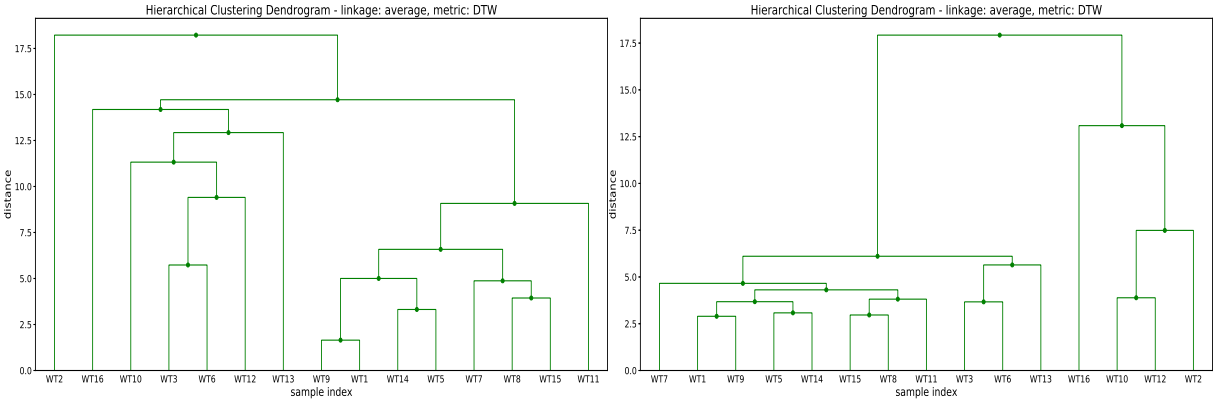
Similarity in shape

Now for the cluster analysis with the objective of similarity in shape. The first observation to be made is that the internal indexes for both univariate and the multivariate case results in same linkage criterion, but did not recommend the same number of clusters. The multivariate case finds the optimal number of clusters to be four, compared to seven for the univariate case. Both cluster analyses manage to separate the time series which are different in terms of shape but have slightly different clustering objectives. The univariate case cluster the turbines only based upon the different shapes in the gearbox temperature, and the multivariate case has a more general clustering objective which also takes into account other dimensions.

As for similarity in time, the dendrograms of the univariate and multivariate analysis are compared. Both the dendrograms - with average as the linkage criterion - for the univariate and the multivariate analysis is presented in Figure 7.9. At first glance, the dendrograms seem to be quite dissimilar to each other; however, further inspection shows that the clusters assignment for the dendrograms is indeed quite similar. The only significant difference can be observed in the assignment of turbine 3, 6 and 13. For the multivariate analysis, these turbines are clustered together along with the majority (see Figure 7.7) and are considered quite similar to those, but, for the univariate case, these are separated and considered quite dissimilar to any other turbines and to themselves. As with the objective of similarity in time, this can be justified by that the three turbines are very similar to the other when comparing the shape of the wind speed, generator speed and position deviation to the majority of the other turbines in a multivariate case. If only the gearbox temperature is compared - as in the univariate case - the three turbines would be quite dissimilar to the others and are therefore separated from the majority. Smaller differences can also be observed. For instance, the multivariate case states that turbine 10 and 12 are more similar compared to the univariate case. This is because these two turbines are slightly different in shape by comparing their gearbox temperature. But in terms of their wind speed, generator speed and position deviation, the shape/behaviour of the time series are close to identical. Clustering the turbines with respect to only one parameter would cluster the time series more naively. What is meant by a more naively, in this case, is that the assignment is only dependent on the shape of that specific univariate time series - this results in a more confined cluster problem which only takes into account the shape of that specific dimension (i.e. the gearbox temperature). By including more dimensions (or parameters of interest) - as done in the multivariate case - we would acquire a more general cluster assignment which is not only based upon the behaviour of one parameter but several.

(repeated experiment 7.1.3)

As mentioned when analysing the clustering results from the objective of similarity in time, the different univariate time series which the multivariate time series is comprised of, can be weighted differently. This can be done by adjusting the corresponding alpha value for each of the univariate data sets. In Appendix B.3.3, the cluster analysis of similarity in shape was repeated with modifications to the alpha vector (weighting vector) - same cluster analysis as in Section 7.1.3, only with a different alpha vector. In this cluster analysis, all alphas



(a) Univariate analysis (Univariate_V2) **(b)** Multivariate analysis (Multivariate_V1)

Figure 7.9: Comparison between the dendrograms for similarity in shape for both univariate and multivariate data set.

were set equal to one except for the alpha corresponding to the gearbox temperature which was set to eighth. This increases the significance of the dissimilarities in the gearbox temperature by a factor of eighth. The internal indexes could be compared to the univariate case where only the gearbox temperature is clustered; it shows that by increasing the corresponding gearbox temperature, the internal indexes become more and more similar to that of the univariate case. For this configuration, the optimal number of clusters was found to be equal to five. However, the difference in the internal indexes is not that significant when comparing a cut of five to a cut of six, or even seven. Arguably, a cut of seven could also reflect the optimal number of cuts to the dendrogram. Actually, a cut of seven separates the time series into the exact same cluster assignment as for clustering the univariate time series. The example illustrates how the alphas could be manipulated in order to customise the clustering algorithm to the specific application. As the alpha is increased further, and the alpha corresponding to the other univariate data sets remains small, the multivariate case would give the exact same results as the univariate case.

The average within-cluster variance (or the MSSSE value) can be observed to be slightly lower in the multivariate case than in the univariate case (this can be done by comparing the internal indexes in Table 6.6 and 7.3). The explanation for this is that when clustering the multivariate case, the algorithm also takes into account the other parameters, such as the wind direction, generator speed and position deviation. The wind speed, generator speed (and position deviation) have very similar shapes for all time series and therefore would result in a low within-cluster variance for each of them. The average within-cluster variance for the multivariate case is basically the summation of the within-cluster variance for each dimension divided by the number of dimensions in the data set. If three of the dimensions has a relatively low within-cluster variance and the fourth dimension has a relatively high within-cluster variance (i.e. the gearbox temperature), then the average within-cluster variance would alwas be lower than the dimension which has the largest within-cluster variance. Therefore, the within-cluster variance (or the MSSSE index) for the univariate case is larger than the multivariate case. Additionally, it is worth noting that the MSSSE value is only summarised if the corresponding alpha is nonzero; the number of dimensions which the sum is divided by - to get the average values - corresponds to the sum of all alphas in the alpha vector. If the alpha for the corresponding univariate data set is zero, then it is the same as not including the time series in the analysis at all.

A final remark which requires some attention is the running time of the DTW algorithm. From the univariate case in Section 6.2.3, the calculation of the distance matrix - where the algorithm was constrained with the Sakeo-Chiba band - took close to 7 minutes. Calculation of the distance matrix in the multivariate case increased to 29 minutes, which is roughly 4 time as much (the number of dimensions in the data set). The running time for the DTW algorithm, with global restriction, has increased quite drastically. The next multivariate data set which will be analysed is even larger than the one analysed in this section. The calculation of the distance matrix, with the DTW distance constrained, is expected to take several hours. Further comparison between the running time will be made after the analysis of the second and last multivariate data set in the next section.

7.2 Clustering of 'Multivariate_V2'

The second and last multivariate data set which is going to be clustered is the 'Multivariate_V2' data set introduced in Section 5.2.2 and visually presented in Appendix A.3. The interval of the new data set is increased. This is done in order to compare different lengths and find a proper interval to cluster; a proper interval should be large enough to avoid having unobserved dynamics and not too large to avoid superposition of different dynamics. Too short intervals have the risk that relevant dynamics of the wind turbines may be unobserved. Too long intervals include these dynamics but may include too much and therefore being uninformative. After counselling with the supervisors, an interval of 1 month was chosen. However, because of insufficient measurements in the software, the interval found which was closest to one month was an interval of 22 days. The extracted data set is presented in the aforementioned section, Section 5.2.2. In this section, the same procedure as for the previous data sets will be followed. In Section 7.2.1, the clustering results with the objective of similarity in time will be presented. The analysis of the clustering results will be presented in the following section, Section 7.2.2. The same layout will be used when clustering the data set with the objective of similarity in shape. The clustering results is presented in Section 7.2.3 followed by an analysis of the results in Section 7.2.4.

7.2.1 Similarity in time - Results from hierarchical clustering of the scaled data set

First of all, the data set will be clustered with the objective of similarity in time. The filtered data set contains features which are of different units of measurements. This makes adding together dissimilarity matrices inconvenient. Before applying any kind of clustering, the time series has to be scaled. For convenience, the univariate data sets must be scaled individually so that the minimum and maximum values for each univariate data set are 0 and 1, respectively. Scaling the data removes the fact that the variables are in different units of measure. As all the univariate data sets - which the multivariate data set is comprised of - are individually scaled, clustering of the data set can begin. As in Section 5.2.1, the best linkage criterion for hierarchical clustering algorithm will be found by comparing their *Cophenetic correlation coefficient* and the number of cuts to the dendrogram will be determined by viewing the internal performance indexes: *Silhouette index* and *MSSSE* index. More information about these specific indexes can be found in Chapter 3. The plots of the univariate data sets which the current multivariate data set are comprised of can be seen in Appendix A.3. After reviewing the time series corresponding to the external temperatures in Figure A.3.6, it can be observed that it contain several measurements which are obviously invalid. For instance, turbine 12 has an outside temperature which varies between -200 and 200 degrees Celsius. The other turbines which do not contain valid measurements are turbine 3 and 13. Clustering the entire multivariate data set, which also includes the external temperature, would yield invalid clustering results. Therefore, the external temperature is removed from the multivariate data set. Now the data set has the shape $(5, 15, 6295)$, rather than $(6, 15, 6295)$. In this section, two scenarios will be analysed. Scenario one will be a cluster analysis of the entire data set where all alphas are equal to one except for the alpha corresponding to the external (outside) temperature which is set to zero (setting the alpha to zero is the same as excluding it from the cluster analysis). The second scenario will be a cluster analysis where all alphas are equal to one except for the one corresponding to the external temperature and the wind direction which is set to zero. Justification for the latter will be done after presenting the results of the former.

Scenario 1: External temperature disregarded

For this analysis, the alphas are all equal to one except for the alpha corresponding to the external (outside) temperature which is set to zero. This has the same effect as not including the external temperature at all in the cluster analysis. In other words, the shape of the multivariate data set analysed is now $(5, 15, 6295)$ instead of $(6, 15, 6295)$. The summary plot and table for the internal indexes of cutting each dendrogram into *k* partitions - each with different linkage criterion - can be seen in Figure 7.10 and Table 7.4. The cophenetic correlation coefficient can be seen in the upper right corner in the aforementioned figure (Figure 7.10).

First of all, reviewing the cophenetic correlation coefficient in Figure 7.10, it can be seen that all dendrograms experience similar high values, all above 0.9. The cophenetic correlation coefficient is close to 1 for all dendrogram; values close to 1 indicate that all the dendrograms represent a high-quality solution. The dendrogram with the highest cophenetic index is the dendrogram constructed with the linkage criterion set as *single*. Therefore, the dendrogram built with linkage criterion as *single* will be cut and its assignment will be analysed. Secondly, reviewing the summary table and the corresponding summary plot, it can be observed that the silhouette index is largest for a cut of three and slightly smaller for the neighbouring cuts, two and four. The corresponding MSSSE value for a cut of two to three decreases exponentially from a value of 224.94 at a cut of two to 131.59 at a cut

Table 7.4: Summary table with the linkage criteria: single, complete, average and ward. The data set which was clustered is 'Multivariate_V2' with the objective of similarity in time. The silhouette indexes and MSSSE values for a cut K - from 2 to 13 - are presented. The alphas are all equal to one except for the alpha corresponding to the external (outside) temperature which is set to zero.

Method	Internal index	Number of cuts to the dendrogram											
		K = 2	K = 3	K = 4	K = 5	K = 6	K = 7	K = 8	K = 9	K = 10	K = 11	K = 12	K = 13
Single	Silhouette	0.43	**0.45**	0.37	0.18	0.20	0.17	0.14	0.12	0.10	0.11	0.09	0.05
	MSSSE	224.94	**131.59**	118.06	97.36	51.09	41.78	30.26	22.37	17.58	12.07	9.08	4.30
Complete	Silhouette	0.47	0.45	0.22	0.18	0.20	0.16	0.14	0.12	0.11	0.07	0.07	0.05
	MSSSE	173.33	131.59	110.89	97.36	51.09	39.57	30.26	22.37	16.96	12.19	7.30	4.30
Average	Silhouette	0.47	0.45	0.22	0.18	0.20	0.16	0.14	0.12	0.11	0.11	0.07	0.05
	MSSSE	173.33	131.59	110.89	97.36	51.09	39.57	30.26	22.37	16.96	12.07	7.30	4.30
Ward	Silhouette	0.47	0.45	0.26	0.24	0.20	0.17	0.14	0.12	0.11	0.11	0.07	0.05
	MSSSE	173.33	131.59	80.52	64.63	51.09	41.78	30.26	22.37	16.96	12.07	7.30	4.30

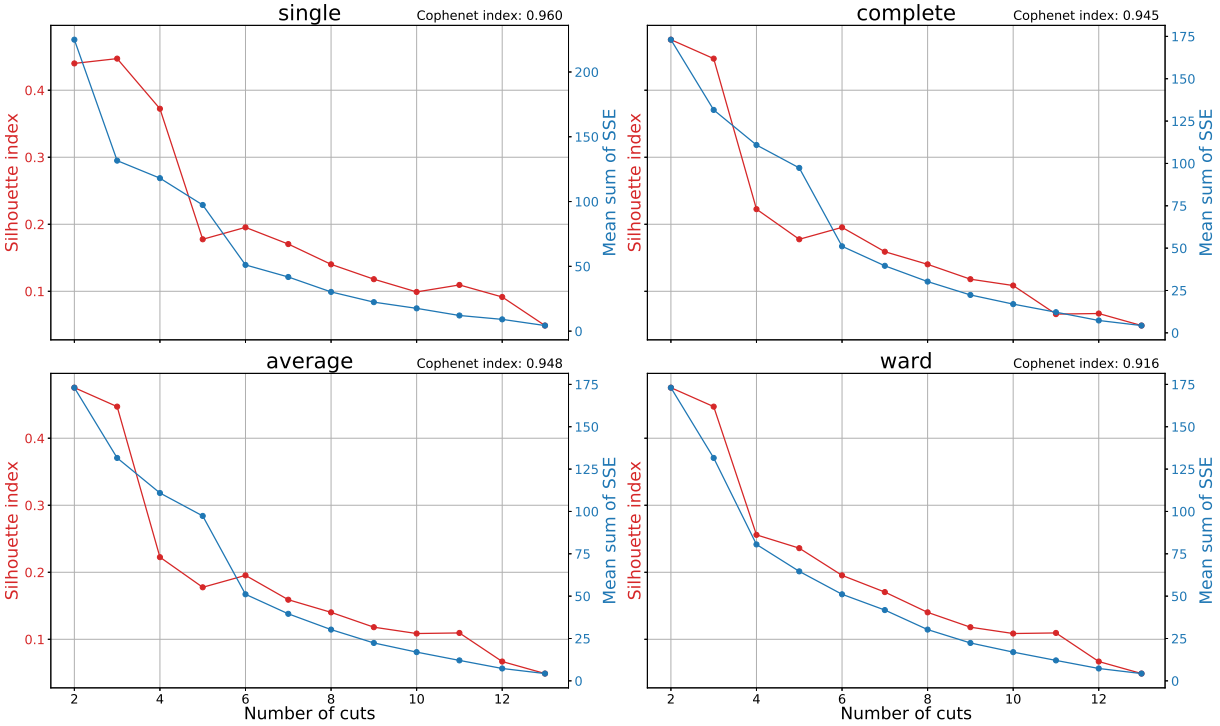


Figure 7.10: Summary plots with the linkage criteria: single, complete, average and ward. The data set which was clustered is 'Multivariate_V2' with the objective of similarity in time. The silhouette indexes and MSSSE values for a cut K - from 2 to 13 - is presented along with the cophenetic correlation coefficient in the upper right corner. The alphas are all equal to one except for the alpha corresponding to the external (outside) temperature which is set to zero.

of three. No significant reduction in the MSSSE value is observed for a larger number of cuts; a larger number of cuts also experience quite a significant decrease in its silhouette index value. Therefore, a cut of three seems to be the optimal configuration where the compromise between separation and intra-cluster similarity is at its best. The corresponding dendrogram - with *single* as linkage criterion and the external temperature excluded - can be seen in Figure 7.11.

Furthermore, now the corresponding dendrogram in Figure 7.11 will be cut. From analysing the internal indexes in Figure 7.10, the optimal number of cuts to the dendrogram was found to be three. In Figure 7.12, the cluster assignment along with the time series are visualised for a cut of three. Note that the different clusters now spans the columns of the subplot and not the rows as in Section 7.1. Each row spans the different univariate data sets. For instance, the first row shows the power production of each wind turbine, all of which assigned to a specific cluster. As can be observed in Figure 7.12, a cut of three to the dendrogram manages to separate the three most dissimilar turbines from the majority. The within-cluster variance for each cluster can be observed to be relatively small for all clusters and parameters, except for the time series describing the wind direction. The

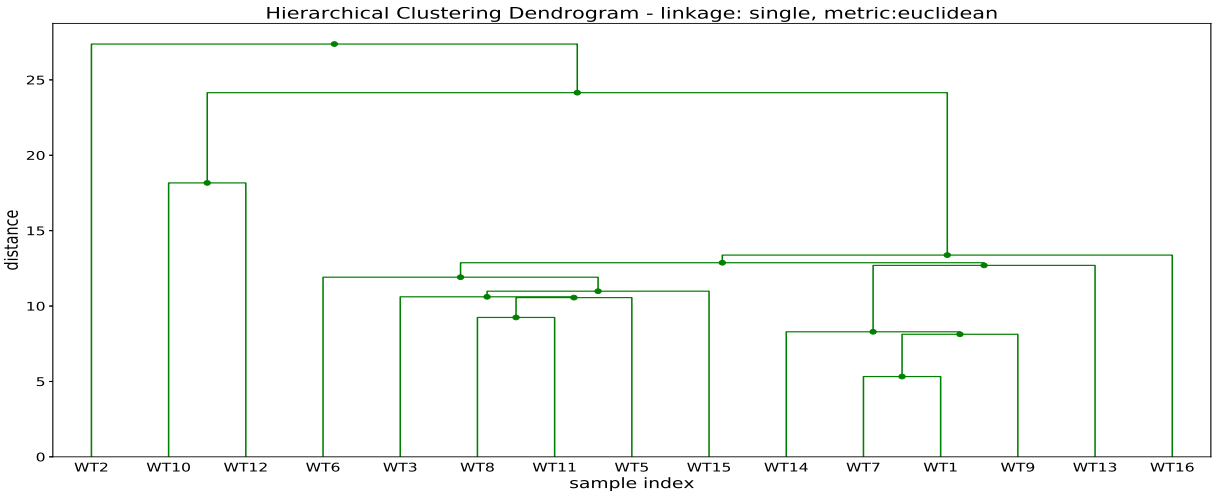


Figure 7.11: Dendrogram with *single* as linkage criteria and distance as Euclidean distance implemented on 'Multivariate_V2' data set. The alphas are all equal to one except for the alpha corresponding to the external (outside) temperature which is set to zero.

time series for the wind direction are not consistent with the cluster assignments. For example, the time series for the wind direction in cluster 2 is not really similar to each other with regards to the objective of similarity in time. Actually, comparing the wind direction to those of clusters 1 and 3, no significant differences can be observed between them. Because the time series for the wind directions are so different from each other, including them would results in a large within-cluster variance (i.e. a large MSSSE value), regardless of the cluster the time series is assigned to. By including the wind direction when doing the cluster analysis, it could negatively impact clusters of larger size/entries. Therefore, in the next section, the cluster analysis is repeated on the multivariate data set where both the external temperature and the wind direction are excluded from the analysis (i.e. setting the corresponding alphas to zero).

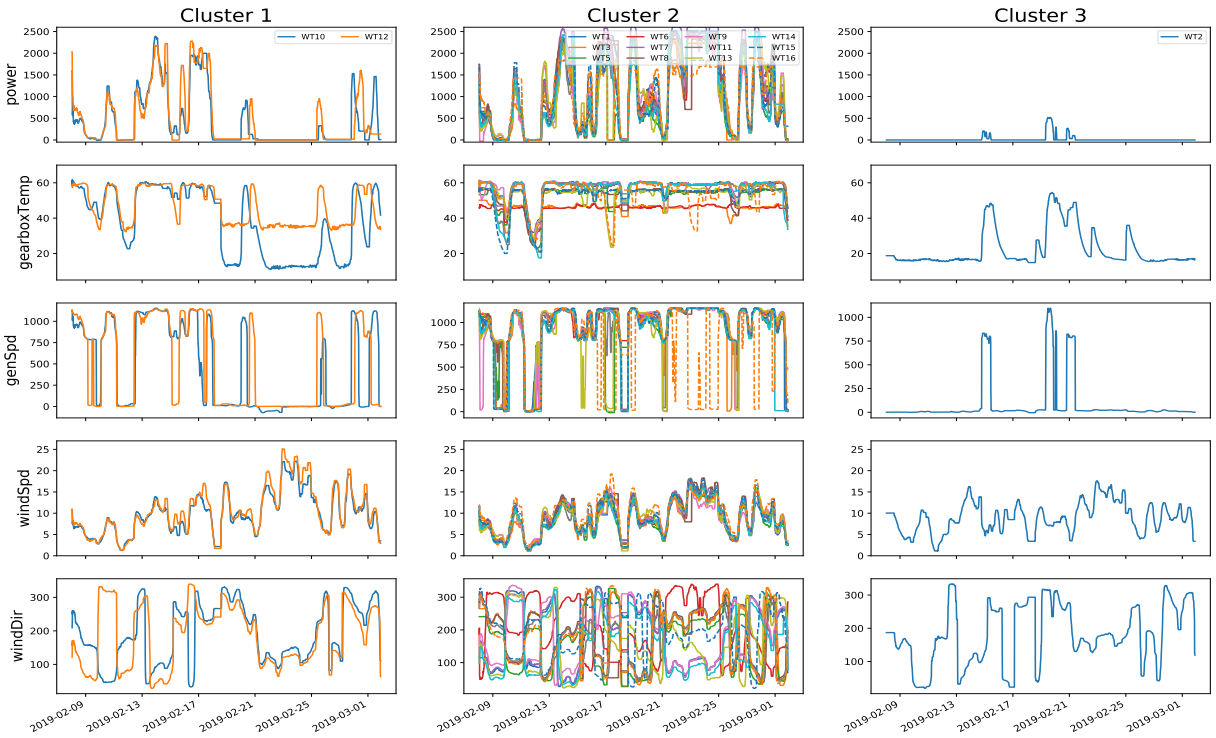


Figure 7.12: Clustering results from the 'Multivariate_V2' data set with Euclidean distance. Summary plot of a cut of three to the dendrogram with the linkage criterion set as 'single' (from Figure 7.11)

Scenario 2: External temperature and the wind direction disregarded

For this cluster analysis, the alphas are all equal to one except for the alphas corresponding to the external (outside) temperature and the wind direction which is set to zero. This has the same effect as not including the external temperature or the wind direction at all in the cluster analysis. In other words, the shape of the multivariate data set analysed is now $(4, 15, 6295)$ compared to $(5, 15, 6295)$ in the previous scenario. The summary plot and table for the internal indexes of cutting each dendrogram into k partitions - each with different linkage criterion - can be seen in Figure 7.13 and Table 7.5. The cophenetic correlation coefficient can be seen in the upper right corner in the aforementioned figure (Figure 7.13).

Table 7.5: Summary table with the linkage criteria: single, complete, average and ward. The data set which was clustered is 'Multivariate_V2' with the objective of similarity in time. The silhouette indexes and MSSSE values for a cut K - from 2 to 13 - are presented. The alphas are all equal to one except for the alpha corresponding to the external (outside) temperature and wind direction which is set to zero.

Method	Internal index	Number of cuts to the dendrogram											
		K = 2	K = 3	K = 4	K = 5	K = 6	K = 7	K = 8	K = 9	K = 10	K = 11	K = 12	K = 13
Single	Silhouette	0.51	0.55	0.47	0.29	0.13	0.07	0.10	0.07	0.11	0.08	0.07	0.03
	MSSSE	180.22	73.91	61.49	42.00	33.07	29.18	20.34	16.10	10.38	8.03	5.55	3.21
Complete	Silhouette	0.56	0.55	0.34	0.29	0.13	0.14	0.14	0.13	0.11	0.08	0.05	0.04
	MSSSE	120.55	73.91	54.42	42.00	33.07	24.23	17.70	13.81	10.38	8.03	6.25	3.77
Average	Silhouette	0.51	**0.55**	0.47	0.29	0.13	0.14	0.10	0.13	0.11	0.08	0.07	0.03
	MSSSE	180.22	**73.91**	61.49	42.00	33.07	24.23	20.34	13.81	10.38	8.03	5.55	3.21
Ward	Silhouette	0.56	0.55	0.34	0.29	0.13	0.14	0.14	0.13	0.11	0.08	0.07	0.03
	MSSSE	120.55	73.91	54.42	42.00	33.07	24.23	17.70	13.81	10.38	8.03	5.55	3.21

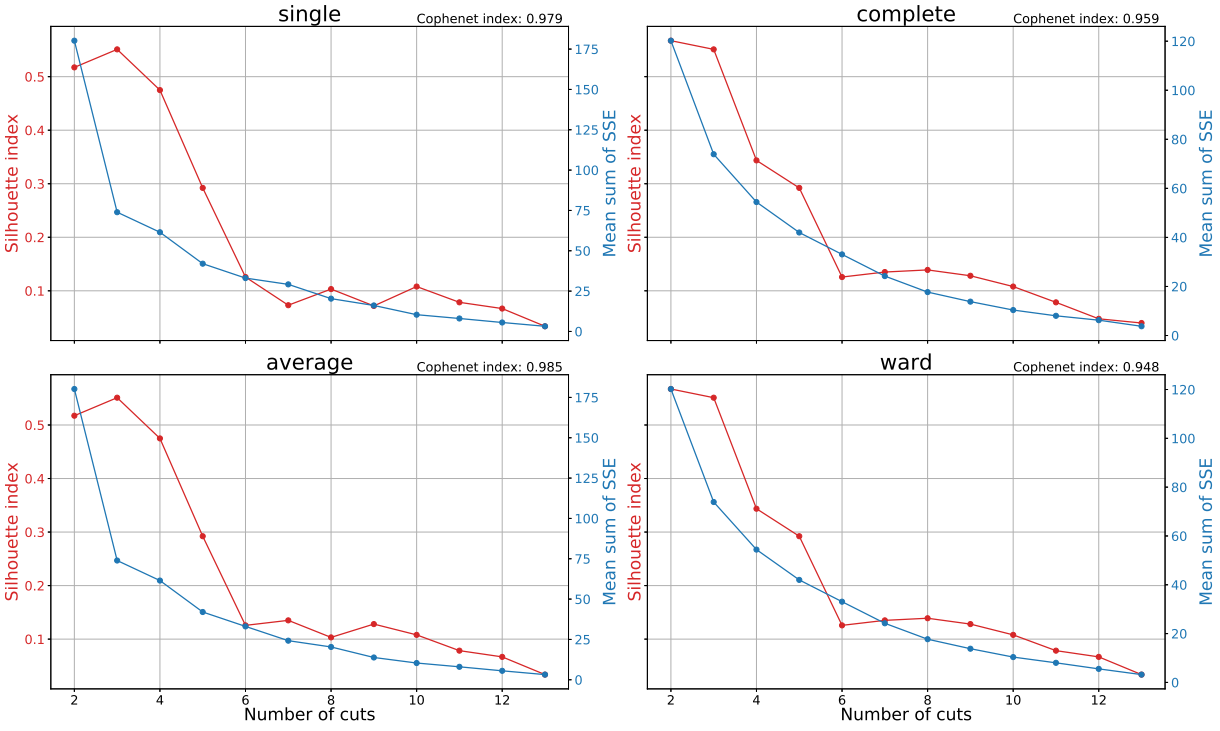


Figure 7.13: Summary plots with the linkage criteria: single, complete, average and ward. The data set which was clustered is 'Multivariate_V2' with the objective of similarity in time. The silhouette indexes and MSSSE values for a cut K - from 2 to 13 - is presented along with the cophenetic correlation coefficient in the upper right corner. The alphas are all equal to one except for the alpha corresponding to the external (outside) temperature and the wind direction which is set to zero.

First of all, reviewing the cophenetic correlation coefficient in Figure 7.13, it can be seen that all dendrograms experience similar high values, all above or around 0.95. The cophenetic correlation coefficient is close to 1 for all dendrogram; values close to 1 indicate that all the dendrograms represent a high-quality solution. The

dendrogram with the highest cophenetic index is the dendrogram constructed with the linkage criterion set as *average*. Therefore, the dendrogram built with linkage criterion as *average* will be cut and its assignment will be analysed. Secondly, reviewing the summary table and the corresponding summary plot, it can be observed that the silhouette index is largest for a cut of three and slightly smaller for the neighbouring cuts, two and four. The corresponding MSSSE value from a cut of two to three decreases exponentially from a value of 180.22 at a cut of two to 73.91 at a cut of three. No significant reduction in the MSSSE value is observed for a larger number of cuts; a larger number of cuts also experience a quite significant decrease in its silhouette index value. Therefore, a cut of three seems to be the optimal configuration where the compromise between separation and intra-cluster similarity is at its best. The corresponding dendrogram - with *average* as the linkage criterion and the external temperature and wind direction excluded - can be seen in Figure 7.14.

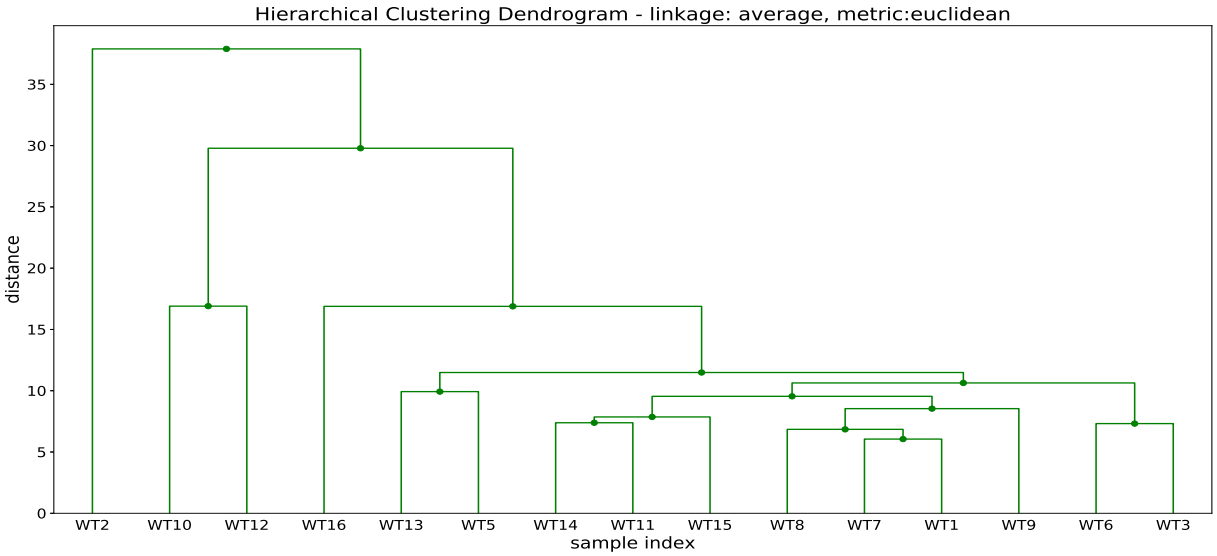


Figure 7.14: Dendrogram with *average* as the linkage criterion and distance as Euclidean distance implemented on 'Multivariate_V2' data set. The alphas are all equal to one except for the alpha corresponding to the external (outside) temperature and the wind speed which is set to zero.

From observing the dendrogram, one can see that the exact same cluster assignment is obtained by removing the wind direction from the analysis. When comparing the internal indexes from scenario 1 and scenario 2, a significant change in the internal indexes can be observed. If one of the univariate data sets which are associated with a relatively large withing-cluster variance is removed, then the average within-cluster variance for each data set across all clusters would be reduced. This is exactly what is observed between the two scenarios by comparing their internal indexes. The corresponding silhouette index has now changed from 0.45 to 0.55 and the corresponding MSSSE value from 131.59 to 73.91. This is a strong indication that by including the wind direction data set resulted in significant within-cluster dissimilarities (this can also be observed by looking at the time series assigned to each cluster in Figure 7.15). If one of the univariate data sets which are associated with a relatively large withing-cluster variance is removed, then the average within-cluster variance for each data set across all clusters would be reduced. Regardless of clustering scenario 1 or 2, the same cluster assignment was achieved and will be further analysed in the consecutive section.

7.2.2 Analysing the clustering results from similarity in time

In this section, the clustering results from clustering 'Multivariate_V2' data set in Section 7.2.1 (scenario 2) - where the external temperature and the wind direction is excluded - will be analysed. The corresponding dendrogram which will be cut can be seen in Figure 7.14. From analysing the internal indexes from the previous section, a cut of three to the dendrogram was found to be the optimal number of cuts. In Figure 7.15 the cluster assignment and the time series are visualised for a cut of three. Note that the different clusters now spans the columns of the subplot and not the rows as in the analysis of the clustering results in Section 7.1. Each row spans the different univariate data sets. For instance, the first row shows the power production of each wind turbine; all of which assigned to a specific cluster.

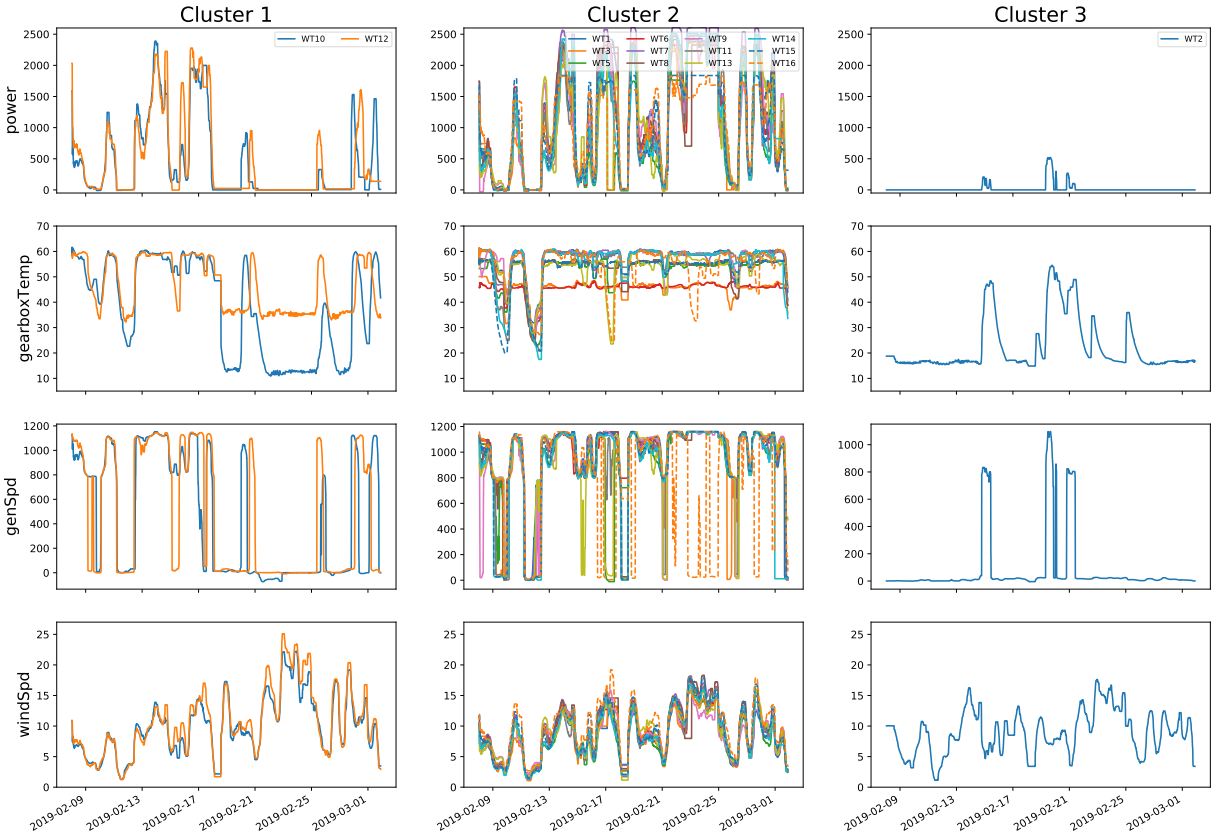


Figure 7.15: Clustering results from the 'Multivariate_V2' data set with Euclidean distance. Summary plot of a cut of three to the dendrogram with the linkage criterion set as 'average' (from Figure 7.14)

Initially, it can be observed that the cutting the dendrogram in three, it manages to separate the three most dissimilar time series from the majority. The within-cluster variation within each cluster can be observed to be relatively small for all clusters. There are some time series within cluster 2 which are more dissimilar than the majority. Visually, turbine 3 (orange), 6 (red) and 16 (orange dashed lines) can be observed to deviate to a certain degree from the rest: Turbine 3 and 6 deviates only in time series for the gearbox temperature; turbine 16 deviates (to a small degree) from the majority with respect to all its parameters. Reviewing the dendrogram (Figure 7.14), it can be observed that for a deeper cut of four to the dendrogram, it would either split the current cluster 1 in two or separate turbine 16 from cluster 2 (it is hard to say because the horizontal lines have close to identical height). Either way, a cut of five would both split the current cluster 1 in two and separate turbine 16 from cluster 2. However, reviewing the internal indexes for the analysis (Table 7.5) the silhouette index or the MSSSE value does not recommend a deeper cut than three. Therefore, the assignment for a cut of three will be further analysed.

Cluster 1: Turbines assigned to this cluster is turbine 10 and 12. Both turbines experience similar behaviour for the power production (i.e. quite varying levels of power production). A strong correlation between the time series for the power production, gearbox temperature and generator speed can be observed: A low generator speed results in a low power production and a reduction in the gearbox temperatures. The physical interpretation of the sudden

halt in generator speed can be observed by viewing the wind speeds for both turbines. During the second half the interval both of the turbine experience relatively large wind speeds. These turbines are subjected to rougher wind conditions than the rest and ultimately results in engaging the brakes more frequently. These turbines can actually be observed to be close to identical to those of cluster 2 during the first half of the interval, but very dissimilar to the others when reviewing the second half of the interval.

Cluster 2: The majority of the turbines are associated with this grouping. The turbines associated with this grouping is characterised by elevated levels of power production as the turbines have more favourable wind condition. More favourable wind conditions result in a more consistence generator speed and power production for the individual turbines. These turbines associated with this group can be assumed to be operating under normal conditions and deviations from these should be inspected further.

Cluster 3: Groupings 3 contains only turbine 2. Turbine 2 is characterised by low to no power production and generator speed. This is also transparent in the gearbox temperature, as the aforementioned time series are highly correlated to each other. The first assumption is that the turbines experience elevated wind conditions which result in the brakes being engaged. However, the wind speeds are similar to that of the turbines in cluster 2 and no wind speeds of critical levels are observed. The lack of power production is not explained by reviewing any of the time series in the data set. However, reviewing the maintenance log of the corresponding wind park it becomes clear that turbines 2 have been stopped manually due to maintenance. Even though the physical behaviour is not explained by the information gathered, the cluster analysis still manages to separate it from the rest as the behaviour is highly dissimilar to the others.

7.2.3 Similarity in shape - Results from hierarchical clustering of the normalised data set

The objective is now to cluster the time series with similarity in shape and not the absolute difference and offset. Therefore, the clustering analysis is performed on the normalised data set. Because of the invalid measurements of the external temperature, the corresponding univariate data set will be excluded from the cluster analysis in this section. The other time series associated with 'Multivariate_V2', after preprocessing, can be found in Appendix A.3. None of these time series for any of the univariate data sets experiences flat and noisy behaviour (as in Section 7.1.3 for the generator speed). The time series can, therefore, be normalised prior to clustering with the objective of similarity in shape. For similarity in shape, the DTW distance (3.7) measure is used instead of the Euclidean distance. Same constraint as in Section 6.1.4, 6.2.3 and 7.1.3 is applied to the DTW algorithm. DTW has much higher complexity then Euclidean distance and the calculation of the condensed distance matrix took now close to two hours. The condensed distance matrix is then fed to the hierarchical clustering algorithm. As before, the best linkage criterion for the hierarchical clustering algorithm and the optimal number of clusters will be found with the same internal indexes used so far: Cophenetic correlation coefficient, Silhouette index and MSSSE. More information about these specific indexes can be found in Chapter 3. The summary plot and table for the internal indexes of cutting each dendrogram - each with different linkage criterion - into k cuts can be seen in Figure 7.16 and Table 7.6. The cophenetic correlation coefficient can be seen in the upper right corner in the aforementioned figure (Figure 7.16). Because we do not wish to include the external temperature (obviously invalid/wrong measurement) the corresponding alpha is set to zero, disregarding the measurements completely. The alpha corresponding to the wind direction is kept as 1, as it still might be similar in shape.

Table 7.6: Summary table with the linkage criteria: single, complete, average and ward. The data set which was clustered is 'Multivariate_V2' with the objective of similarity in shape. The silhouette indexes and MSSSE values for a cut K - from 2 to 13 - are presented. The alphas are all equal to one except for the alpha corresponding to the external (outside) temperature which is set to zero.

Method	**Internal index**	**Number of cuts to the dendrogram**											
		K = 2	K = 3	K = 4	K = 5	K = 6	K = 7	K = 8	K = 9	K = 10	K = 11	K = 12	K = 13
Single	Silhouette	0.51	**0.57**	0.33	0.27	0.15	0.08	0.20	0.16	0.13	0.10	0.09	0.08
	MSSSE	7.99	**5.76**	4.84	4.25	3.70	3.15	2.14	1.72	1.35	1.02	0.69	0.39
Complete	Silhouette	0.60	0.57	0.33	0.27	0.20	0.21	0.20	0.16	0.13	0.10	0.09	0.08
	MSSSE	7.01	5.76	4.84	3.71	3.11	2.57	2.14	1.72	1.35	1.02	0.69	0.39
Average	Silhouette	0.60	0.57	0.33	0.22	0.15	0.21	0.20	0.16	0.13	0.10	0.09	0.08
	MSSSE	7.01	5.76	4.84	4.30	3.70	2.57	2.14	1.72	1.35	1.02	0.69	0.39
Ward	Silhouette	0.60	0.57	0.25	0.28	0.28	0.21	0.20	0.16	0.13	0.10	0.09	0.08
	MSSSE	7.01	5.76	4.52	3.83	3.17	2.57	2.14	1.72	1.39	1.02	0.69	0.39

First of all, reviewing the cophenetic correlation coefficient in Figure 7.16, it can be seen that all dendrograms experience similar high values; most of the dendrograms have high values equal or above 0.95. The cophenetic correlation coefficient is very close to 1 and well above 0.75 which indicates that the dendrograms represent a high-quality solution. The dendrogram with the largest cophenetic index is the dendrogram constructed with the linkage criterion set as *single*. Therefore, the dendrogram built with the linkage criterion as *single* will be cut and its assignment will be analysed. Secondly, reviewing the summary table and the corresponding summary plot, it can be observed that the silhouette index is largest for a cut of three and decreases exponentially for deeper cuts; the corresponding MSSSE value has a relatively steep descend from a cut of two to a cut of three, but for deeper cuts it decreases more slowly. A cut of three seems to be the optimal configuration where the compromise between separation and intra-cluster similarity is at its best. Therefore, a cut of three to the dendrogram will be further analysed in the consecutive section. The corresponding dendrogram - with *single* as the linkage criterion and the external temperature excluded - can be seen in Figure 7.17. As the Silhouette index and the MSSSE values are identical for a cut of three to all dendrograms, one could assume that the different dendrograms produce the same cluster assignments.

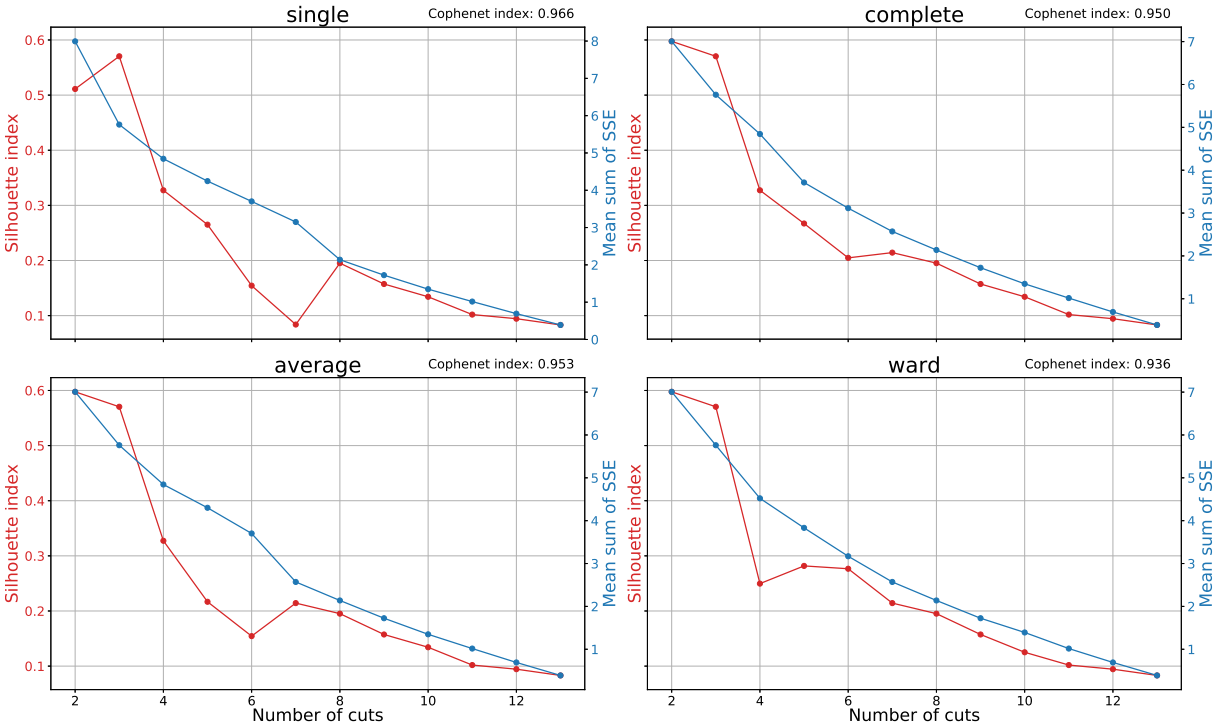


Figure 7.16: Summary plots with the linkage criteria: single, complete, average and ward. The data set which was clustered is 'Multivariate_V2' with the objective of similarity in shape. The silhouette indexes and MSSSE values for a cut K - from 2 to 13 - is presented along with the cophenetic correlation coefficient in the upper right corner. The alphas are all equal to one except for the alpha corresponding to the external (outside) temperature which is set to zero.

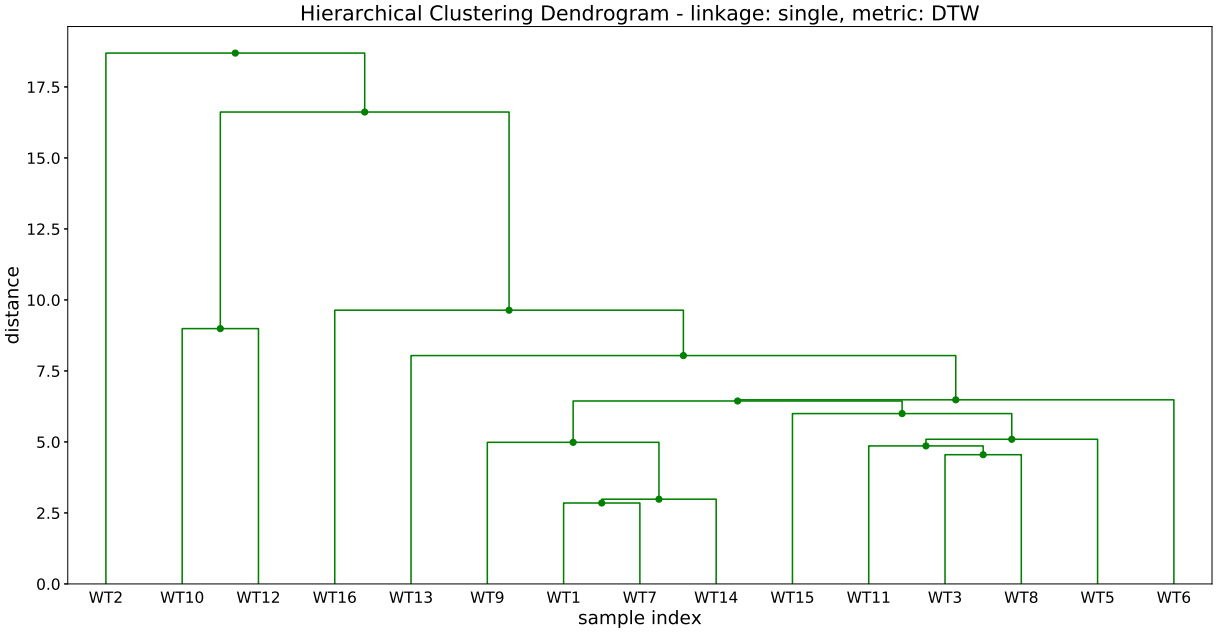


Figure 7.17: Dendrogram with *single* as the linkage criterion and distance DTW distance implemented on 'Multivariate_V2' data set. The alphas are all equal to one except for the alpha corresponding to the external (outside) temperature which is set to zero.

7.2.4 Analysing the clustering results from similarity in shape

In this section, the clustering results from clustering 'Multivariate_V2' data set in Section 7.2.3 - where the external temperature is excluded - will be analysed. The corresponding dendrogram which will be cut can be seen in Figure 7.17. In Figure 7.18 the cluster assignment and the time series are visualised for a cut of three.

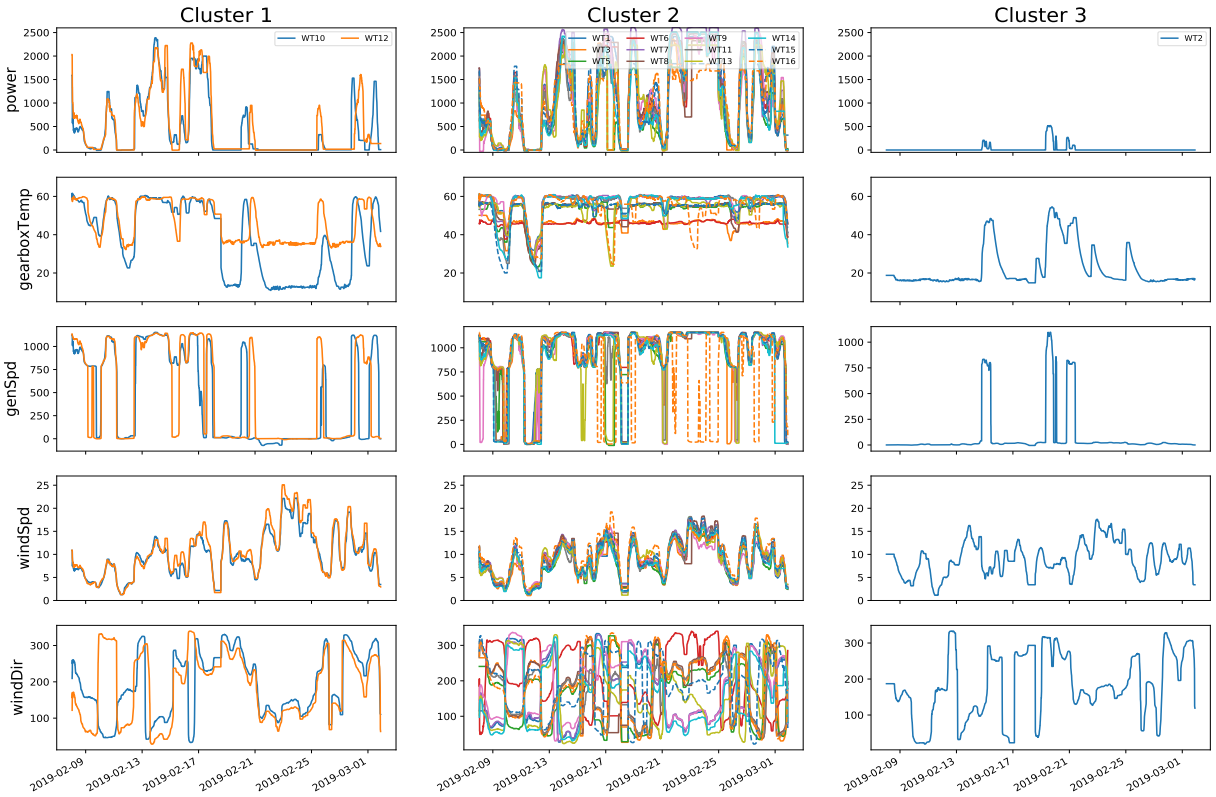


Figure 7.18: Clustering results from the 'Multivariate_V2' data set with DTW distance. Summary plot of a cut of three to the dendrogram with the linkage criterion set as 'single' (from Figure 7.17)

Initially, it can be observed that a cut of three to the dendrogram results in the exact same cluster assignment as for similarity in time analysis. It manages to separate the three most dissimilar time series from the majority. Because the cluster assignment is the same for both objectives, the clusters formed are both similar in time and similar in shape. The same physical interpretation as for similarity in time in Section 7.2.2 holds for this analysis. Additional physical interpretation from clustering with the objective of similarity in shape is not transparent.

Furthermore, the similarity in shape analysis was repeated on the data set for scenario 2, described in the similarity in time analysis. In scenario 2, the time series for the wind direction were also excluded, in addition to excluding the time series for the external temperature. The clustering results can be seen in Appendix B.4.3. The internal indexes find the best dendrogram and the optimal number of cuts to the dendrogram to be the same as in this section. As with the objective of similarity in time, an improvement in the internal indexes was observed. The silhouette index increases from 0.57 to 0.65 and the MSSSE index decreases from 5.76 to 4.26. This indicates that the inclusion of the wind direction in the analysis resulted in a significant increase in the within-cluster variance for each cluster.

7.3 Summary and comparison

So far in this chapter, two multivariate data set, namely, 'Multivariate_V1' and 'Multivariate_V2', have been clustered and the corresponding clustering results have been analysed. The clustering results and analysis for both data sets have been performed in Section 7.1 and 7.2, respectively. The major difference between the data sets is the interval of extraction where the latter have measurements extracted over a period of 22 days, in contrast to the former where the interval was only 1 day long. As a consequence, the resampling and aggregation intervals are different for both extractions. The experiment of clustering the second data set ('Multivariate_V2') was supposed to be repeated on a different set of measurements (same turbines, same sensors and same length of the interval, but sampled at a different time) in order to strengthen the physical interpretation of the clustering results. However, because no similar interval could be extracted from the software, this was not possible. Nonetheless, as the results are similar to that acquired from analysing the other data sets, the validity of the clustering results is still strong. In this section, a brief summary of the results of the multivariate cluster analyses will be presented, followed by a comparison between clustering results obtained by clustering the time series of short and modest length.

7.3.1 Similarity in time

In this section, a brief summary of the clustering results from clustering both of the multivariate data sets - with the objective of similarity in time - will be presented. For the first data set clustered ('Multivariate_V1'), the optimal configuration was found to be the dendrogram with average as the linkage criterion and a corresponding cut of five to the dendrogram. All univariate time series were weighted equally during this analysis (i.e. all alphas were set to one). By cutting the dendrogram into five partitions, it managed to separate four turbines into their own separate cluster; all of which deviated significantly from the majority. In short, by clustering the first data set, it managed to separate turbines which experienced different levels of the gearbox temperature, wind speed, generator speeds and the position deviation; a clear separation between the rotating and non-rotating turbines was made by cutting the dendrogram into two partitions. Deeper cuts separated turbines with regards to smaller dissimilarities, such as turbine 16, which experienced quite abnormal behaviour throughout the interval.

The cluster analysis of 'Multivariate_V2' data set presented two scenarios which were clustered: Scenario one, was a cluster analysis on the entire data set where all alphas were set to one except for the one corresponding to the external temperature, which was set to zero; scenario two, was a cluster analysis where all alphas were set to one except for the one corresponding to the wind direction and the external temperature, both alphas were set to zero. For both scenarios, a cut of three to their corresponding dendrogram was found to be the optimal number of clusters. The corresponding cluster assignments were identical for both scenarios and the assignment can be viewed in Figure 7.15. From reviewing the cluster assignment, the majorities of the turbines were assigned to the same cluster; for the two remaining clusters, three turbines were assigned in total. These three turbines were characterised by power production, gearbox temperature and generator speed which were significantly lower than the majority. For two of these turbines, the low generator speed could be explained by an alarmingly high wind speed; however, the third turbine separated did not experience alarmingly high wind speed, but, after reviewing the maintenance log, was confirmed to be under maintenance and therefore manually stopped.

7.3.2 Similarity in shape

In this section, a brief summary of the clustering results from clustering both of the multivariate data sets - with the objective of similarity in shape - will be presented. For the first data set ('Multivariate_V1'), the optimal configuration was found to be the dendrogram with average as the linkage criterion and a corresponding cut of four to the dendrogram. All of the univariate time series were weighted equally during the analysis (i.e. all alphas were set to one). This separated four of the most dissimilar time series from the majority which could be assumed to be operating under normal conditions. In short, the algorithm managed to separate turbines which are different in shape across all dimensions.

For the second data set - similar to the objective of similarity in time - the external temperature was removed prior to clustering (i.e. scenario 1 from before). The optimal configuration was found to be the dendrogram with single as the linkage criterion and a corresponding cut of three to the dendrogram. All alphas were set to one, weighing the different univariate data set equally during clustering. A cut of three separated the three most dissimilar time series from the majority. Cutting the corresponding dendrogram into three partitions resulted in the exact same cluster assignment as for similarity in time. This indicates that the time series are both (dis)similar with respect to the objective of similarity in time and shape. The analysis was also repeated on the reduced data set (i.e. scenario 2 from before), where the time series associated with the wind direction were removed. The cluster

assignment resulted in the exact same cluster assignment as for scenario 1, indicating that the time series for the direction of the wind for the different turbines did not affect the assignment of the cluster.

7.3.3 Comparison - short versus long extraction interval

So, what are the benefits of increasing the length of the time series, and what do we get out of clustering these rather than the shorter ones? For both cluster analysis - regardless of short or long interval - the turbines were successfully separated with respect to dissimilarities in the time series. A clear separation between the rotating and non-rotating turbines was made in both cases. Because many of the parameters included are highly correlated to each other, the separation between rotating and non-rotating is also analogous to the separation between high and low gearbox temperature or high and low power production. So, for both cases, the classification of turbines experiencing different levels of generator speed, gearbox temperature and wind speed were made. The obvious difference between clustering the data set of the short or long interval is that for the data set with the longer interval, the similarity (in time and/or shape) is captured over a longer time interval. Then, local discrepancies - for instance, where the turbines experience different behaviour for a small time span of say 1 hour - becomes less significant to the dissimilarity between two time series and furthermore the assignment of the turbines. In other words, when clustering the short time series, the (dis)similarities is more affected by local (dis)similarities than clustering of the longer time series; clustering of the longer time series captures a more generalised (dis)similarity ((dis)similarity in time, measured on a higher level). Example of local discrepancies is the oscillatory behaviour observed for the shorter time series of the gearbox temperature; these oscillations are not present in the longer time series.

Furthermore, the data set which is extracted over a longer time interval includes dynamics which are unobserved for the data set extracted over a shorter interval. For example, the response of the parameters due to the engaging of the brakes. This response can be observed for the longer time series but is not present in the shorter time series. The engaging of the brakes results in an immediate reduction in the power production and generator speed, and a slower decrease in the gearbox temperature. The threshold value for the wind speed, where the brakes are engaged/disengaged, can be assumed to lie roughly around 15-20 m/s by reviewing the time series in Figure 7.15. Other dynamics which becomes more transparent when reviewing the longer interval is the behaviour of the turbines during steady-state conditions and how the relationship between the different parameters are. Oscillatory behaviour in the time series for the gearbox temperature, such as those found in the shorter data sets, are not present in the longer time series. This is because the aggregation and resampling intervals are significantly larger and basically filters out these oscillations. Increasing the length of the time series may include dynamics which are unobserved during a shorter interval, or it may remove dynamics which are only observed during shorter intervals. The clustering results and the partitioning of the turbines are dependent on which dynamics are observable during the interval of extraction. Depending on the application and the objective, the length of the extracted time series should be carefully chosen: Too short interval has the risk that the relevant dynamics of the wind turbines may be unobserved; too long interval may include these dynamics but may include too much and therefore being uninformative.

Table 7.7: Comparison between the running time of Euclidean and DTW distance on the different data sets. The running time refers to the time it takes to calculate all elements in the condensed distance matrix. Note that the Euclidean distance is in milliseconds and the DTW distance is in seconds.

Data sets	Similarity in time	Similarity in shape	
	Euclidean distance [ms]	DTW distance constrained [s]	DTW distance unconstrained [ms]
Univariate_V1	1.32	413	46
Univariate_V2	1.31	408	44
Multivariate_V1	4.8	1752	161
Multivariate_V2	8.4	6958	480

The running time for the different algorithms, or similarity measures, will now be compared. The different running times for all data sets with the objective of both similarity in time and similarity in shape is presented in Table 7.7. The running times can be observed to increase for both the Euclidean and the DTW distance when going from a univariate case to a multivariate case. The running time for calculations of the Euclidean distance matrix is still relatively low; but the running time for the calculation of the DTW distance matrix, for both the constrained and the unconstrained case, are getting problematically high. The running time of the constrained case

goes from roughly 7 minutes in the univariate case to 29 minutes in for the first multivariate data set, and finally, close to 2 hours for the second and last multivariate data set. The difference between the running time for DTW distance, unconstrained case versus the constrained case, can be observed to be quite significant with a 10 times longer running time for the constrained case. The is because a different function is used for calculation of the DTW distance in the unconstrained case. Sadly, this function did not work in conjunction with the global constraint and, therefore, the slower version had to be implemented. Several DTW algorithms were compared in Appendix C.2, in order to find the implementation with the lowest running time and the most accurate results. The comparison was made for both the constrained and the unconstrained case where the current implementations were superior to the others. There are possibly other Python libraries which have a lower running times, but to the best of the author's knowledge, no such Python libraries exist. With the current implementation of the DTW algorithm, the poor scalability for the calculations of the DTW distance is illustrated in this theses and is therefore not applicable for large data sets.

Finally, the longer the time series are - or the longer the aggregation interval is - the less dominant is the effect of time shifts (time lags). This is because the aggregation and resampling intervals are larger for the data set extracted over a larger time span and the corresponding length of the simple moving median filter is also increased. Therefore, all sampled values are more aggregated (or averaged) and less dependent on small shifts in time. Then, the objective of similarity in time becomes more similar to the objective of similarity in shape. Recall, that similarity in shape is a more general case of similarity, where the time of occurrence of patterns is not important (Aghabozorgi et al., 2015; Zhang et al., 2011). In Appendix B.4.5, the normalised 'Multivariate_V2' data set - not the scaled data set as with the previous similarity in time analysis in Section 7.2.1 - is clustered with the Euclidean distance metric, and the performance is compared to the similarity in shape analysis with DTW distance in Appendix B.4.3. The same dendrogram and the exact same number of cuts to the dendrogram were recommended by comparing the internal indexes for both analyses. Actually, the exact same cluster assignment is achieved by cutting these dendrograms up to and including six partitions. The calculation of the dissimilarity matrix with Euclidean distance took only 8.4 milliseconds compared to, approximately two hours with the DTW distance. Based on these results, the following statement applies for data set extracted over a longer interval: *If the cluster accuracy is more important than the running speed, then, the DTW similarity measure should be used; if the running time is more important than the accuracy, then, the Euclidean distance should be used.* But are longer time series less affected by these time shifts? It is at least obvious that the oscillations observed in the 'Multivariate_V1' data sets could cause some serious "miscalculations" if the Euclidean distance is used instead of the DTW distance; very small shifts in time could be the difference between two time series being very similar to each other or vastly different. Consider the Euclidean distance and the DTW distance between a sinus and cosine wave. Now consider the following example where two cosine waves have very high-frequency oscillations. If one of these is subjected to a relatively small time delay - which results in a phase lag of close to 90 degrees - the corresponding Euclidean distance between these two signals would become vastly different. This time delay does not need to be large, as long as it causes a significant shift in the phase. Longer time series does not contain such high-frequency components - at least not with the current extraction settings and preprocessing steps - and is therefore slightly less affected by small shifts in time (or delays).

Chapter 8

Discussion

The objective of this thesis is to do exploratory work for using clustering techniques for wind turbine condition-based monitoring. More specifically, to explore, understand and summarise the practical implications of using clustering algorithms for automatic classification of wind turbines. To solve this objective, univariate and multivariate time series of different lengths have been clustered with respect to both the clustering objective of similarity in time and similarity in shape. The Euclidean distance was used as the similarity measure to solve the clustering objective of similarity in time and the DTW distance was used as the similarity measure to solve the clustering objective of similarity in shape. Both objectives managed to group similar time series and separate dissimilar ones with respect to their specific similarities. A summary of the major findings from the univariate case was presented in Section 6.3. Then, a brief summary and a comparison between the clustering results obtained from clustering the 'Univariate_V2' and the 'Multivariate_V1' data set were presented in Section 7.1.5. Finally, a summary and a comparison between the length of the multivariate data sets were performed in Section 7.3. The major findings from these sections will be presented along with the relevance and importance of the research question.

8.1 Clustering time series with the objective of similarity in time

First of all, clustering the time series based on the temperature in the gearbox successfully manages to separate turbines experiencing different temperature levels. The obvious practical application for this is that it manages to effectively partition turbines with respect to their overall temperature levels. Because the gearbox temperature is highly correlated to the rotation of the turbines (or generator), the difference between temperatures can easily be justified by a rotation speed which is equally different; turbines experiencing relatively low-temperature levels also experiencing low to no rotation of the generator, and vice versa. By clustering the gearbox temperature, the algorithm successfully manages to automatically classify wind turbines of different temperature levels, which is highly correlated to the classification of turbines experiencing different generator speeds. The assumption was strengthened from clustering the second univariate data set which contains more diverse time series. Initially, it managed to separate those turbines which are not rotating into the same cluster. However, one turbine - which is not rotating nor clustered in the same group - experienced elevated temperature levels in the gearbox, despite not rotating. It was therefore not clustered together with the other non-rotating turbines but was placed in a separate cluster containing only itself. This is expected as the turbine shows abnormal temperature levels, despite not rotating. Because the gearbox temperature and the generator speed are highly correlated to each other, the temperature levels of the gearbox should be similar to the other non-rotating turbines. This demonstrates a situation for where a turbine has been identified, which possibly require some maintenance or at least additional analysis to explain this behaviour. Univariate clustering with the objective of similarity in time separates the turbines based only on their temperature level in the gearbox. If the time series for the generator speed are clustered instead, it would not manage to separate it from the other non-rotating turbines, but it would partition the turbines with respect to whether or not they are rotating. This is because the objective of similarity in time for the generator speed solves a different clustering problem than that of clustering the gearbox temperature. Depending on the time series clustered, careful considerations must be made when defining the problem which the clustering algorithm solves. On the other hand, a multivariate case would solve a more bounded clustering problem, which has more dependencies. This is because, in a multivariate case, the similarity is measured across all dimensions and not only one specific parameter.

Secondly, both multivariate data sets contain a lot of highly correlated time series, which are all highly correlated to the rotation of the turbines. The practical applications of classifying turbines with respect to whether or not they are rotating remain strong for the multivariate case. Actually, the separation between the rotating and non-rotating turbines are more prominent because of the inclusion of several highly correlated time series (i.e. gearbox temperature, generator speed and power). Because many of the time series associated with a dynamic process are highly correlated to each other, the separation between those dynamics becomes more evident if such time series are included. A comparison between the dendrograms of the univariate and the multivariate analysis was made in Section 7.1.5. Recall, that the two data sets were extracted during the exact same time interval and with the same aggregation and resampling intervals. A comparison between the two dendrograms showed that in both cases, the cluster assignment was close to identical to each other; only some minor differences could be observed in the similarities between some of the turbines. This is justified by that the parameter analysed in the univariate case (i.e. gearbox temperature) explains much of the variance between the different turbines and many of the other parameters analysed in the multivariate analysis are highly correlated to this parameter. Careful selection of the different parameters clustered and their correlation to each other must be accounted for prior to any multivariate clustering analysis; the classification problem changes accordingly.

Thirdly, the length of the extracted time series was increased in Section 7.2. This was done to compare the clustering result of time series with different intervals; a proper interval should be large enough to avoid having unobserved dynamics and not too large to avoid superposition of the different dynamics. A summary and comparison between short and long interval were made in Section 7.3. For both cluster analyses - short or long interval - the wind turbines were successfully separated with regards to their differences in the time series. A clear separation between turbines experiencing different behaviour in the gearbox temperature, generator speed and wind speeds was made. When clustering the short time series, the similarity was more affected by local (dis)similarities than clustering of the longer time series; clustering longer time series captures a more generalised (dis)similarity ((dis)similarity in time on a higher level). Depending on the application and the objective, the length of the extracted time series should be carefully chosen to include the relevant dynamics and the (dis)similarities one would like to base the partition on.

Finally, the longer the time series are - or the longer the aggregation interval is - the less dominant is the effect of time shifts (time lags). This is because the aggregation and resampling intervals are larger for the data set extracted over a larger time span and the corresponding length of the SMM filter is also increased. Therefore, all sampled values are more aggregated (or averaged) and less dependent on small shifts in time. Then, the objective of similarity in time becomes more similar to the objective of similarity in shape. Recall, that similarity in shape is a more general case of similarity, where the time of occurrence of patterns is not important (Aghabozorgi et al., 2015; Zhang et al., 2011). In Appendix B.4.5, the normalised 'Multivariate_V2' data set - not the scaled data set as with the previous similarity in time analysis in Section 7.2.1 - is clustered with the Euclidean distance metric, and the performance is compared to the similarity in shape analysis with DTW distance in Appendix B.4.3. The same dendrogram and the exact same number of cuts to the dendrogram were recommended by comparing the internal indexes for both analyses. Actually, the exact same cluster assignment is achieved by cutting these dendrograms up to and including six partitions. The calculation of the dissimilarity matrix with Euclidean distance took only 8.4 milliseconds compared to, approximately two hours with the DTW distance. Based on these results, the following statement applies for data set extracted over a longer interval: If the cluster accuracy is more important than the running speed, then, the DTW similarity measure should be used; if the running time is more important than the accuracy, then, the Euclidean distance should be used. But are longer time series less affected by these time shifts? It is at least obvious that the oscillations observed in the 'Multivariate_V1' data sets could cause some serious "miscalculations" if the Euclidean distance is used instead of the DTW distance; very small shifts in time could be the difference between two time series being very similar to each other or vastly different. Consider the Euclidean distance and the DTW distance between a sinus and cosine wave. Now consider the following example where two cosine waves have very high-frequency oscillations. If one of these is subjected to a relatively small time delay - which results in a phase lag of close to 90 degrees - the corresponding Euclidean distance between these two signals would become vastly different. This time delay does not need to be large, as long as it causes a significant shift in the phase. Longer time series does not contain such high-frequency components - at least not with the current extraction settings and preprocessing steps - and is therefore slightly less affected by small shifts in time (or delays).

8.2 Clustering time series with the objective of similarity in shape

Clustering with the objective of similarity in shape - for both univariate data sets - effectively manages to separate the time series which are dissimilar in shape. Automatic classification of the turbines experiencing different behaviour (i.e. different shapes) can easily be achieved. The practical implication for this separation is the identification of turbines which are behaving abnormally compared to the majority. The turbines separated can then be used as a standpoint for further analysis. Furthermore, due to a strong correlation between the parameters included in the analysis, the clustering results from the univariate and the multivariate analysis are quite similar. Comparison between the dendrograms was made in Section 7.1.5 and showed very similar cluster assignment for both cluster analyses, with the exception of three turbines. Three of the wind turbines were quite dissimilar in shape when only comparing the gearbox temperature. But when considering additional parameters, the clustering results showed that the turbines were indeed quite similar to the majority. This is justified by that these turbines have very similar time series with regards to the new dimensions included in the multivariate case, but quite a different shape of the time series for which the univariate analysis was based upon. In other words, most of the variance (in terms of shape) can be observed in the time series analysed in the univariate case (i.e. the gearbox temperature). To put more emphasis on the similarity between the time series of the gearbox temperature - making it more similar to the univariate case - the corresponding alpha value can be adjusted to have a higher importance. This will be addressed in more detail in the next section.

For the multivariate data set with a relatively short extraction interval (i.e. 'Multivariate_V1'), the shape of the time series for the gearbox temperature was primarily dominated by oscillatory behaviour which was caused by the activation and deactivation of the cooling pump, and the trend associated with the rise or fall in the quality of the wind conditions. For the second multivariate data set, the interval was increased by a factor of 22 along with an appropriate increase of the aggregation and resampling intervals. The effects of increasing the aggregation interval are time series that are smoother and the underlying trend is captured to a greater extent. This removes the effect of the oscillations in the gearbox temperature caused by the cooling pump. The partitioning of the time series is based upon the similarity comparison performed on a higher level than for the shorter time series. The practical applications for clustering short and long time series are different: When clustering the short time series, the similarity is more affected by local dissimilarities than clustering of the longer time series; clustering of the longer time series captures a more generalised similarity (similarity in shape on a higher level). In the previous section (for similarity in time), it was shown that longer time series were less affected by shifts in time and that the Euclidean distance performs surprisingly well when being used instead of the DTW distance on the normalised data set.

The time complexity of using the dynamic time warping distance deserves some additional comments. The running times of the Euclidean distance and the DTW distance on the different data sets can be seen in Table 7.7. The running times can be observed to increase for both the Euclidean and the DTW distance when going from a univariate case to a multivariate case. The running time for calculations of the Euclidean distance matrix is still relatively low; but the running time for the calculation of the DTW distance matrix, for both the constrained and the unconstrained case, are getting problematically high. The running time of the constrained case goes from roughly 7 minutes in the univariate case to 29 minutes in for the first multivariate data set, and finally, close to 2 hours for the second and last multivariate data set. The difference between the running time for DTW distance, unconstrained case versus the constrained case, can be observed to be quite significant with a 10 times longer running time for the constrained case. The is because a different function is used for calculation of the DTW distance in the unconstrained case. Sadly, this function did not work in conjunction with the global constraint and, therefore, a slower version had to be implemented. Several DTW algorithms were compared in Appendix C.2, in order to find the implementation with the lowest running time and the most accurate results. The comparison was made for both the constrained and the unconstrained case where the current implementations were superior to the others. There are possibly other Python libraries which have a lower running times, but to the best of the author's knowledge, no such Python libraries exist. The current implementation of the DTW distance has a high running time and poor scalability and is therefore not suitable for large data sets. The time complexity for the similarity measure plays a vital role in time series clustering (Roelofsen, 2018) and the experiments illustrate the poor scalability of the DTW algorithm with only 15 turbines in the data sets. Imagine, a preferred scenario where the wind park contains several hundred, maybe thousands of wind turbines. In this case, automatic classification by the implementation of the DTW distance metric becomes impractical and obviously cannot be implemented on live data streams. However, if the cluster analysis is only supposed to be run a couple of times a day, the significance of the time complexity places a smaller role when deciding on which similarity measure to implement. For clustering of longer time series, feature-based methods should be implemented instead, as some of these have almost linear time complexity.

8.3 Observations common for both methods

It is important to keep in mind that when performing a cluster analysis on any data set, the clustering algorithm will always manage to partition the data set into a predetermined number of clusters, except for the trivial case where all the time series are identical (James et al., 2013). It is up to the user to interpret the meaning of the cluster assignment (i.e. defining the clustering problem appropriately) and its validity. The definition of the clustering problem is dependent on many factors, such as the parameters included in the data set, the number of dimensions analysed, the combination of the weighting vector and how the number of clusters is determined, to mention the most critical components addressed in this thesis. By carefully considering which parameters to include in the analysis and adjusting the importance of the individual time series appropriately, the clustering problem can be customised to the specific application.

The choice of which time series to include in a multivariate cluster analysis and how they are weighted are important considerations when defining the clustering problem. Without adjusting the weighting vector, all the individual (dis)similarities are averaged across all dimensions to construct the averaged dissimilarity matrix which the clustering is based upon. If this is preferred, the corresponding weighting values (alpha) can also be adjusted and customised to the specific application and objective. For example, if for the gearbox temperature it is much more critical to separate those experiencing abnormal temperature levels rather than those experiencing different wind speeds, then, a high weighting would, in this case, be given to gearbox temperature and a lower weighting would be given to the wind speed parameter. This will weigh the influence of the (dis)similarities observed in the gearbox temperature more than the (dis)similarities observed in the time series for the wind speed. Another example where adjusting the alpha mighty become advantageous is in the case where several highly correlated parameters are included in the analysis. If these are the majority of the parameters included in the multivariate analysis, their dynamics will dominate the cluster assignment. The corresponding alpha value can, in this case, be adjusted to average out this effect. The practical applications vary from the choice of which parameters to include in the analysis and how the corresponding parameters are weighted with the alpha values.

In the multivariate cluster analysis, clustering is performed on time series which are highly correlated to each other. Clustering of the gearbox temperature, generator speed and power production are all associated with similar problem definition. The problem definition could be stated in numerous ways, all of which are similar to each other. For example, the current partitioning of the wind turbines - in the 'Multivariate_V2' analysis - could be defined as the partitioning or classification of the wind turbines based on the similarity in their power production. Similarly, this also corresponds to the partition of turbines associated with similarities in generator speed and gearbox temperature. The objective of the partitioning further depends on how similar turbines should be within all clusters; the compromise between separation and within-cluster similarity becomes an important consideration. With that in mind, another way of cutting the dendrogram is to decide on an upper threshold value which corresponds to the largest dissimilarity allowed within each cluster (i.e. largest within-cluster variance for each cluster). The algorithm will then consistently cut the dendrogram to form clusters which have, for example, a within-cluster variance of less than 5%. However, as the dissimilarities between the separated turbines are getting smaller, the interpreting the physical meaning and practical application of the partitioning become harder and less trivial to find. Finding the best compromise between separation and similarity is a difficult task and is an important consideration for all clustering applications.

Finally, the K-means algorithm was implemented on the first univariate data set in Section 6.1.2. The clustering results were shown to be inconsistent and the partitioning proves hard to verify even with random initiation, increasing the number iterations, or adjusting the termination criteria for the optimisation problem. In this thesis, for a small data set of only 15 turbines (or objects), the benefit from using K-means algorithm over the hierarchical clustering algorithm is non-existing. However, for the scaled-up version where the number of wind turbines could potentially be several hundred turbines, maybe thousands, the K-means algorithm should be considered as it has much lower time complexity than the hierarchical clustering algorithm. Actually, it could be used in conjunction with the hierarchical clustering algorithm as a hybrid clustering approach to improve the accuracy of the hierarchical clustering algorithm (Aghabozorgi et al., 2015).

8.4 Limitations of the study

The limitations of the study will be presented in the following list, where the importance of the limitations is presented in a decreasing manner (i.e. the first mentioned is of highest importance):

- The number of turbines in each data set analysed were limited to only 15 turbines. The reason for the small number of turbines analysed is that the current wind park - from which the time series were extracted from - contained only 16 wind turbines, where 15 of them had measurements which were available for extraction. Preferably, a wind park of several hundred wind turbines is an optimal scenario and if available, the experiment should be repeated to verify the results of this thesis. However, it is of the author's opinion that even though the data sets contained only 15 turbines, it shows the practical implications for using clustering techniques to classify turbines within a wind park to a large extent.
- The clustering objectives were limited to similarity in time and similarity in shape, and clustering was not performed with regards to the similarity in change (i.e. the third and last clustering objective).
- The analysis of the second multivariate data set on a longer time interval was not repeated, but the results were consistent with the clustering results from that of the other data sets. Thus, the legitimacy of the results achieved for the second analysis is still strong.
- Only one similarity measure has been implemented for each of the objectives: Euclidean and DTW similarity measure. These are the two most commonly used similarity methods in the literature and are very competitive, as justified in Chapter 2. The scope of this thesis is limited to implementing these two. In the bigger picture, other similarity measures should also be implemented for comparison.
- The Sakeo-Chiba band for restricting the DTW algorithm were set to a width of 10%, as most of the literature use this width. The choice was made based upon a comparison between a width of 5%, 10% and the unconstrained case. No detailed analysis of the clustering accuracy for different widths could be performed as the ground truth was not available. However, no significant improvements in the accuracy is expected from correcting the width but should be tested if the ground truth is available and a more detailed approach is possible.

Chapter 9

Conclusion and future work

9.1 Conclusion

The aim of this thesis was to do exploratory work for using clustering techniques for wind turbine condition-based monitoring. More specifically, to explore, understand and summarise the practical implications of using clustering algorithms for automatic classification of wind turbines. Two methods - which has two different clustering objectives, namely similarity in time and similarity in shape - have been implemented to cluster univariate and multivariate data sets. Both methods effectively managed to partition turbines with regards to their specific measure of (dis)similarity: Euclidean distance in conjunction with the agglomerative hierarchical clustering algorithm effectively solves the objective of similarity in time; DTW distance in conjunction with the agglomerative hierarchical clustering algorithm solves the objective of similarity in shape.

Automatic classifications of wind turbines based on their differences were made - this includes classification of wind turbines experiencing different gearbox temperatures, generator speed, power production and local wind conditions. The classification of turbines by clustering the univariate and the multivariate data sets show similar cluster assignments. This is because many of the parameter associated with a dynamic process are highly correlated to each other. The classification of wind turbines in the multivariate case is, therefore, more prominent as many of the included parameters as highly correlated to the parameter analysed in the univariate case.

Both clustering methods manage to identify turbines which are experiencing quite abnormal behaviour compared to the others; this can be used as a standpoint for further, and more detailed, analysis of the identified turbines. The length of the clustered time series plays a vital part in the definition of the clustering problem; a more generalised clustering problem is obtained by clustering longer time series rather than short. As the length of the time series increases, the difference between the two clustering objectives becomes less significant. In this situation, DTW distance can be substituted with the Euclidean distance, as the objectives of similarity in time and similarity in shape are more similar. Euclidean distance can effectively be used in a scaled-up version, whereas the DTW distance cannot because of the quadratic time complexity of the DTW algorithm. The poor scalability of the DTW algorithm is illustrated in the thesis.

A better method for dealing with time shifts should be implemented if the running time is an issue. However, the benefit of using DTW distance over Euclidean distance is that the measure is not limited to the (dis)similarity between equal-length time series and can also effectively deal with time shifts. This generally results in higher accuracy in terms of the clustering results (as shown by numerous papers) and should be used if the increased running time is of no concern. Both methods solve its corresponding objective with a satisfactory result. This thesis forms the basis for further research into the practical implications of using clustering algorithms for automatic classification of wind turbines.

9.2 Future work

This thesis has compared several methods to explore the practical implications of using clustering algorithms for automatic classification of time series clustering. To better understand the practical implications of the results, future studies could apply the same methods to a wind park of larger dimensionality. Furthermore, the scope of the thesis is limited to clustering time series with the shape-based approach, which limits the objective of the

time series clustering to the objectives of similarity in time and similarity in shape. One interesting continuation could be the implementation of the third, and the last objective of time series clustering, namely, the objective of similarity in change. To cluster the objective of similarity in change, the model-based approach as introduced in Section 2.2.2 should be implemented. It would be interesting to compare the practical implication of clustering the time series with the third objective and compare it to the remaining objectives analysed in this thesis. Another continuation could be to find a replacement for the DTW algorithm as the similarity measure was shown to not be inappropriate for large data sets and therefore impractical to use in a scaled-up cluster analysis. This can be done by clustering the time series through the feature-based approach; converting the time series into feature vectors and then these features are clustered with conventional clustering algorithms.

Bibliography

Aghabozorgi, S., Seyed Shirkhorshidi, A., Ying Wah, T., 2015. Time-series clustering - a decade review. Information Systems 53, 16–38.

Ahmad, A., Khan, S. S., 2018a. A survey of mixed data clustering algorithms. CoRR abs/1811.04364. URL http://arxiv.org/abs/1811.04364

Ahmad, A., Khan, S. S., 2018b. A survey of mixed data clustering algorithms.

Algorithmia Inc., April 2018. Introduction to unsupervised learning [blogg]. https://blog.algorithmia.com/introduction-to-unsupervised-learning/, accessed 22. January 2018.

Almaliki, Z. A., July 2018. Standardization vs normalization. https://medium.com/@zaidalissa/standardization-vs-normalization-da7a3a308c64, accessed 25. February 2019.

Aloise, D., Deshpande, A., Hansen, P., Popat, P., 2009. Np-hardness of euclidean sum-of-squares clustering. Machine Learning 75 (2), 245–248.

Antunes, C., Oliveira, A., 01 2001. Temporal data mining: An overview. KDD Workshop on Temporal Data Mining, 1–13.

Argyros, T., Ermopoulos, C., 2003. Efficient subsequence matching in time series databases under time and amplitude transformations. In: Third IEEE International Conference on Data Mining. IEEE, pp. 481–484.

Bagnall, A., Janacek, G., 2005. Clustering time series with clipped data. Machine Learning 58 (2), 151–178.

Bagnall, A., Ratanamahatana, C., Keogh, E., Lonardi, S., Janacek, G., 2006. A bit level representation for time series data mining with shape based similarity. Data Mining and Knowledge Discovery 13 (1), 11–40.

Beringer, J., Hüllermeier, E., 2006. Online clustering of parallel data streams. Data & Knowledge Engineering 58 (2), 180–204.

Berndt, D. J., Clifford, J., 1994. Using dynamic time warping to find patterns in time series. In: KDD workshop. Vol. 10, No. 16. Seattle, WA, pp. 359–370.

Bradley, P. S., Fayyad, U. M., Reina, C., et al., 1998. Scaling clustering algorithms to large databases. In: KDD. Vol. 98. pp. 9–15.

Brownlee, J., December 2016a. How to normalize and standardize time series data in python. https://machinelearningmastery.com/normalize-standardize-time-series-data-python/, accessed 09. April 2019.

Brownlee, J., December 2016b. How to resample and interpolate your time series data with python. https://machinelearningmastery.com/resample-interpolate-time-series-data-python/, accessed 08. April 2019.

Cassisi, C., Montalto, P., Aliotta, M., Cannata, A., Pulvirenti, A., et al., 2012. Similarity measures and dimensionality reduction techniques for time series data mining. InTech.

Chen, J. R., 2005. Making subsequence time series clustering meaningful. In: Fifth IEEE International Conference on Data Mining (ICDM'05). IEEE, pp. 8–pp.

Chen, J. R., 2007. Useful clustering outcomes from meaningful time series clustering. In: Proceedings of the sixth Australasian conference on Data mining and analytics-Volume 70. Australian Computer Society, Inc., pp. 101–109.

Diez, A., Khoa, N. L. D., Makki Alamdari, M., Wang, Y., Chen, F., Runcie, P., Jul 2016. A clustering approach for structural health monitoring on bridges. Journal of Civil Structural Health Monitoring 6 (3), 429–445.
URL https://doi.org/10.1007/s13349-016-0160-0

Dunn, K., 2010. Process improvement using data. Retrieved on, 11–2016Accessed 20. October 2018.

Duval, L., February 2016. What does prototype mean in clustering? https://dsp.stackexchange.com/questions/28593/what-does-prototype-mean-in-clustering, [Online discussion group] Accessed 10. February 2019.

Elsworth, S., September 2017. Dynamic time warping. http://www.maths.manchester.ac.uk/~mbbx2se2/Docs/Dynamic_time_warping(Steven_Elsworth).pdf, accessed 22. February 2019.

Esling, P., Agon, C., 2012. Time-series data mining. ACM Computing Surveys (CSUR) 45 (1), 1–34.

Farahani, E., Hosseinzadeh, N., Ektesabi, M., 2012. Comparison of fault-ride-through capability of dual and single-rotor wind turbines. Renewable Energy 48, 473–481.

Fowlkes, E. B., Mallows, C. L., 1983. A method for comparing two hierarchical clusterings. Journal of the American Statistical Association 78, 553–569, http://wildfire.stat.ucla.edu/pdflibrary/fowlkes.pdf.

Fujita, A., Severino, P., Kojima, K., Sato, J. R., Patriota, A. G., Miyano, S., 2012. Functional clustering of time series gene expression data by granger causality. BMC systems biology 6.

Global Wind Energy Council, 2019a. Africa and middle east installed 962mw new wind capacity in 2018 – over 300mw more than in 2017. https://gwec.net/africa-and-middle-east-installed-962mw-new-wind-capacity-in-2018-over-300mw-more-than-in-2017/, accessed 09. February 2019.

Global Wind Energy Council, 2019b. Americas install 11.9gw wind capacity in 2018 – increase by 12%. https://gwec.net/americas-install-11-9gw-wind-capacity-in-2018-increase-by-12/, accessed 09. February 2019.

Górecki, T., 2018. Classification of time series using combination of dtw and lcss dissimilarity measures. Communications in Statistics-Simulation and Computation 47 (1), 263–276.

Gruber, P., 2017. Comparison of different outlier filtering methods with applications to ultrasonic flow measurements. In: IGHEM Conference 2016, Linz, Austria. pp. 1–13.

Gullo, F., Ponti, G., Tagarelli, A., Tradigo, G., Veltri, P., 2012. A time series approach for clustering mass spectrometry data. Journal of Computational Science 3 (5), 344 – 355, advanced Computing Solutions for Health Care and Medicine.
URL http://www.sciencedirect.com/science/article/pii/S1877750311000627

Guo, C., Jia, H., Zhang, N., 2008. Time series clustering based on ica for stock data analysis. In: 2008 4th International Conference on Wireless Communications, Networking and Mobile Computing. IEEE, pp. 1–4.

Hastie, T., Tibshirani, R., Friedman, J., 2009. The elements of statistical learning: data mining, inference, and prediction.

Hautamaki, V., Nykanen, P., Franti, P., 2008. Time-series clustering by approximate prototypes. In: 2008 19th International Conference on Pattern Recognition. IEEE, pp. 1–4.

Hossain, M., Abu-Siada, A., Muyeen, S., 2018. Methods for advanced wind turbine condition monitoring and early diagnosis: A literature review. Energies 11 (5).

Huang, X., Ye, Y., Xiong, L., Lau, R. Y., Jiang, N., Wang, S., 2016. Time series k-means: A new k-means type smooth subspace clustering for time series data. Information Sciences 367-368, 1 – 13.
URL http://www.sciencedirect.com/science/article/pii/S0020025516303796

Hubert, L., Arabie, P., Dec 1985. Comparing partitions. Journal of Classification 2 (1), 193–218.
URL https://doi.org/10.1007/BF01908075

Iglesias, F., Kastner, W., 2013. Analysis of similarity measures in times series clustering for the discovery of building energy patterns. Energies 6 (2), 579–597.
URL http://search.proquest.com/docview/1537076266/

IMB, n.d. Data aggregation. https://www.ibm.com/support/knowledgecenter/en/SSBNJ7_1.4.2/dataView/Concepts/ctnpm_dv_use_data_aggreg.html, accessed 08. April 2019.

Indian Agricultural Statistics Research Institute, July 2011. Data preprocessing techniques for data mining. http://iasri.res.in/ebook/win_school_aa/notes/Data_Preprocessing.pdf, accessed 29. Mars 2019.

Iwok, I. A., Okpe, A. S., 2016. A comparative study between univariate and multivariate linear stationary time series models. In: American Journal of Mathematics and Statistics. Vol. 6 No. 5. pp. 203–212, doi: 10.5923/j.ajms.20160605.02.

Jain, A. K., Dubes, R. C., et al., 1988. Algorithms for clustering data. Vol. 6. Prentice hall Englewood Cliffs.

James, G., Witten, D., Hastie, T., Tibshirani, R., 2013. An Introduction to Statistical Learning: with Applications in R2. Springer, New York, NY.

Kaufman, L., 1990. Finding groups in data : an introduction to cluster analysis.

Kaushik, S., November 2016. An introduction to clustering and different methods of clustering [blogg]. https://www.analyticsvidhya.com/blog/2016/11/an-introduction-to-clustering-and-different-methods-of-clustering/, accessed 26. January 2018.

Keogh, E., Lin, J., 2005. Clustering of time-series subsequences is meaningless: implications for previous and future research. Knowledge and Information Systems 8 (2), 154–177.

Keogh, E. J., Pazzani, M. J., 2000. A simple dimensionality reduction technique for fast similarity search in large time series databases. In: Pacific-Asia conference on knowledge discovery and data mining. Springer, pp. 122–133.

Kongsberg Digital, Mars 2019. Kognifai ecosystem and applications. https://www.kongsberg.com/en/kongsberg-digital/renewables%20and%20utilities/wind-farm-management-prod/, accessed 29. Mars 2019.

Lapointe, F.-J., Legendre, P., 1995. Comparison tests for dendrograms: A comparative evaluation. Journal of Classification 12 (2), 265–282.

Lin, J., Keogh, E., Lonardi, S., Chiu, B., 2003a. A symbolic representation of time series, with implications for streaming algorithms. In: Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery. ACM, pp. 2–11.

Lin, J., Keogh, E., Truppel, W., 2003b. Clustering of streaming time series is meaningless. In: Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery. ACM, pp. 56–65.

Lin, J., Vlachos, M., Keogh, E., Gunopulos, D., 2004. Iterative incremental clustering of time series. In: International Conference on Extending Database Technology. Springer, pp. 106–122.

Liu, H. M., Lu, J. G., 2015. Brief survey of k-means clustering algorithms. Applied Mechanics and Materials 740-740 (Mechanical, Information and Industrial Engineering), 624–628.

Lkhagva, B., Suzuki, Y., Kawagoe, K., 2006. New time series data representation esax for financial applications. In: 22nd International Conference on Data Engineering Workshops (ICDEW'06). IEEE, pp. x115–x115.

MacQueen, J., 1967. Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics. University of California Press, Berkeley, Calif., pp. 281–297.
URL https://projecteuclid.org/euclid.bsmsp/1200512992

Maini, V., August 2017. Machine learning for humans, part 3: Unsupervised learning. https://medium.com/machine-learning-for-humans/unsupervised-learning-f45587588294, accessed 24. January 2018.

Manning, C., Raghavan, P., Schütze, H., 2010. Introduction to information retrieval. Natural Language Engineering 16 (1), 100–103.

Mitsa, T., 2010. Temporal data mining. Chapman and Hall/CRC.

Montero, P., Vilar, J. A., et al., 2014. Tsclust: An r package for time series clustering. Journal of Statistical Software 62 (1), 1–43.

Mori, U., Mendiburu, A., Lozano, J. A., 2016. Similarity measure selection for clustering time series databases. Knowledge and Data Engineering, IEEE Transactions on 28 (1), 181–195.

Nagpal, A., Jatain, A., Gaur, D., April 2013a. Review based on data clustering algorithms. In: 2013 IEEE Conference on Information Communication Technologies. pp. 298–303.

Nagpal, A., Jatain, A., Gaur, D., April 2013b. Review based on data clustering algorithms. In: 2013 IEEE Conference on Information Communication Technologies. pp. 298–303.

NCSS, LLC, 2019. Ncss statistical software: Chapter 445 - hierarchical clustering dendrograms. https://ncss-wpengine.netdna-ssl.com/wp-content/themes/ncss/pdf/Procedures/NCSS/Hierarchical_Clustering-Dendrograms.pdf, accessed 26. April 2019.

Nguyen, H. D., Ullmann, J. F. P., McLachlan, G. J., Voleti, V., Li, W., Hillman, E. M. C., Reutens, D. C., Janke, A. L., 2018. Whole-volume clustering of time series data from zebrafish brain calcium images via mixture modeling. Statistical Analysis and Data Mining: The ASA Data Science Journal 11 (1), 5–16.
URL https://onlinelibrary.wiley.com/doi/abs/10.1002/sam.11366

Niennattrakul, V., Ratanamahatana, C., 11 2006. Clustering multimedia data using time series (pdf). Hybrid Information Technology, International Conference on 1, 372–379.

Niennattrakul, V., Ratanamahatana, C. A., 2007. Inaccuracies of shape averaging method using dynamic time warping for time series data. In: International conference on computational science. Springer, pp. 513–520.

Patibandla, R. L., Veeranjaneyulu, N., 2018. Survey on clustering algorithms for unstructured data. In: Intelligent Engineering Informatics. Springer, pp. 421–429.

Petitjean, F., Gançarski, P., 2011. Summarizing a set of time series by averaging: From steiner sequence to compact multiple alignment. Theoretical Computer Science 414 (1).

Pyatnitskiy, M., Mazo, I., Shkrob, M., Schwartz, E., Kotelnikova, E., 2014. Clustering gene expression regulators: new approach to disease subtyping. PLoS ONE 9 (1).
URL https://doaj.org/article/34462269f5c148f9b41b413a6d7140bf

Rai, P., Shubha, S., 10 2010. A survey of clustering techniques. International Journal of Computer Applications 7.

Rakthanmanon, T., Campana, B., Mueen, A., Batista, G., Westover, B., Zhu, Q., Zakaria, J., Keogh, E., 2012. Searching and mining trillions of time series subsequences under dynamic time warping. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '12. ACM, New York, NY, USA, pp. 262–270.
URL http://doi.acm.org/10.1145/2339530.2339576

Ratanamahatana, C., Keogh, E., 01 2004. Everything you know about dynamic time warping is wrong.

Ratanamahatana, C., Keogh, E., Bagnal, A., Lonardi, S., 2005. A novel bit level time series representation with implication of similarity search and clustering. Advances In Knowledge Discovery And Data Mining, Proceedings 3518, 771–777.

Rodpongpun, S., Niennattrakul, V., Ratanamahatana, C. A., 2012. Selective subsequence time series clustering. Knowledge-Based Systems 35, 361–368.

Rodriguez, M. Z., Comin, C. H., Casanova, D., Bruno, O. M., Amancio, D. R., Costa, L. d. F., Rodrigues, F. A., 2019. Clustering algorithms: A comparative approach.(research article). PLoS ONE 14 (1).

Roelofsen, P., March 2018. Time series clustering. (Master thesis, Vrije Universiteit Amsterdam) From https://www.math.vu.nl/~sbhulai/papers/thesis-roelofsen.pdf, accessed 26. April 2019.

Rousseeuw, J., 1989. A graphical aid to the interpretation and validation of cluster analysis. Journal of Computational Application Math.

Rousseeuw, P. J., 1987. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics 20, 53 – 65.
URL http://www.sciencedirect.com/science/article/pii/0377042787901257

Saraçli, S., Doğan, N., Doğan, İ., Apr 2013. Comparison of hierarchical cluster analysis methods by cophenetic correlation. Journal of Inequalities and Applications 2013 (1), 203.
URL https://doi.org/10.1186/1029-242X-2013-203

Sarah Guido, A. C. M., 2013. Introduction to machine learning with python.

Sardá-Espinosa, A., 2017. Comparing time-series clustering algorithms in r using the dtwclust package. Vienna: R Development Core Team.

Sayad, S., n.d. Hierarchical clustering. https://www.saedsayad.com/clustering_hierarchical.htm, accessed 24. February 2019.

Sequeira, K., Zaki, M., 2002. Admit: anomaly-based data mining for intrusions. In: Proceedings of the eighth ACM SIGKDD international conference on knowledge discovery and data mining. KDD '02. ACM, pp. 386–395.

Serega, November 2015. What does prototype mean in clustering? https://dsp.stackexchange.com/questions/27349/moving-average-vs-moving-median, [Online discussion group] Accessed 10. February 2019.

Shieh, J., Keogh, E., 2009. i sax: disk-aware mining and indexing of massive time series datasets. Data Mining and Knowledge Discovery 19 (1), 24–57.

Singh, A., Yadav, A., Rana, A., 04 2013. K-means with three different distance metrics. International Journal of Computer Applications 67, 13–17.

Soua, S., Van Lieshout, P., Perera, A., Gan, T.-H., Bridge, B., 2013. Determination of the combined vibrational and acoustic emission signature of a wind turbine gearbox and generator shaft in service as a pre-requisite for effective condition monitoring. Renewable Energy 51 (C), 175–181.

Starczewski, A., Krzyżak, A., 2015. Performance evaluation of the silhouette index. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L. A., Zurada, J. M. (Eds.), Artificial Intelligence and Soft Computing. Springer International Publishing, Cham, pp. 49–58.

Stephenson, D., February 2000. Filtering and smoothing. https://folk.uib.no/ngbnk/kurs/notes/node107.html, accessed 08. April 2019.

Subhani, N., Rueda, L., Ngom, A., Burden, C. J., 2010. Multiple gene expression profile alignment for microarray time-series data clustering. Bioinformatics 26 (18), 2281–2288.

Tan, P., Steinbach, M., Kumar, V., 01 2005. Cluster analysis: Basic concepts and algorithms. Introduction to Data Mining, 487–568.

Teknomo, K., 2017. Cophenetic correlation coefficient. https://people.revoledu.com/kardi/tutorial/Clustering/Cophenetic.htm, accessed 09. April 2019.

Torabi, A. J., Er, M. J., Li, X., Lim, B. S., Peen, G. O., June 2016. Application of clustering methods for online tool condition monitoring and fault diagnosis in high-speed milling processes. IEEE Systems Journal 10 (2), 721–732.

Trevor, H., Robert, T., JH, F., 2009. The elements of statistical learning: data mining, inference, and prediction.

Viswanathan, M., November 2010. Moving average filter (ma filter). https://www.gaussianwaves.com/2010/11/moving-average-filter-ma-filter-2/, accessed 08. April 2019.

Vlachos, M., Gunopulos, D., Das, G., 2004. Indexing time-series under conditions of noise. In: Data Mining In Time Series Databases. World Scientific Publishing Co. Pte. Ltd., pp. 67–100.

Vlachos, M., Kollios, G., Gunopulos, D., 2002. Discovering similar multidimensional trajectories. In: Proceedings 18th international conference on data engineering. IEEE, pp. 673–684.

Vlachos, M., Lin, J., Keogh, E., Gunopulos, D., 2003. A wavelet-based anytime algorithm for k-means clustering of time series. In: In proc. workshop on clustering high dimensionality data and its applications. Citeseer, pp. 23–30.

Wang, X., Smith, K., Hyndman, R., 2006a. Characteristic-based clustering for time series data. Data mining and knowledge Discovery 13 (3), 335–364.

Wang, X., Smith, K. A., Hyndman, R., Alahakoon, D., 2004. A scalable method for time series clustering. Technical report.

Wang, X., Smith-Miles, K., Hyndman, R., 09 2006b. Characteristic-based clustering for time series data. Data Min. Knowl. Discov. 13, 335–364.

Warren Liao, T., 2005. Clustering of time series data - a survey. Pattern Recognition 38 (11), 1857–1874.

Wikimedia Foundation Inc., Mars 2019. Aggregate data. https://en.wikipedia.org/wiki/Aggregate_data, accessed 08. April 2019.

Wolfram Research, 2004. Fuzzy clustering. https://reference.wolfram.com/legacy/applications/fuzzylogic/Manual/12.html, accessed 26. January 2019.

Wong, K.-C., 2015. A short survey on data clustering algorithms. In: 2015 Second International Conference on Soft Computing and Machine Intelligence (ISCMI). IEEE, pp. 64–68.

Xi, X., Keogh, E., Shelton, C., Wei, L., Ratanamahatana, C. A., 2006. Fast time series classification using numerosity reduction. In: Proceedings of the 23rd international conference on Machine learning. ACM, pp. 1033–1040.

Xu, D., Tian, Y., 2015a. A comprehensive survey of clustering algorithms. Annals of Data Science 2 (2), 165–193.

Xu, D., Tian, Y., 2015b. A comprehensive survey of clustering algorithms. Annals of Data Science 2 (2), 165–193.

Yang, W., Court, R., Jiang, J., 2013. Wind turbine condition monitoring by the approach of scada data analysis. Renewable Energy 53 (C), 365–376.

Yu, P., November 2015. What does prototype mean in clustering? https://dsp.stackexchange.com/questions/27349/moving-average-vs-moving-median, [Online discussion group] Accessed 10. February 2019.

Zappi, P., Stiefmeier, T., Farella, E., Roggen, D., Benini, L., Troster, G., Dec 2007. Activity recognition from on-body sensors by classifier fusion: sensor scalability and robustness. In: 2007 3rd International Conference on Intelligent Sensors, Sensor Networks and Information. pp. 281–286.

Zhang, H., Ho, T. B., Zhang, Y., Lin, M.-S., 2006. Unsupervised feature extraction for time series clustering using orthogonal wavelet transform. Informatica 30 (3).

Zhang, W., Ma, X., 2016. Simultaneous fault detection and sensor selection for condition monitoring of wind turbines. Energies 9 (4).
URL http://www.mdpi.com/1996-1073/9/4/280

Zhang, X., Liu, J., Du, Y., Lv, T., 2011. A novel clustering method on time series data. Expert Systems With Applications 38 (9), 11891–11900.

Zhang, Z., Verma, A., Kusiak, A., 2012. Fault analysis and condition monitoring of the wind turbine gearbox. IEEE Transactions on Energy Conversion 27 (2), 526–535.

Zolhavarieh, S., Aghabozorgi, S., Teh, Y. W., 2014. A review of subsequence time series clustering. The Scientific World Journal.

Appendix

Appendix A - Plots and summary tables for different univariate time series

In this appendix, time series associated with the different data sets is presented, as long as these are not already presented in Chapter 5. Also, additional time series used for interpreting the results of the univariate clustering results in Chapter 6 are presented in this appendix. In Appendix A.1, additional time series used during the analysis of the first univariate data set, 'Univariate_V1', is presented. As for the remaining appendices, A.2 and A.3, the time series associated with the data sets, 'Multivariate_V1' and 'Multivariate_V2' are presented, respectively.

A.1: Additional time series during the same interval as 'Univariate_V1'

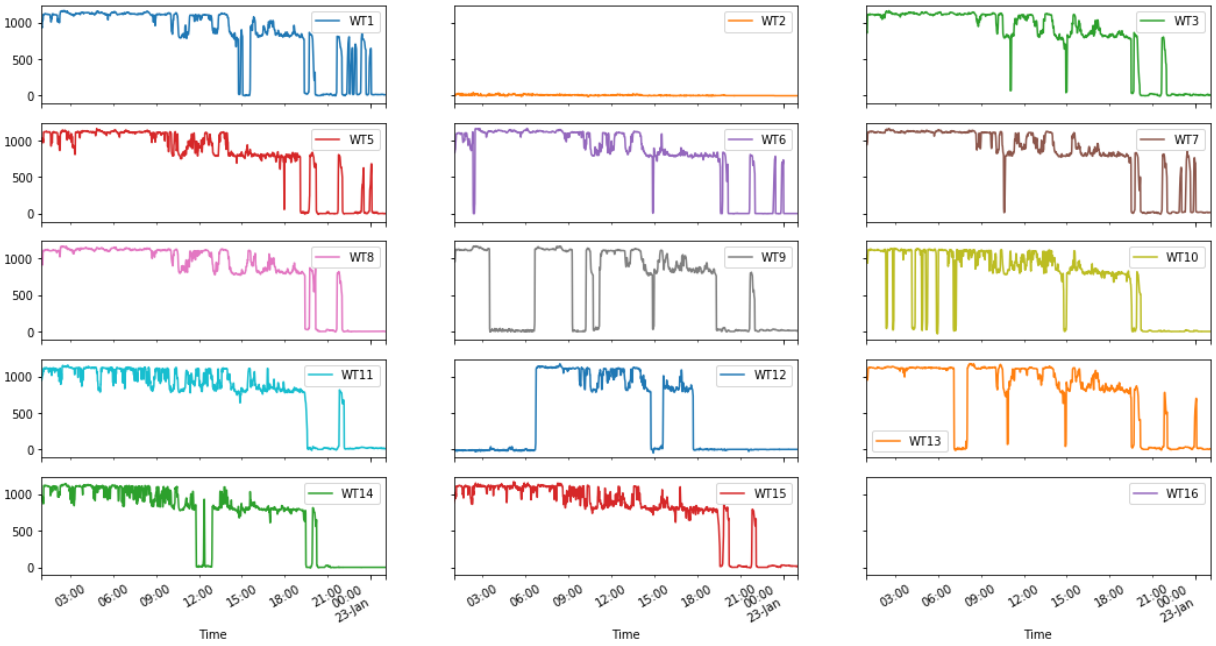


Figure A.1.1: Subplots of the generator speed during the same interval as the data 'Univariate_V1'.

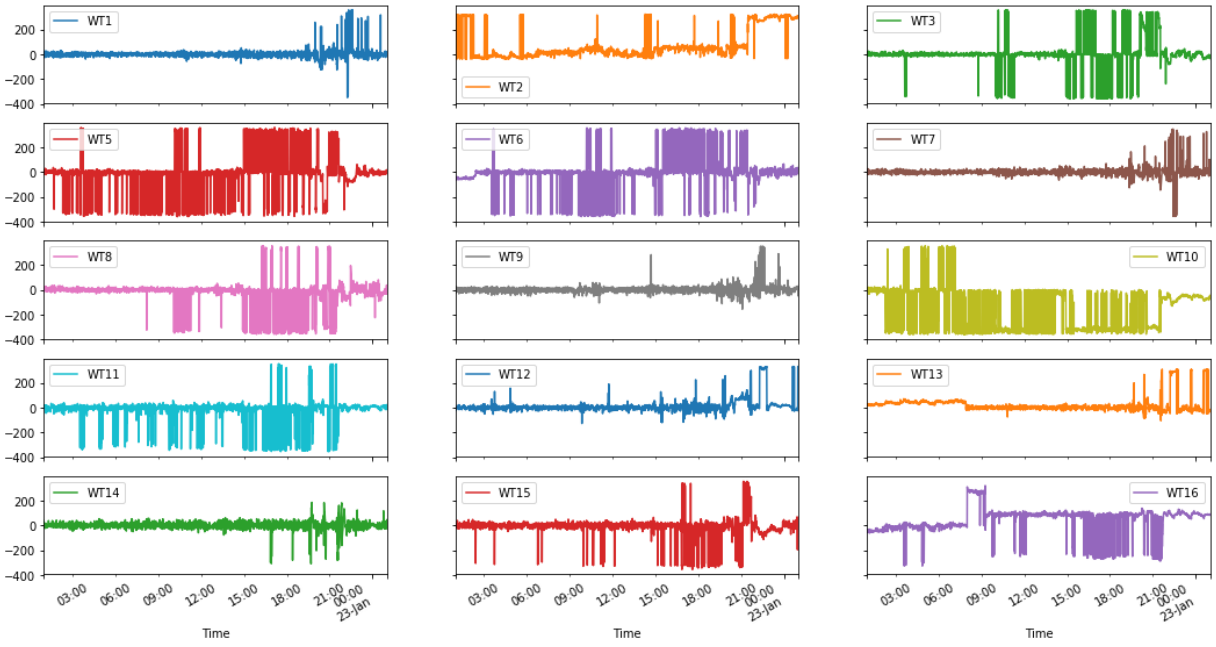


Figure A.1.2: Subplots of the positional deviations during the same interval as the data 'Univariate_V1'. The spikes indicate a misinterpretation of the angles as the spikes have values of -360 and 360, which is the same as saying the angle is 0.

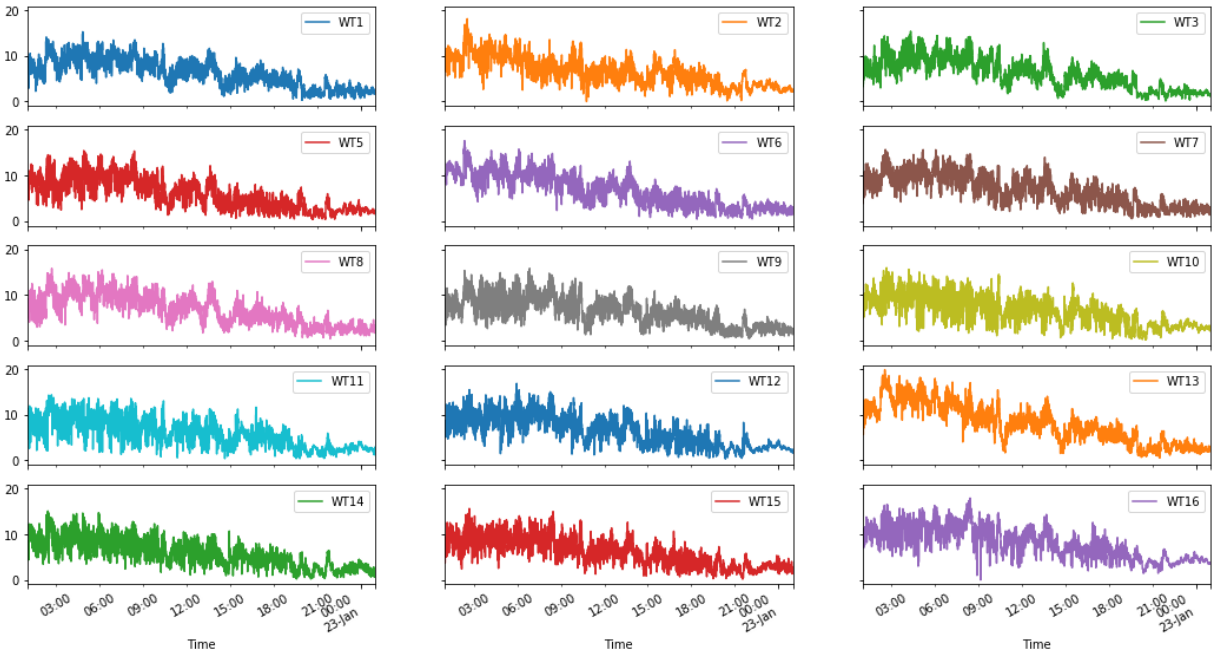


Figure A.1.3: Subplots of the wind speed [m/s] measured at the nacelle during the same interval as the data 'Univariate_V1'. All experience quite similar behaviour.

Table A.1.1: Descriptive statistics for the wind speed of Figure A.1.3

	WT1	WT2	WT3	WT5	WT6	WT7	WT8	WT9	WT10	WT11	WT12	WT13	WT14	WT15	WT16
count	2881	2881	2881	2881	2881	2881	2881	2881	2881	2881	2881	2881	2881	2881	2881
mean	6.2	6.7	6.1	6.0	6.7	6.8	6.8	6.3	6.5	6.0	6.4	8.0	5.9	6.1	8.1
std	3.1	3.1	3.3	3.4	3.6	3.4	3.4	3.0	3.2	3.1	3.3	4.3	3.1	3.2	3.5

A.2: Multivariate_V1

In this appendix, the time series corresponding to the multivariate data set named 'Multivariate _V1' is presented. The time series presented are all filtered with an SMM filter of length 5, except for the one corresponding to the positional deviation (i.e. angles) where phase unwrapping is applied prior to filtering. The specific parameters for extraction can be seen in Section 5.2.1. The univariate time series corresponding to the generator speed, the wind speed and the position deviation and - before any preprocessing - can be seen in the consecutive figures (Figure A.2.1, A.2.2 and A.2.3, respectively). The time series of the position deviation after phase unwrapping is presented in Figure A.2.4. Then, a comparison between measured rotor and generator speed, after filtering, is presented in Figure A.2.5 and A.2.6, respectively. At last, the filtered time series for the positional deviations (which is phase unwrapped first) and the wind speed are presented in FigureA.2.7 and A.2.8, respectively

Initial plots before any preprocessing

First of all, the initial plots without any preprocessing is presented.

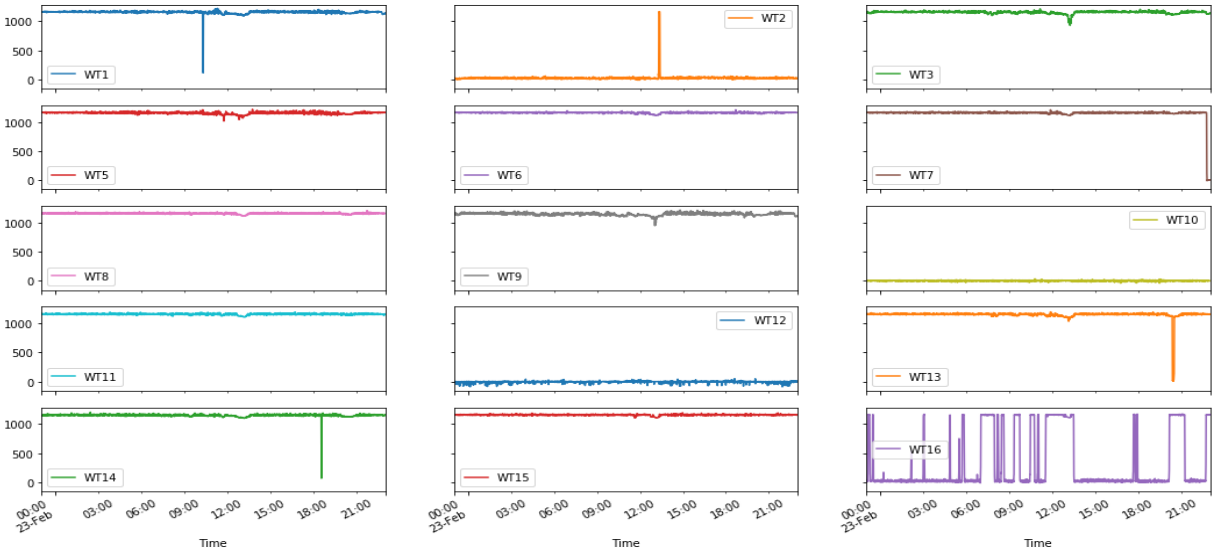


Figure A.2.1: Subplots of the generator speed (before any filtering) during the interval as stated in Table 5.4.

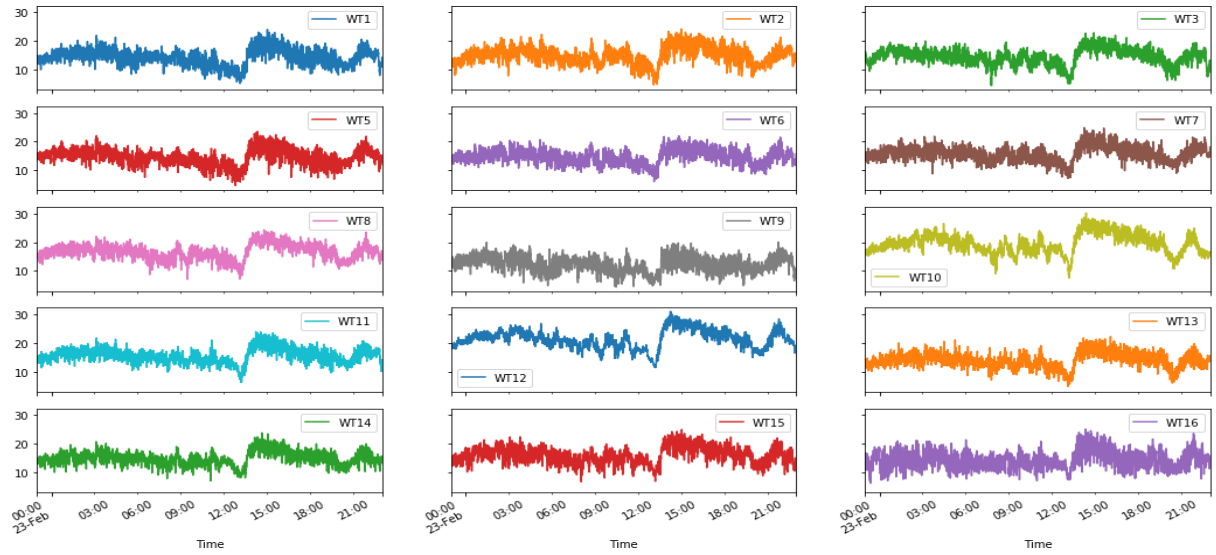


Figure A.2.2: Subplots of the wind speed [m/s] (before any filtering) measured at the nacelle during the interval as stated in Table 5.4. All experience quite similar behaviour except for turbine 10 and 12 which has slightly higher values.

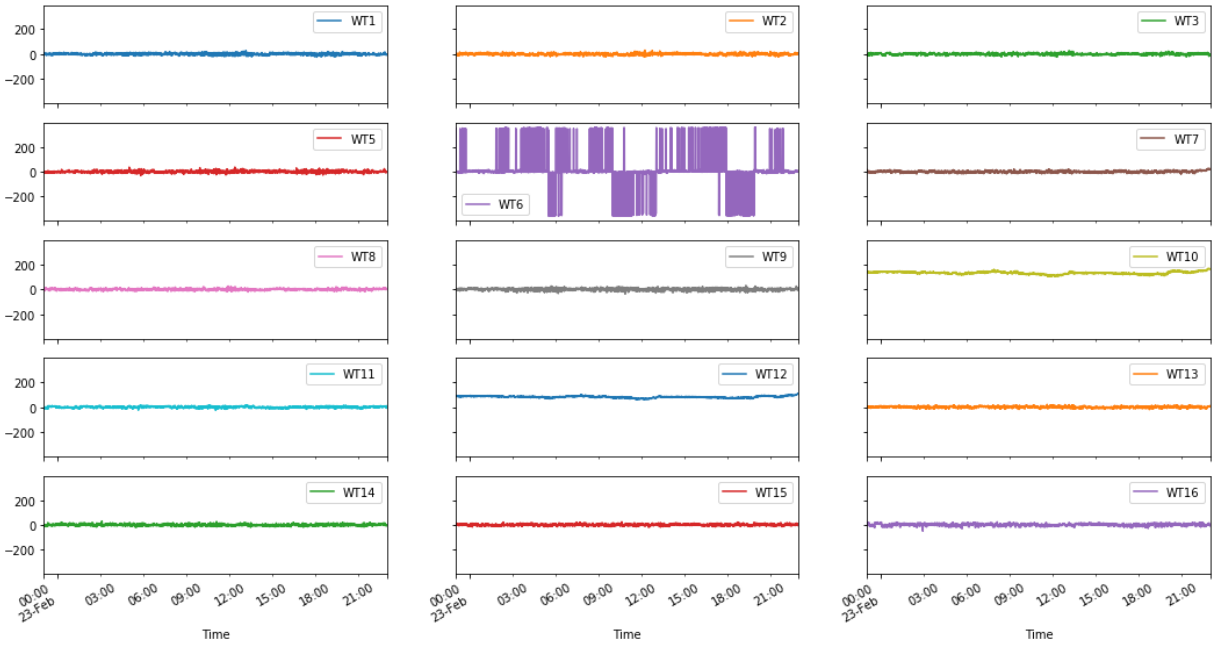


Figure A.2.3: Subplots of the positional deviations (before any filtering) during the interval as stated in table 5.4. The spikes indicate misinterpretation of the angles as the spikes are of negative -360 and 360 which is the same as saying the angle is 0. Before filtering, phase unwrapping is applied to the time series of position deviation. The resulting plot can be viewed in Figure A.2.4

Phase unwrapping of the position deviation

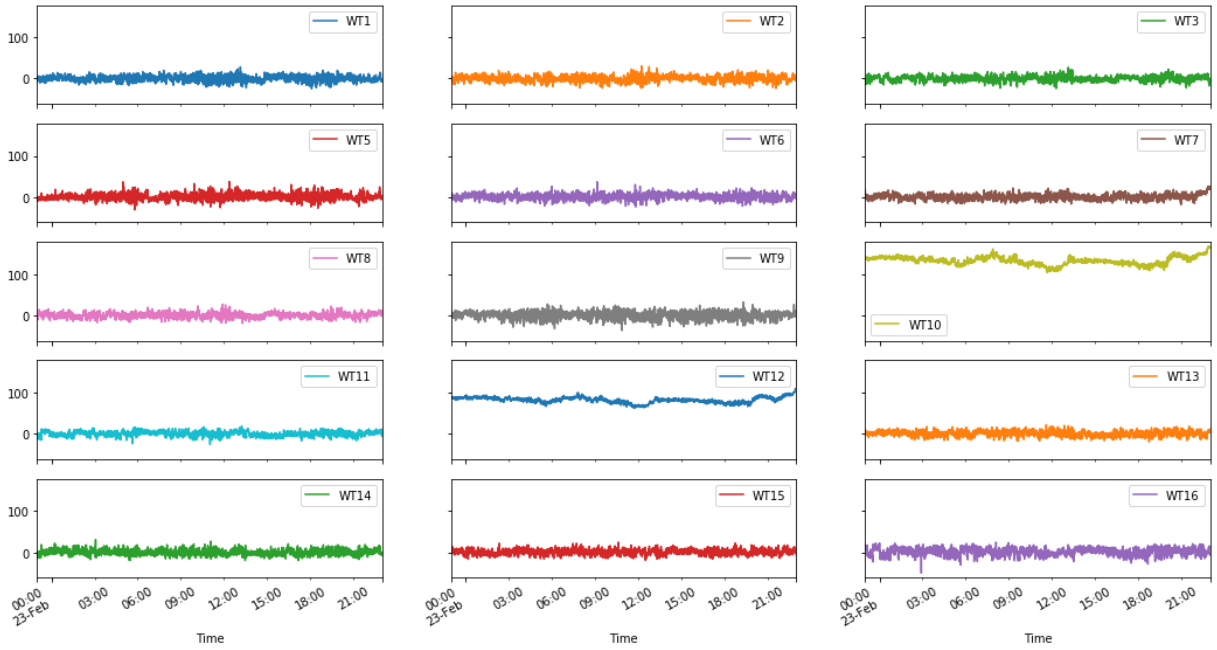


Figure A.2.4: Subplots of the positional deviations during the same interval as the time series in Figure A.2.3, only after phase unwrapping. It can be observed that the spikes in the time series associated with turbine 6, in the previous plot, do not appear in this one.

Comparison between the measured rotor and generator speed after filtering

As can be seen in these plots, the generator and rotor speed reflects the exact same property. Generator speed is originally measured where the rotor speed is an approximation of the generator speed. The only difference between them is the ratio, which is caused by the gearbox mounted between the rotor and the generator shafts. From now on, the generator speed will be used instead of the rotor speed. If the generator speed misses values, the time series (which are missing) for the generator speed will be substituted by the rotor speed time series, if available.

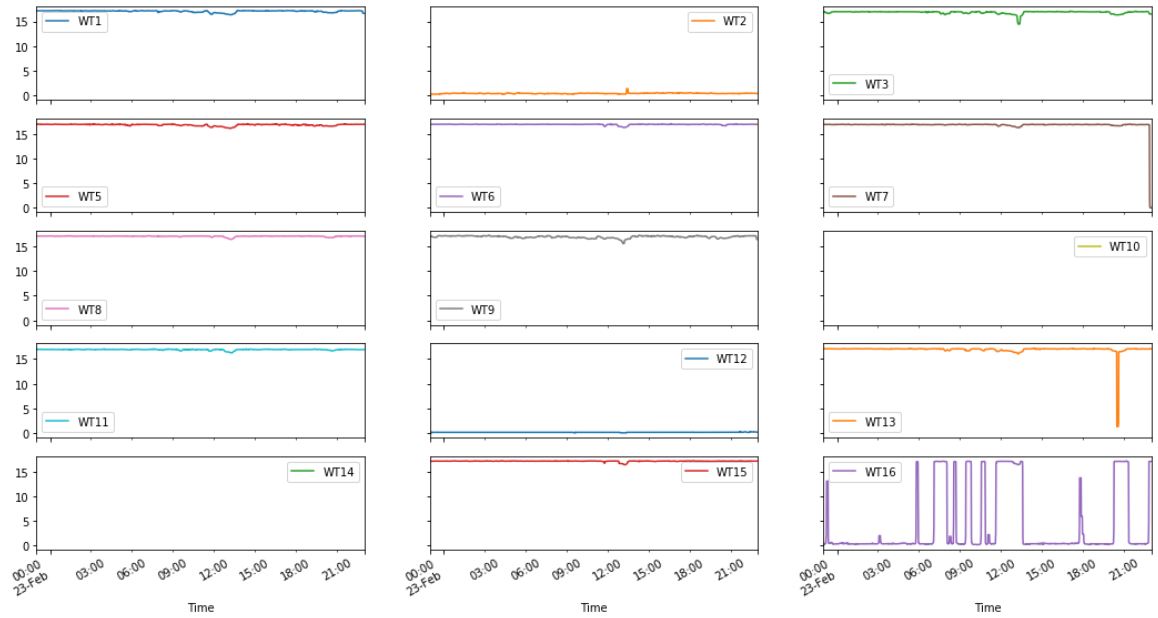


Figure A.2.5: Subplots of the rotor speed (after filtering) during the exact same interval as the generator speed provided in Figure A.2.6.

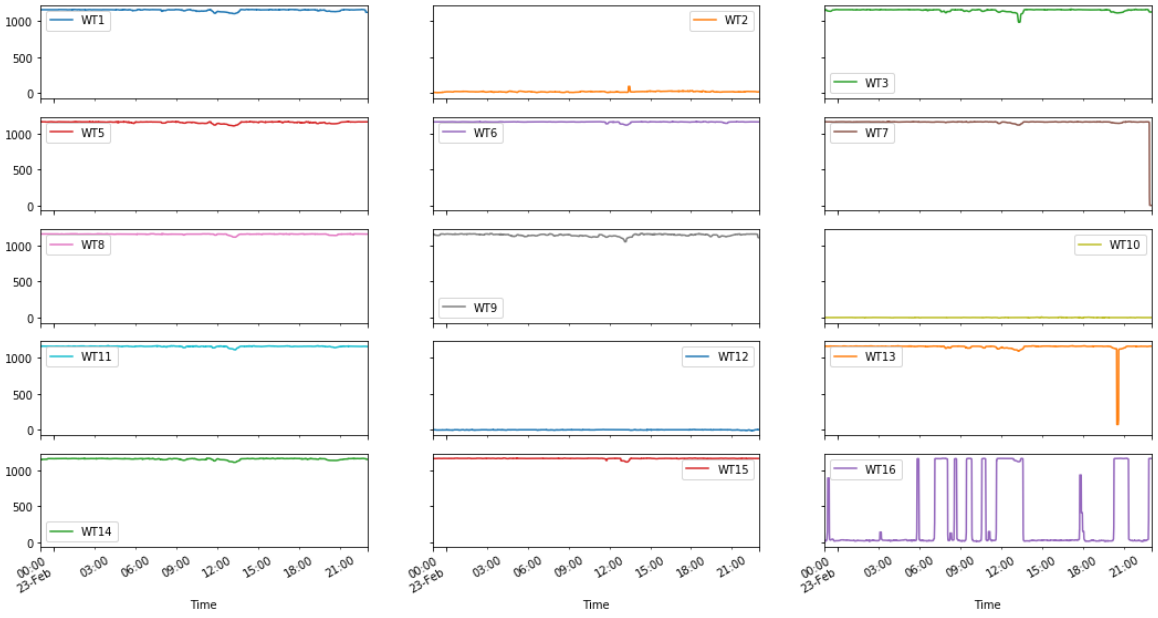


Figure A.2.6: Subplots of the generator speed (after filtering) during the exact same interval as the rotor speed provided in Figure A.2.5.

The remaining preprocessed time series

The consecutive figures presented in this subsection is the positional deviation after phase unwrapping and the wind speed after an SMM filter of length 5 is applied.

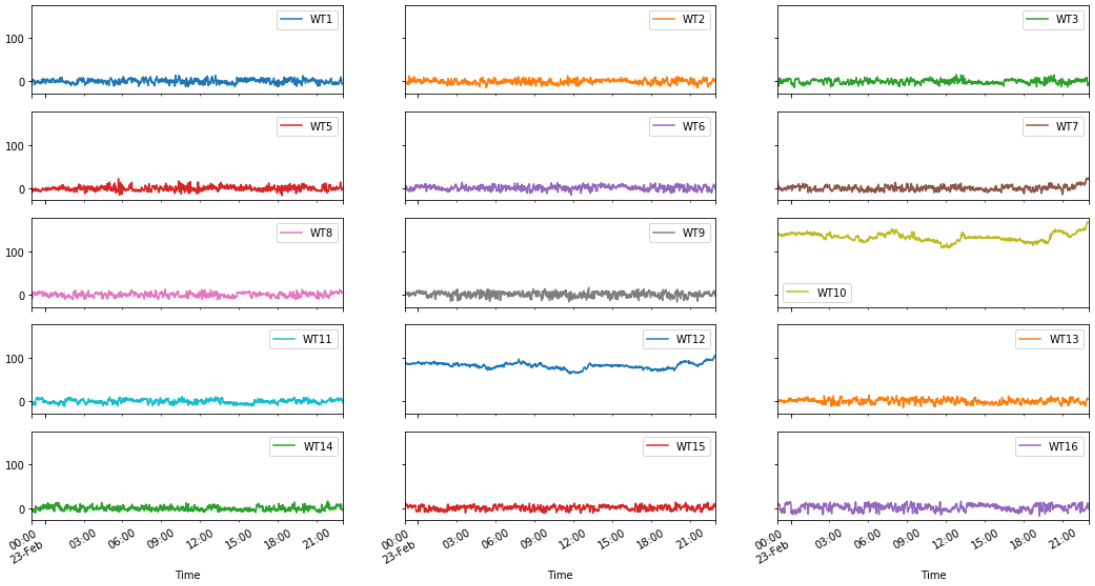


Figure A.2.7: Subplots of the positional deviations during the interval as stated in Table 5.4, only after phase unwrapping. Considerably less noise can be observed in these plots, compared the time series prior to filtering.

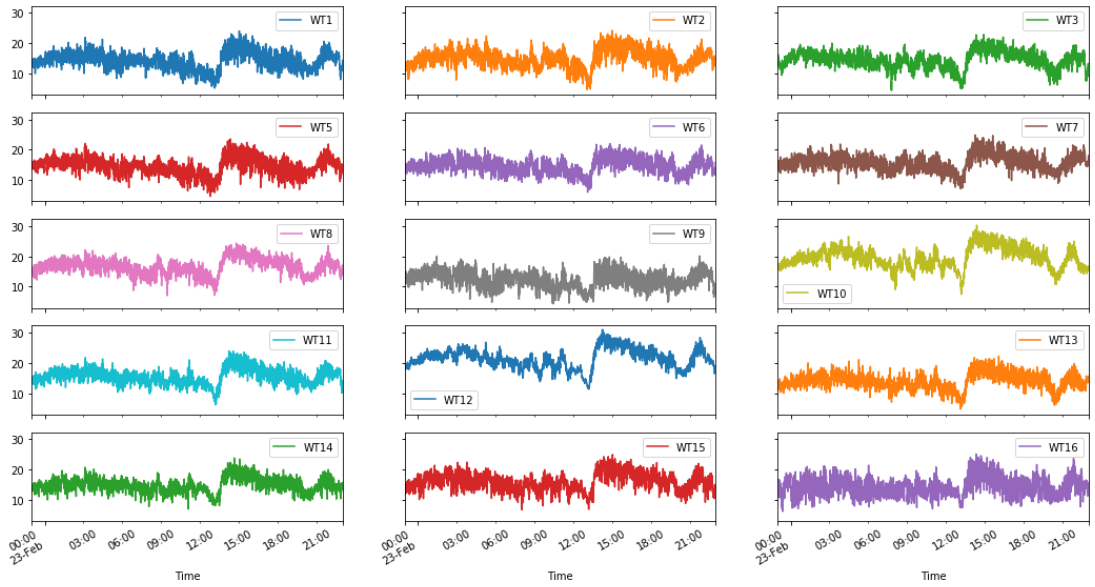


Figure A.2.8: Subplots of the wind speed [m/s] (after filtering) measured at the nacelle during the interval as stated in Table 5.4. All experience quite similar behaviour except for turbine 10 and 12 which has slightly higher values.

Table A.2.1: Descriptive statistics for the wind speed of Figure A.2.8.

	WT1	WT2	WT3	WT5	WT6	WT7	WT8	WT9	WT10	WT11	WT12	WT13	WT14	WT15	WT16
count	2881	2881	2881	2881	2881	2881	2881	2881	2881	2881	2881	2881	2881	2881	2881
mean	14.2	14.9	14.4	14.6	14.6	15.8	16.7	12.4	19.6	15.6	21.4	14.1	14.9	16.1	14.3
std	2.9	3.0	2.8	2.7	2.5	2.6	2.6	2.6	3.4	2.5	3.3	2.7	2.3	2.8	3.0

A.3: Multivariate_V2

In this section, the time series corresponding to the multivariate data set named 'Multivariate _V2' is presented. The time series presented are all filtered with an SMM filter of length 100. The specific parameters for extraction can be seen in Section 5.2.2. The univariate time series corresponding to power, gearbox temperature, generator speed, wind direction, wind speed and the external temperature can be seen in the consecutive figure (Figure A.3.1 to A.3.6).

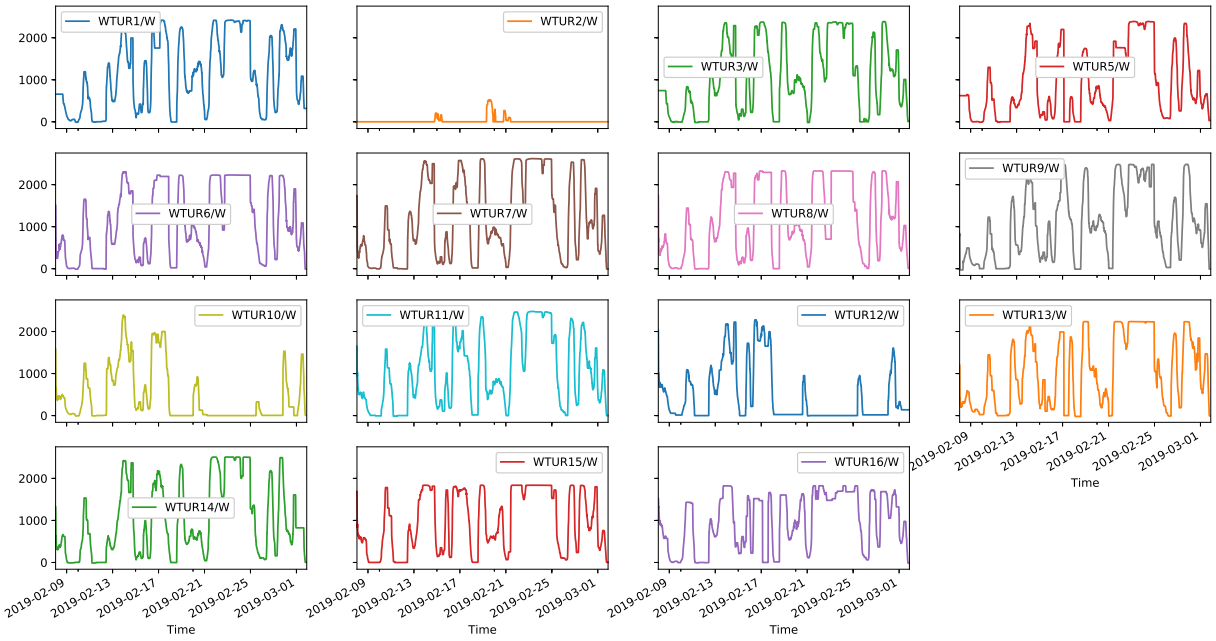


Figure A.3.1: Subplots of the power [W] for 'Multivariate_V2' post filtering. Length of the moving window was set to 100 and the missing values are handled with forwarding and backward filling.

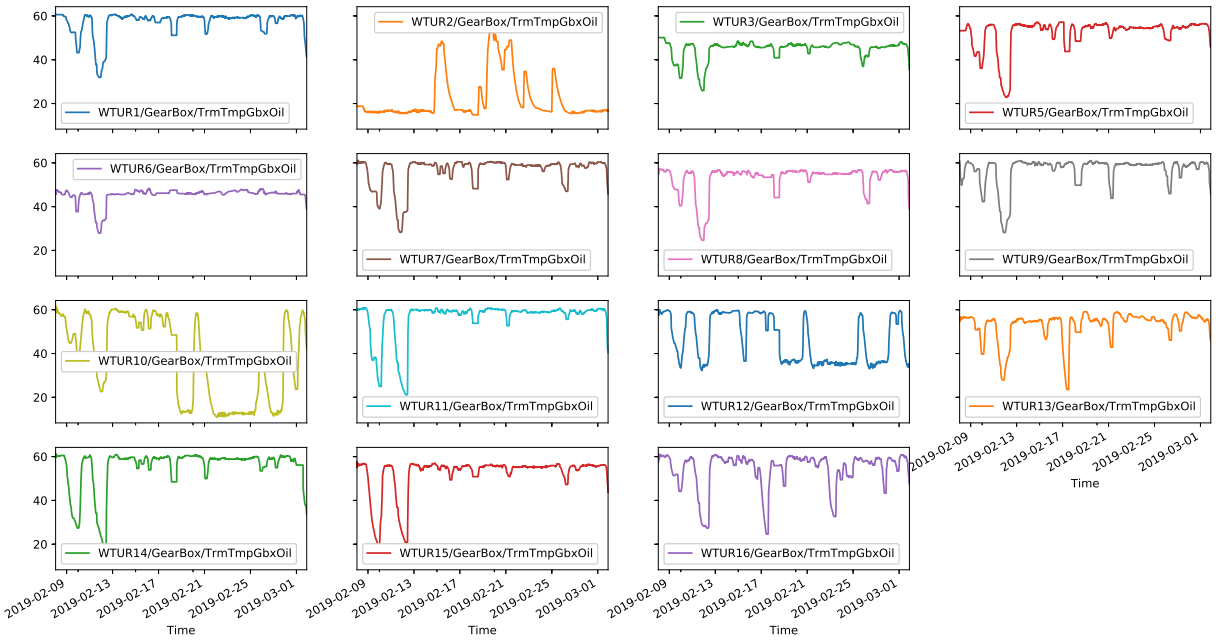


Figure A.3.2: Subplots of the gearbox temperature [degrees Celsius] for 'Multivariate_V2' post filtering. Length of the moving window was set to 100 and the missing values are handled with forwarding and backward filling.

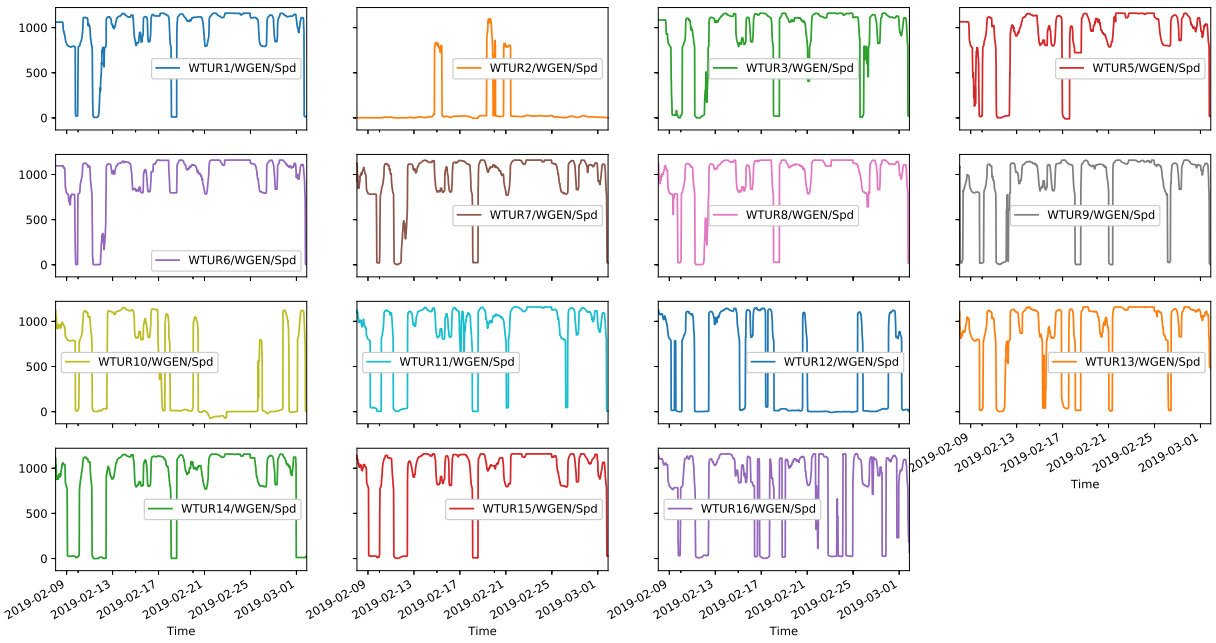


Figure A.3.3: Subplots of the generator speed [rpm] for 'Multivariate_V2' post filtering. Length of the moving window was set to 100 and the missing values are handled with forwarding and backward filling.

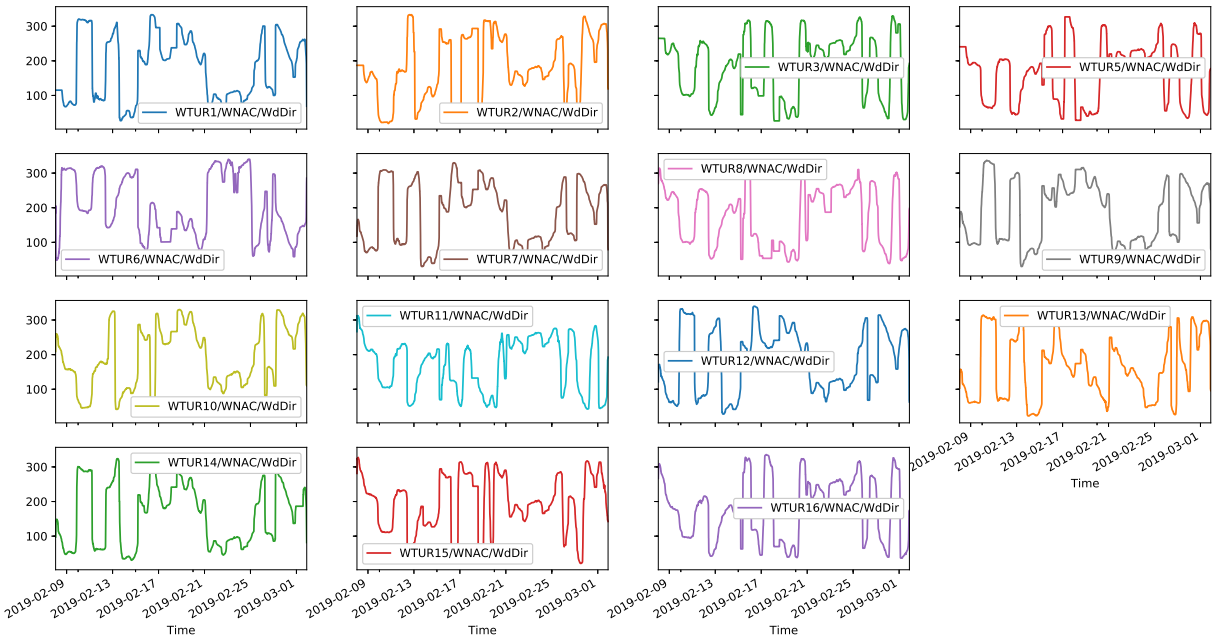


Figure A.3.4: Subplots of the wind direction [degrees] for 'Multivariate_V2' post filtering. Length of the moving window was set to 100 and the missing values are handled with forwarding and backward filling.

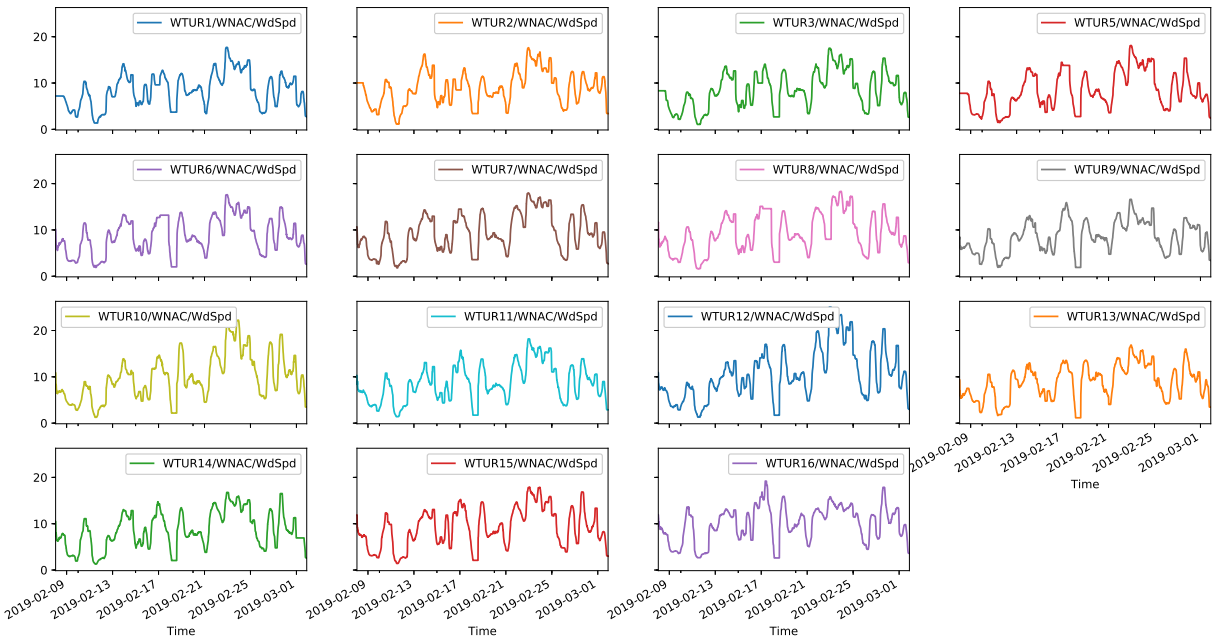


Figure A.3.5: Subplots of the wind speed [m/s] for 'Multivariate_V2' post filtering. Length of the moving window was set to 100 and the missing values are handled with forwarding and backward filling.

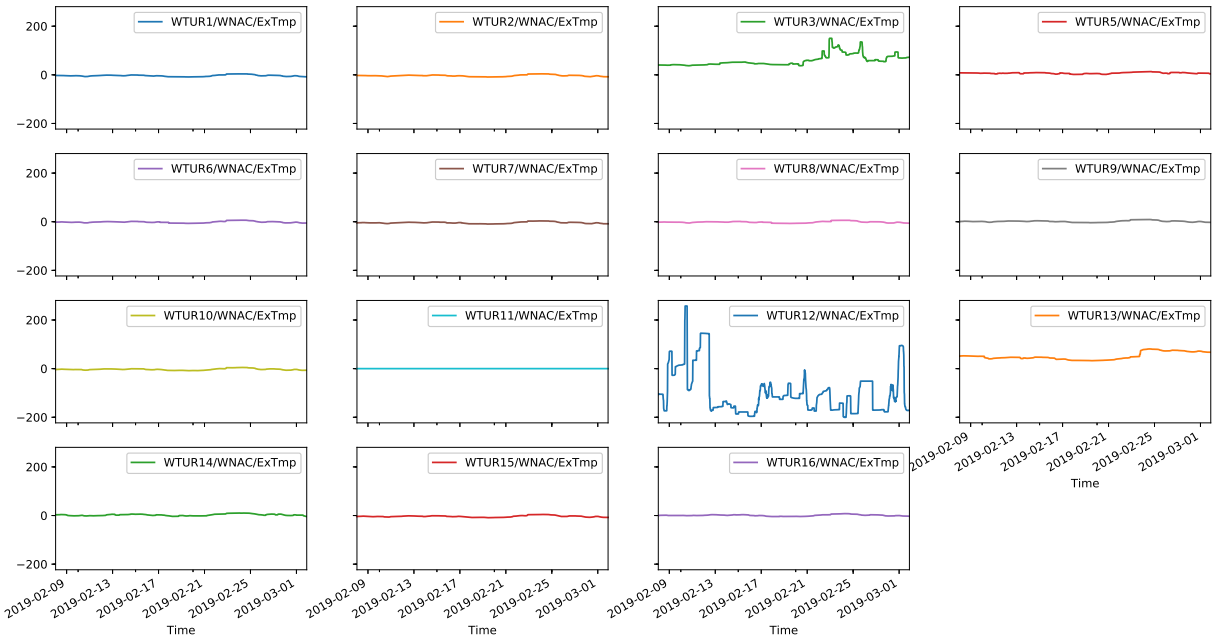


Figure A.3.6: Subplots of the outside temperature [degrees Celsius] for 'Multivariate_V2' post filtering. Length of the moving window was set to 100 and the missing values are handled with forwarding and backward filling.

Appendix B - Additional clustering results

Within Appendix B, additional clustering results for all data sets will be presented: Additional clustering results for clustering 'Univariate_V1', 'Univariate_V2', 'Multivariate_V1' and 'Multivariate_V2' is presented in Appendix B.1, B.2, B.3 and B.4, respectively.

B.1: Additional clustering results from clustering the 'Univariate_V1' data set

B.1.1: Similarity in shape - thetheDTW algorithm not constrained

The objective is now to cluster the time series with the objective of similarity in shape and not the absolute difference and offset. Therefore, the clustering analysis is performed on the normalised data set. The similarity measure used to calculate the dissimilarity matrix (DM) for all time series is now the dynamic time warping (DTW) measurement presented in Section 3.1. As DTW is not supported as a similarity measure for the hierarchical clustering algorithm the DTW distance between each time series are calculated first and then stored in a dissimilarity matrix (DM) (as in Equation. (3.3)). The DTW distance can be calculated by numerous libraries in Python. The library chosen for calculation of the condensed distance matrix between all time series was the library called *dtaidistance*. The calculation of the DTW will be unconstrained (i.e. Sakeo-Chiba band is not added). DTW has much higher complexity then Euclidean distance and the calculation of the condensed distance matrix took 46.3 seconds compared to Euclidean distance which took only 1.32 ms. The condensed distance matrix is furthermore fed to the hierarchical clustering algorithm. In order to find the appropriate linkage criterion and the optimal number of clusters, the internal indexes are analysed. The best method for hierarchical clustering algorithm will be found by comparing the *Cophenetic correlation coefficient* and the number of cuts to the dendrogram will be determined by viewing the internal performance indexes: *Silhouette index* and *MSSSE*. More information about these specific indexes can be found in Chapter 3. The summary plot and table for the internal indexes of cutting each dendrogram into *k* partitions - each with different linkage criterion - can be seen in Figure B.1.1 and Table B.1.1. The cophenetic correlation coefficient can be seen in the upper right corner in the aforementioned figure (Figure B.1.1).

Table B.1.1: Summary table with the linkage criteria: single, complete, average and ward. The data set which was clustered is 'Univariate_V1' with the objective of similarity in shape. The silhouette indexes and MSSSE values for a cut K - from 2 to 13 - is presented.

Method	Internal index	Number of cuts to the dendrogram											
		K = 2	K = 3	K = 4	K = 5	K = 6	K = 7	K = 8	K = 9	K = 10	K = 11	K = 12	K = 13
Single	Silhouette	0.65	0.57	0.55	0.43	0.11	0.15	0.14	0.17	0.15	0.12	0.06	0.07
	MSSSE	3.08	2.62	1.83	1.39	1.22	0.88	0.70	0.52	0.39	0.30	0.21	0.13
Complete	Silhouette	0.64	0.61	0.48	0.43	0.20	0.18	0.16	0.21	0.15	0.12	0.07	0.07
	MSSSE	3.05	2.29	1.85	1.39	1.18	0.96	0.82	0.51	0.39	0.30	0.23	0.13
Average	Silhouette	0.65	0.61	**0.55**	0.43	0.13	0.18	0.16	0.21	0.15	0.12	0.06	0.07
	MSSSE	3.08	2.29	**1.83**	1.39	1.19	0.96	0.82	0.51	0.39	0.30	0.21	0.13
Ward	Silhouette	0.61	0.58	0.48	0.43	0.20	0.21	0.21	0.21	0.15	0.12	0.07	0.07
	MSSSE	3.26	2.37	1.85	1.39	1.18	0.83	0.66	0.51	0.39	0.30	0.23	0.13

First of all, reviewing the cophenetic correlation coefficient in Figure B.1.1, it can be seen that the largest coefficient is observed for the dendrogram constructed with *average* as the linkage criterion. However, the others still have relatively high values close to 1, indicating that all the dendrograms represent a fairly high-quality solution. The dendrogram built with the linkage criterion set to *average* will be cut. Secondly, reviewing the summary table and the corresponding summary plot, it can be observed that the silhouette index is largest for a cut of two for both the average and single dendrograms, which have the highest cophenetic correlation coefficients. For both dendrograms, only a small reduction in the silhouette index is observed from going from a cut of two to a cut of three and four. The values do not decrease exponentially as it did with the objective of similarity in time. This is because the intercept is somewhat ignored (not the trend though). Reviewing the dendrogram with 'average' linkage criterion, the silhouette decreases slowly for a cut of 2, 3 and 4; where afterwards the silhouette index suddenly plunges. The corresponding MSSSE value decreases exponentially along with deeper cuts in the dendrogram. The values of the MSSSE values can be observed to be significantly smaller during this analysis, than in the cluster analysis with similarity in time as the objective. The reasoning for this is because the intercept is included in the analysis and similarity in time is a stricter measure of similarity. Furthermore the silhouette index and the MSSSE - during this analysis - indicates that a cut of four has a significant level of the silhouette index and a significant reduction

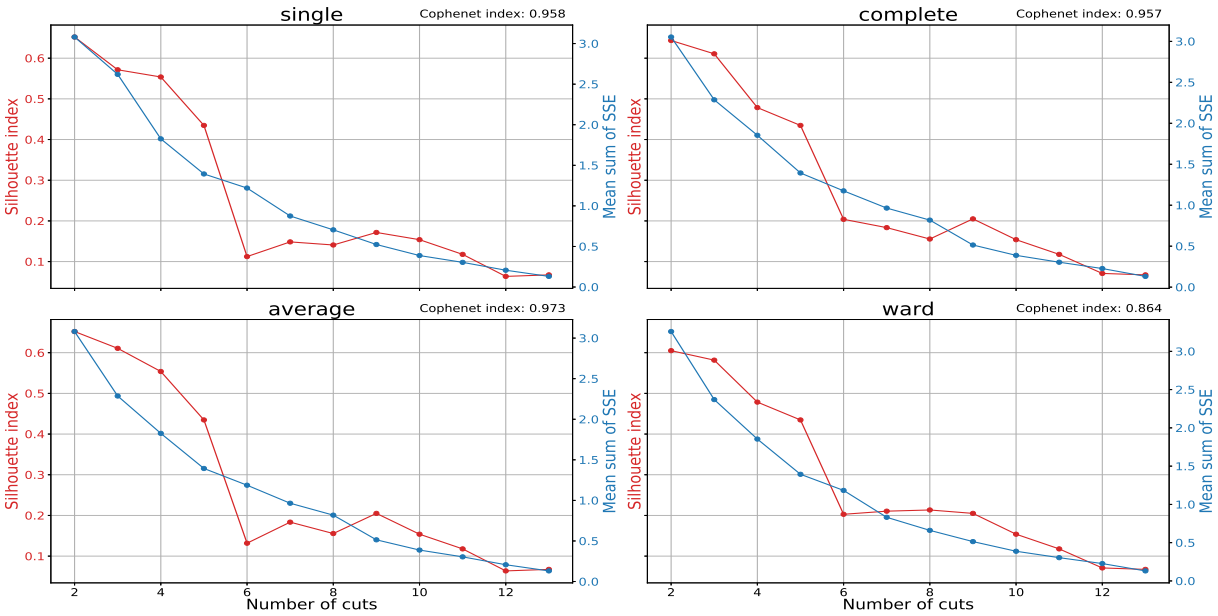


Figure B.1.1: Summary plots with the linkage criteria: single, complete, average and ward. The data set which was clustered is 'Univariate_V1' with the objective of similarity in shape. The silhouette indexes and MSSSE values for a cut K - from 2 to 13 - is presented along with the cophenetic correlation coefficient in the upper right corner.

in the MSSSE from the previous cuts. The best compromise between separation and similarity is, therefore, a cut of four the dendrogram built with average as the linkage criterion. Deeper cuts to the dendrogram significantly reduce the silhouette index value and an insignificant reduction in the MSSSE value. The optimal configuration for the current data set is chosen to be *average* as the linkage criterion and cutting the dendrogram such that it constructs four partitions. A deeper insight will be obtained by reviewing the corresponding dendrogram along with the assigned clusters.

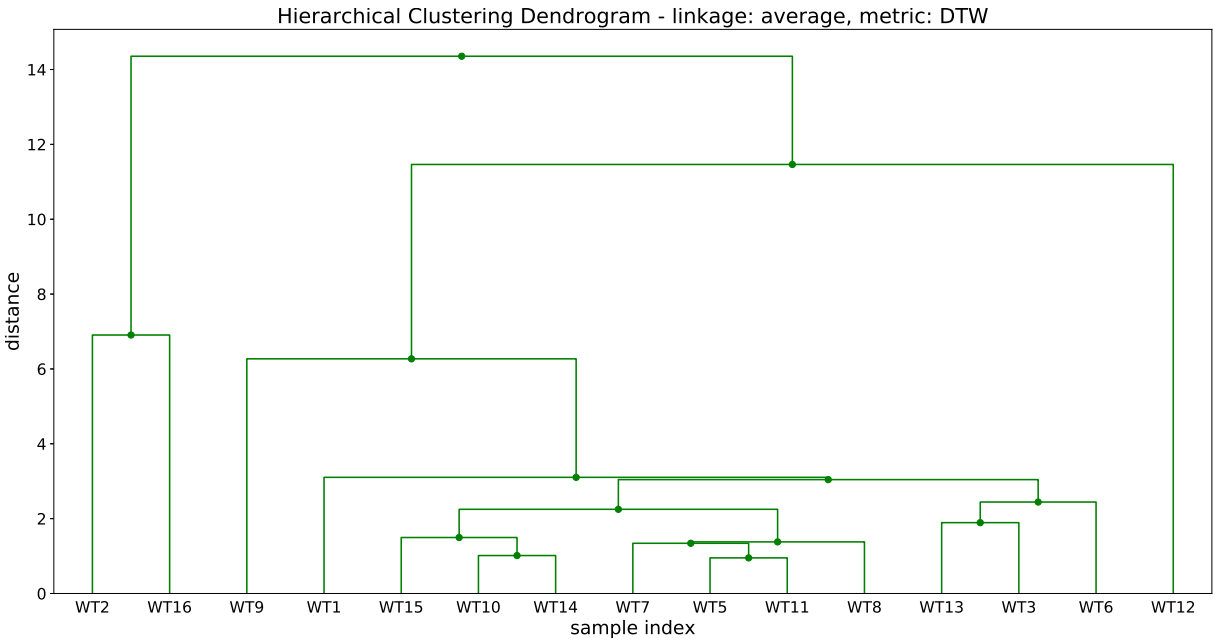


Figure B.1.2: Hierarchical clustering with DTW as similarity measures.

The resulting dendrogram - with *average* as the linkage criterion - can be viewed in Figure B.1.2. Viewing the dendrogram and the corresponding time series associated with each partition, a deeper cut of five to the dendrogram would result in a significant reduction in the average distance between each cluster. The difference can be seen by reviewing the time series in Figure B.1.3 where a cut of five manages to separate the most abnormal looking time series to those which experience similar behaviour. However, as the silhouette index value decreases quite significantly from a cut of four to a cut of five, a cut of five is disregarded. Therefore, the physical interpretation of a cut of four will be examined in greater detail in the consecutive section.

B.1.2: Analysing the clustering results from similarity in shape where DTW algorithm is not constrained

The corresponding dendrogram with the objective of clustering the time series with respect to similarity in shape can be seen in Figure B.1.3. The absolute difference and the offset will be ignored to a certain degree, as normalising the data set will remove the offset and scale the data between 0 and 1 (assuming there are no negative values). Clustering with the objective of similarity in shape manages to separate turbines with different behaviour pattern and manages to group time series subjected to shifts in time. For instance, a cosine wave and a sinus wave are pretty similar when using DTW, but are drastically different when using Euclidean. In other words, these are similar in shape but not in time.

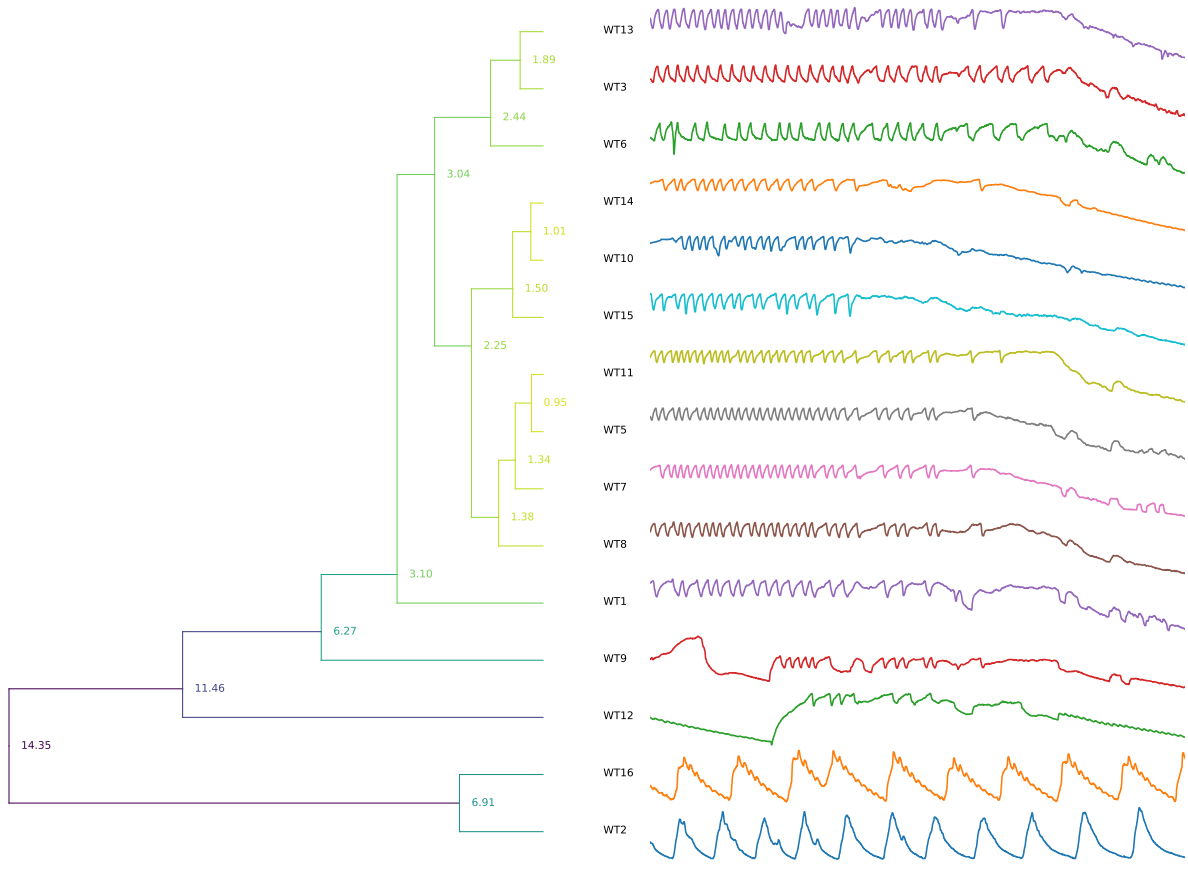


Figure B.1.3: Summary of hierarchical clustering with DTW as similarity measure and the linkage criterion set to 'average'

Initially, we can observe that using DTW as the distance measure the clustering algorithm manages to group the most similar time series in shape and separate the most dissimilar time series. A cut of four separates the three most abnormal time series - time series WT2, WT12 and WT16 - into its own separate clusters (from now on called group A, B and C). A deeper cut of five would also separate WT9 into its own cluster which makes sense when looking at the first quarter of the interval. However, reviewing the remaining interval, the time series looks pretty similar to the rest. This can be verified by reviewing Figure B.1.3. Note that the size of the branches is not scaled properly, which is one of the limitations to the *dtaidistance* library. The number should be observed instead; it indicates the average distance when trees are merged together. Time series associated with turbine 2 and 16 are

merged at an average distance of 6.91, which is slightly higher than the merge of the time series associated with turbine 9 and the majority (currently assigned to group D), 6.27. The rest of the time series are clustered together within the fourth group, group D. Oscillations can be observed for all time series and is the results of the cooling pump which is enabled and disabled when hitting its corresponding threshold values. Analysis of the physical interpretation of the clustering assignment will be performed in the consecutive paragraphs.

Group A: Contains only the time series associated with wind turbine 2. The time series are characterised by medium frequency oscillations and constant mean throughout its entire length. The cooling pump enables when reaching the upper values, cooling it down to its minimum value. The time series does not experience a decreasing trend towards the end; the wind, on the other hand, experiences this trend and should result in lower rotations and lower temperature levels for the associated turbine. Since it does not follow the same trend as the local wind, it is a strong indication that the turbines operate unaffected by the wind. The only possible explanation for this is that the turbine is not rotating. This can be verified by reviewing the generator speed from the previous section.

Group B: Contains only the time series associated with wind turbine 12. The time series are characterised by quite abnormal behaviour pattern in the initial 5 hours (02:00 - 07:00), but pretty similar behaviour as the majority after this interval. As with the objective of similarity in time, the explanation for this is that the turbine experience period at the beginning where the generator/rotor is not rotating. The physical interpretation of this is not transparent in the time series for the wind speed or wind direction. One would expect that the wind during the initial period are too high such that the brakes are engaged, but this is not the case.

Group C: Contains only the time series associated with wind turbine 16. The time series are characterised by oscillation with a frequency close to half of the time series in group A. It also has a lower amplitude than the time series in group A. This might indicate that the settings for the cooling pump are different: Especially with regards to the enable and disable threshold values. This explains the slightly elevated temperature levels also but proves hard to verify as such information is not available. Other than that, the same interpretation as for turbine in group A holds: The turbine is not rotating as it does not follow the trend of its local wind conditions.

Group D: Contains the rest of the turbines. All these turbines experience similar behaviour: relatively high-frequency oscillation, roughly the same amplitude and a decreasing trend towards the end. This is highly correlated to the behaviour of the local wind conditions for each turbine. The hypothesis for these turbines is that they are actively rotating and could be further interpreted as operation under normal conditions. This is verified by viewing the generator speed for all turbines. One small exception is turbine 9 which have similar behaviour as the turbine in group B. Same physical interpretation as for group B holds for turbine 9.

B.2: Additional clustering results from clustering the 'Univariate_v2' data set

B.2.1: Similarity in shape - DTW algorithm not constrained

The objective is now to cluster the time series with regards to the objective of similarity in shape and not the absolute difference and offset. Therefore, the clustering analysis is performed on the normalised data set. The similarity measure used to calculate the dissimilarity matrix (DM) for all time series is now the dynamic time warping (DTW) measurement presented in Section 3.1. The DTW algorithm is implemented with no constraints. DTW has much higher complexity then Euclidean distance and the calculation of the condensed distance matrix took 44.3 seconds compared to Euclidean distance which took only 1.31 ms. The condensed distance matrix is furthermore fed to the hierarchical clustering algorithm and solved with the corresponding linkage criterion set. The optimal hierarchical clustering method and the optimal number of clusters will be found with the same internal indexes used so far: Cophenetic correlation coefficient, Silhouette index and MSSSE. More information about these specific indexes can be found in Chapter 3. The summary plot and table for the internal indexes of cutting each dendrogram - each with different linkage criterion - into k cuts can be seen in Figure B.2.1 and Table B.2.1. The cophenetic correlation coefficient can be seen in the upper right corner in the aforementioned figure (Figure B.2.1).

Table B.2.1: Summary table for each linkage criterion supported by the hierarchical clustering algorithm: single, complete, average and ward. The silhouette index and mean sum of the sum of squared error is plotted for each cut in the dendrogram.

Method	**Internal index**	**Number of cuts to the dendrogram**											
		K = 2	K = 3	K = 4	K = 5	K = 6	K = 7	K = 8	K = 9	K = 10	K = 11	K = 12	K = 13
Single	Silhouette	0.41	0.40	0.37	0.35	0.38	0.26	0.22	0.19	0.23	0.20	0.18	0.13
	MSSSE	5.89	5.08	4.37	3.81	3.26	2.65	2.31	1.64	1.11	0.82	0.56	0.33
Complete	Silhouette	0.31	0.36	0.35	0.36	0.38	0.25	0.23	0.19	0.21	0.20	0.18	0.13
	MSSSE	6.51	5.52	4.71	3.83	3.26	2.47	2.01	1.67	1.13	0.82	0.56	0.33
Average	Silhouette	0.41	0.40	0.37	0.36	**0.38**	0.26	0.24	0.20	0.23	0.20	0.18	0.13
	MSSSE	5.89	5.08	4.37	3.83	**3.26**	2.65	1.88	1.55	1.11	0.82	0.56	0.33
Ward	Silhouette	0.41	0.40	0.35	0.36	0.23	0.25	0.24	0.27	0.23	0.20	0.18	0.13
	MSSSE	5.89	5.08	4.54	3.83	2.96	2.39	1.88	1.44	1.11	0.82	0.56	0.33

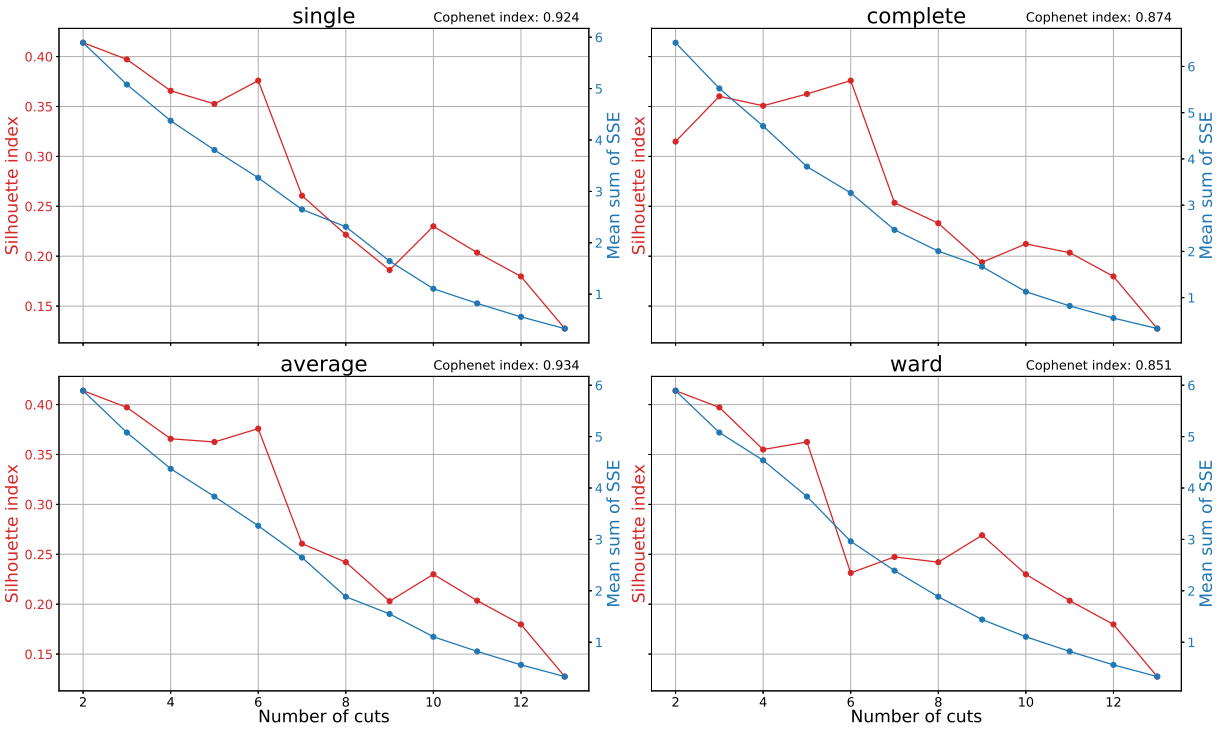


Figure B.2.1: Summary plot for each linkage criterion supported by the hierarchical clustering algorithm: single, complete, average and ward. The silhouette index and mean sum of the sum of squared error is plotted for each cut in the dendrogram.

First of all, reviewing the cophenetic correlation coefficient in Figure B.2.1, it can be observed that all dendrograms have relatively high coefficients close to 1 (all over 0.75). The dendrogram constructed with the linkage criterion set to *average* has the largest correlation coefficient which represents a dendrogram with a high-quality solution. The dendrogram built with average as the linkage criterion will, therefore, be cut and its assignments will be analysed. Secondly, reviewing the summary table and the corresponding summary plot, it can be observed that the silhouette index is largest for a cut of two; the silhouette index value is 0.41. The second and third highest peaks are at a cut of three and a cut of six. The silhouette index value for a cut of six is 0.38, which is only a 0.03 decrease from a cut of two. The corresponding MSSSE value decreases from 5.89 to 3.26, from a cut of two to a cut of six. The reduction is significant for the MSSSE index and only a small reduction in the silhouette index value is observed during this interval. Arguably, a cut of two, three and six could be chosen as the optimal number of cuts. The difference between the three different cuts is simply the ratio between separation and intra-cluster similarity: A cut of three has the largest separation between the clusters and a relatively large intra-cluster similarity; a cut of six has a smaller separation between the clusters, but also a relatively small intra-cluster similarity. The latter partitions time series based on less significant differences and would be more interesting to analyse in terms of acquiring physical interpretations and finding time series which are behaving slightly different from the rest. Therefore, a cut of six to the dendrogram is chosen as the best compromise between separation and intra-cluster similarity. The optimal configuration for the data set is to cluster the data set with average as the linkage criterion and cut the dendrogram to form six clusters. The dendrogram - with average as the linkage criterion - along with visualisation of the corresponding time series can be seen in Figure B.2.2 and B.2.3 (in the consecutive section), respectively.

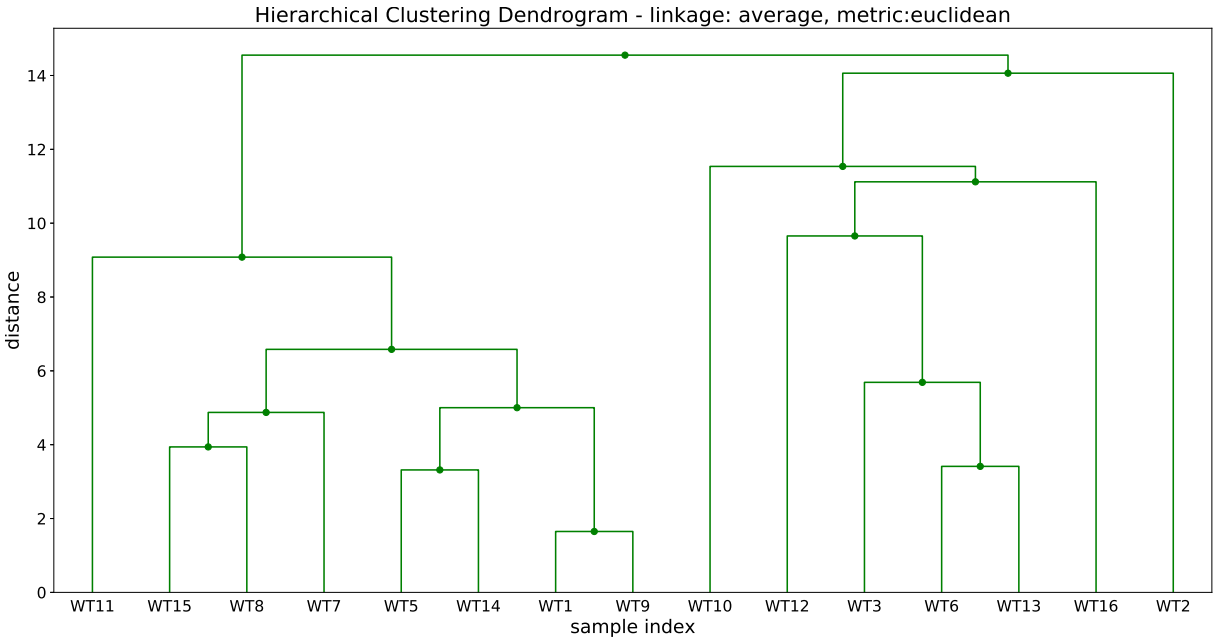


Figure B.2.2: Dendrogram with average as linkage criterion and distance as the DTW distance implemented on 'Univariate_V2' data set.

B.2.2: Analysing the clustering results from similarity in shape where the DTW algorithm is not constrained

In this section, the clustering results from clustering 'Univariate_V2' data set in Section 6.2.3 will be analysed. The corresponding dendrogram with the objective of clustering the time series with respect to similarity in shape can be seen in Figure B.2.2 or B.2.3. The absolute difference and the offset will be ignored to a certain degree, as normalising the data set will remove the offset and scale the data between 0 and 1 (assuming there are no negative values). From analysing the internal indexes in the previous section, best cut to the dendrogram was found to be a cut of six.

Initially, we can observe that when using DTW as the distance measure for calculating the dissimilarity matrix it manages to group the most similar time series in shape and separate the most dissimilar ones from the rest. This is especially true for a cut of two to dendrogram with 'average' as the linkage criterion. This clearly separates the time series experiencing rather stationary periodically oscillations of higher frequency to the one experiencing quite different behaviour. A much more detailed separation which results in only similar time series within each cluster is achieved by cutting the dendrogram in six partitions, as justified in the previous section. A cut of six separates - from top to bottom in Figure B.2.3 - WT12, WT16, WT10 and WT2 into its own cluster containing only itself. The grouping - from top to bottom - are now the following: Group A: {WT13, WT6, WT3}; Group B: {WT12}; Group C: {WT16}; Group D: {WT10}; Group E: {WT2}; Group F: {WT14, WT5, WT9, WT1, WT15, WT8, WT7, WT11}. The plots for the generator speed, position deviation and the wind speed is presented in Section 5.2.1. The summary table for all turbines with descriptive statistics is presented in the analysis of similarity in time in Table 6.5. These parameters will be used to interpret the physical meaning behind the resulting partitioning. The groupings will now be analysed and interpreted in the following paragraphs.

Group A: Contains time series from turbine 3, 6 and 13. These are characterised by low to no oscillations with small fluctuation in the signal. The cooling pump - which is causing the oscillations - does not seem to be engaged during the entire interval. The temperature remains relatively constant through the interval, with the exception of one bump after the midpoint. The bump can be explained by viewing the time series for the wind speed in Figure A.2.8 which experience a decrease and right after an increase in wind speed (right after the midpoint). The lack of oscillatory behaviour could be explained either that the threshold values for the cooling pump are very small or that the gearbox temperature reaches its maximum value, which is lower than the upper threshold for the cooling pump (i.e. when the cooling pump should be activated). The latter could be further explained by that the parts within the gearbox are less worn out or better lubricated, such that the gearbox temperature will not exceed this threshold value for the pump regardless of rotor/generator speed. The former hypothesis is most likely not correct

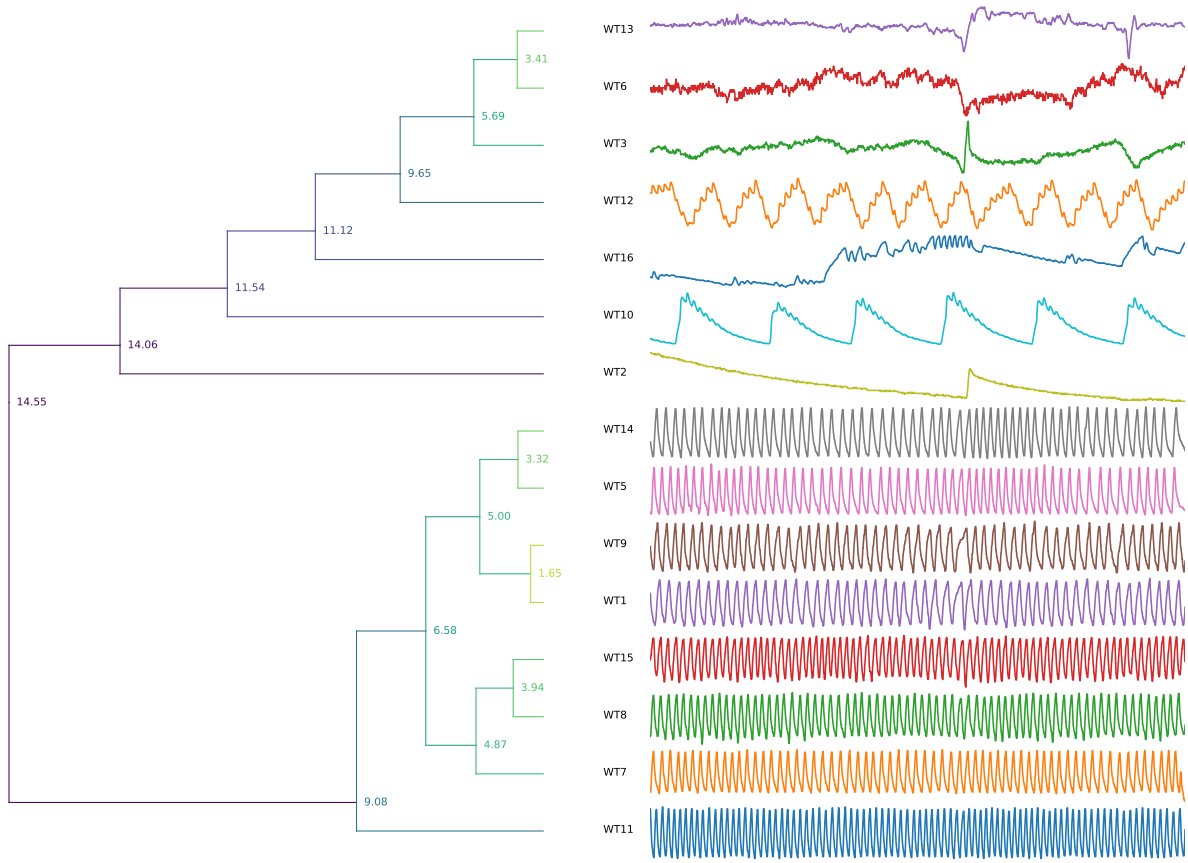


Figure B.2.3: Summary of hierarchical clustering with DTW as similarity measure. Average is chosen as the linkage criterion.

as the overall variance is quite large. The latter hypothesis - as before - proves hard to verify.

Group B: Contains only the time series corresponding to turbine 12. The time series is characterised by low-frequency oscillations. The signal also contains a higher frequency signal component with much lower amplitude. The cooling pump enables when it reaches the upper value and then cools the gearbox down until it reaches the minimum value (both values is set by the parameters for the cooling pump). The wind experiences sudden decrease and then increase right after the midpoint; the time series does not indicate any changes after the midpoint due to this. This indicates that the gearbox temperature is not dependent on the wind speed. The only case in which this is true is when the turbine is not rotating. As with the objective of similarity in time, the wind speed was found to be of critically high values, indicating that the brakes must be engaged.

Group C: Contains only the time series corresponding to turbine 16. The turbine has quite varying gearbox temperature and none to little oscillations. Intervals where the gearbox temperature slightly decreasing in a 'straight' line indicates the intervals where the turbine is not rotating. Same interpretation as for analysis of similarity in time applies: the temperature varies with the generator speed. However, the generator does not seem to stop because of the lack of wind - or position deviation different from zero - but from different reasons. Reasons could be the malfunctioning of the brakes or the malfunctioning of the pitch controller of the blades or something completely else. These assumptions prove hard to verify and additional maintenance might be required.

Group D: Contains only the time series corresponding to turbine 10. It characteristics are similar to that of the time series assigned to group B but the frequency for which it oscillates is decreased by a factor of three. Same interpretation as for group B applies to this: the temperature does not follow the same trend as the wind speed. However, after reviewing the wind speed, it is clear that the wind speed experience critically high levels, indicating that the brakes have to be engaged.

Group E: Contains only the time series corresponding to turbine 2. No periodic oscillations are observed, but it is affected by the change in the wind slightly after the midpoint. Reviewing the generator speed it becomes clear that during that period it experiences a peak value where the turbine is actually rotating. This explains the sudden increase in temperature levels observed in the gearbox. During the remainder of the interval the turbine is not rotating or increasing its temperature (rather trying to reach its equilibrium value with the temperature outside). No oscillatory behaviour is present which might indicate that the thresholds for the cooling pump are not exceeded.

Group F: Turbines associated with this group contains the majority of the turbines and could indicate normal operational behaviour for rotating turbines. The assigned wind turbines within this group are actually the same assignment - with the exception of turbine 11 and 13 - formed from clustering with the objective of similarity in time. Turbine 13 was not found to be of any interest for the analysis of similarity in time. But during this analysis, it can be seen that the shape is quite different; it does not experience the same periodic oscillations as the turbines in this group (group F). The faster the turbines rotate the more heat they produce which requires more cooling. The upper threshold for the cooling pump is reached at the peaks and the cooling pump enables until it reaches the lower threshold. This explains the higher frequency oscillations observed for the temperatures. The fact that it reaches the upper threshold very often suggest that the somewhere in the gearbox additional heat is produced because of poor lubrication or worn out parts. Viewing the dendrogram, a deeper cut would separate turbine 11 from this group (group F). The time series of the gearbox temperature corresponding to this has a larger frequency than the rest of group F, despite having the same rotational speed and wind conditions. A possible explanation for this is that the component in the gearbox of turbine 11 is even more worn out or worse lubrication than the remaining turbines in this group. If this is the case, the temperature increases faster along with an appropriate reaction of the cooling pumps(i.e. faster cooling).

B.3: Additional clustering results from clustering the 'Multivariate_v1' data set

B.3.1: Similarity in shape - DTW algorithm not constrained

The objective is now to cluster the time series with the objective of similarity in shape and not the absolute difference and offset. Therefore, the clustering analysis is performed on the normalised (and partly scaled) data set. The similarity measure used between time series is the dynamic time warping (DTW) measurement presented in Section 3.1. Global restrictions on the DTW algorithm is not implemented (i.e. unconstrained case). DTW has much higher complexity then Euclidean distance and the calculation of the condensed distance matrix takes d additional time: d refers to the dimension of the multivariate data set. The calculation of the dissimilarity matrices took 161.21 seconds (78.64 seconds parallelization with two cores). The condensed distance matrix is then fed to the hierarchical clustering algorithm. The optimal hierarchical clustering method and the optimal number of clusters will be found with the same internal indexes used so far: Cophenetic correlation coefficient, Silhouette index and MSSSE. More information about these specific indexes can be found in Chapter 3. In the first runs, the alphas are all equal to one, which weight the different univariate time series equally. The summary plot and table for the internal indexes of cutting each dendrogram - each with different linkage criterion - into k cuts can be seen in Figure B.3.1 and Table B.3.1. The cophenetic correlation coefficient can be seen in the upper right corner in Figure B.3.1.

Table B.3.1: Summary table with the linkage criteria: single, complete, average and ward. The data set which was clustered is 'Multivariate_V1' with the objective of similarity in shape. The silhouette indexes and MSSSE values for a cut K - from 2 to 13 - is presented. Alpha for all univariate data sets is equal to one.

Method	**Internal index**	**Number of cuts to the dendrogram**											
		K = 2	K = 3	K = 4	K = 5	K = 6	K = 7	K = 8	K = 9	K = 10	K = 11	K = 12	K = 13
Single	Silhouette	0.62	0.57	0.53	0.30	0.20	0.14	0.12	0.07	0.09	0.08	0.07	0.05
	MSSSE	4.47	3.62	3.17	2.47	2.19	1.93	1.65	1.41	1.08	0.85	0.59	0.39
Complete	Silhouette	0.64	0.57	0.53	0.30	0.20	0.22	0.17	0.11	0.09	0.09	0.07	0.05
	MSSSE	4.45	3.62	3.17	2.47	2.19	1.82	1.58	1.33	1.08	0.82	0.59	0.39
Average	Silhouette	0.64	0.57	**0.53**	0.30	0.20	0.22	0.17	0.11	0.09	0.09	0.07	0.05
	MSSSE	4.45	3.62	**3.17**	2.47	2.19	1.82	1.58	1.33	1.08	0.82	0.59	0.39
Ward	Silhouette	0.64	0.57	0.34	0.30	0.21	0.22	0.17	0.15	0.15	0.09	0.07	0.05
	MSSSE	4.45	3.62	2.91	2.47	2.13	1.82	1.58	1.33	1.08	0.82	0.59	0.39

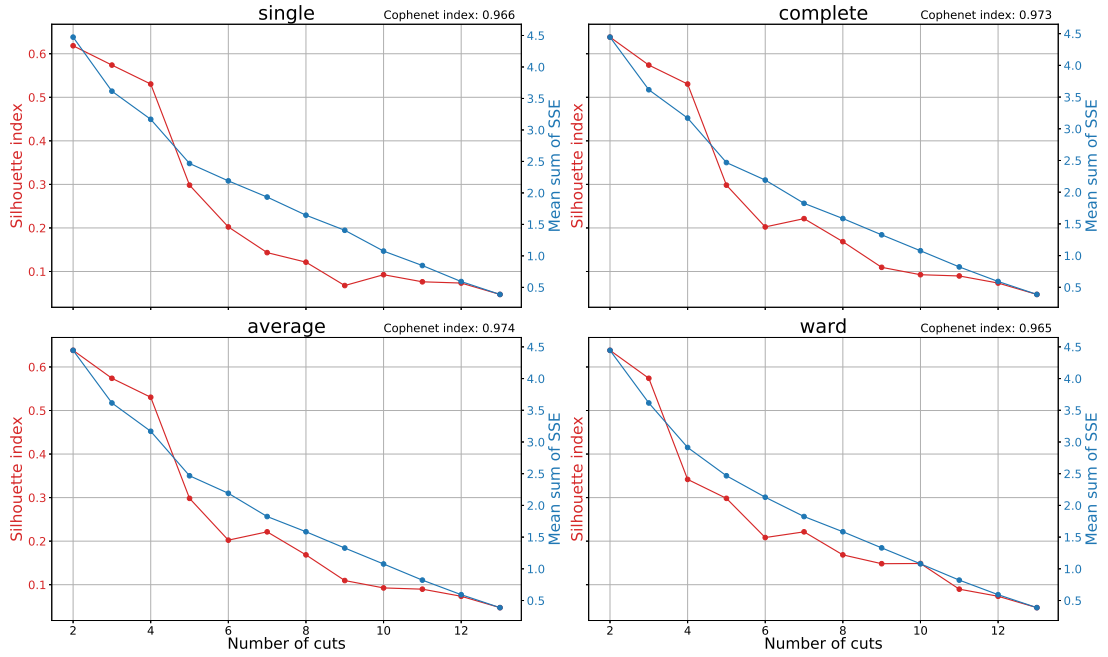


Figure B.3.1: Summary plots with the linkage criteria: single, complete, average and ward. The data set which was clustered is 'Multivariate_V1' with the objective of similarity in shape. The silhouette indexes and MSSSE values for a cut K - from 2 to 13 - is presented along with the cophenetic correlation coefficient in the upper right corner.

First of all, reviewing the cophenetic correlation coefficient in Figure B.3.1, it can be observed that all of the dendrograms have values equal or above 0.965. These values are very close to 1 and indicate a high-quality solution. The dendrogram with the highest cophenetic index is the dendrogram constructed with the linkage criterion set as *average*. Therefore, the dendrogram built with the linkage criterion set as *average* will be cut and its assignment will be analysed. Secondly, reviewing the summary table and the corresponding summary plot, it can be observed that the silhouette index is largest for a cut of two and decreases quite slowly up to a cut of four where it suddenly decreases quite rapidly. The corresponding MSSSE value during these periods decreases relatively steep. A fair reduction in the MSSSE value can be observed by cutting the dendrogram into four partitions; the corresponding silhouette value still remains relatively high. Further reduction will decrease the silhouette index value too much. Therefore, a cut of four seems to be the optimal configuration where the compromise between separation and similarity is at its best. A cut of four will be further analysed in the consecutive section. The corresponding dendrogram - with *average* as linkage criterion - can be seen in Figure B.3.2.

B.3.2: Analysing the clustering results from similarity in shape where the DTW algorithm is not constrained

In this section, the clustering results from clustering 'Multivariate_V1' data set in Appendix B.3.1 - with respect to the objective of similarity in shape - will be analysed. The corresponding dendrogram which will be cut can be seen in Figure B.3.2. In Figure B.3.3, the cluster assignment and the time series are visualised for a cut of four. As before, the summary of the gearbox temperature, generator speed, wind speed and position deviation (difference in angle between the direction of the wind and the direction of the nacelle) can be seen in Table 7.2 in Section 7.1.2.

Initially, one can observe that the turbines are separated well with respect to their overall similarity in shape. As with the objective of similarity in time, most of the dissimilarity between the clusters can be observed in the time series for the gearbox temperature. Cluster 1 can be observed to contain three (WT3, WT6, WT13) time series which are quite different in shape, compared to the rest. Same explanation for similarity in time holds here: The turbines have very similar shapes when reviewing the wind speed, generator speed and position deviation. Actually, cutting the current dendrogram a cut deeper (i.e. number of cuts equal to five) would separate the three turbines within cluster 1 into its own separate cluster. However, this is not indicated by the quite significant reduction in the silhouette index from a cut of four to a cut of five; but it is indicated by a significant decrease in the MSSSE value. More interestingly, the alpha values for weighing the different univariate data sets - which the multivariate data set is comprised of - can be modified. If we want stricter clustering of the gearbox temperature

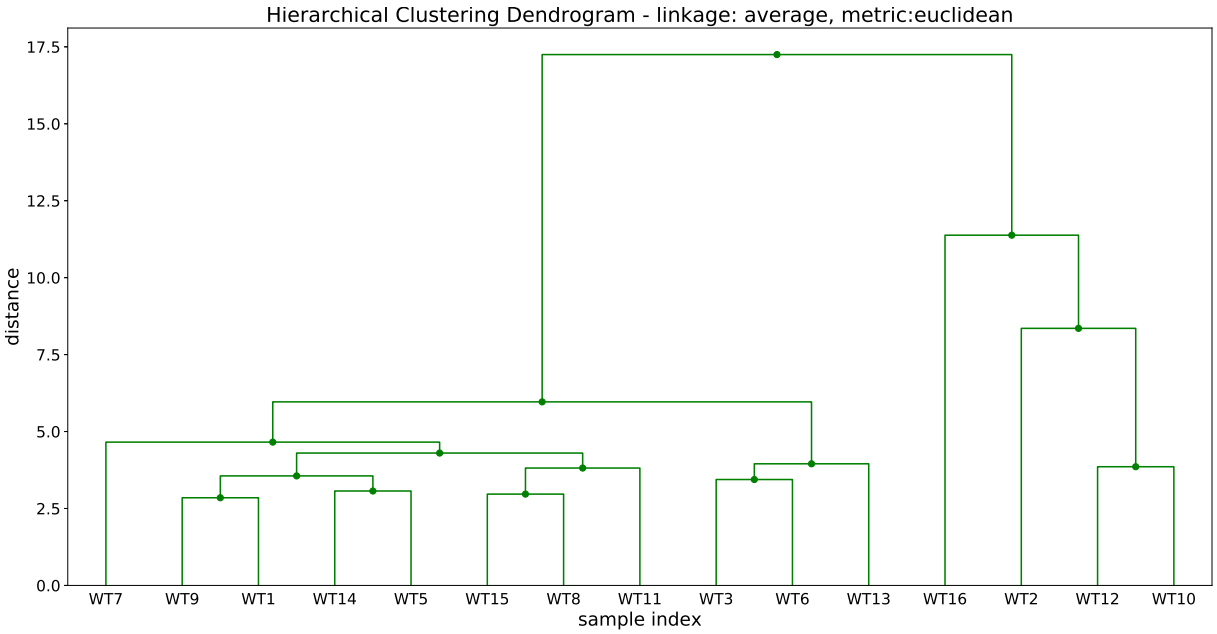


Figure B.3.2: Dendrogram with average as the linkage criterion and distance as DTW distance implemented on 'Multivariate_V1' data set. The weighting vector α is set to 1 for all elements.

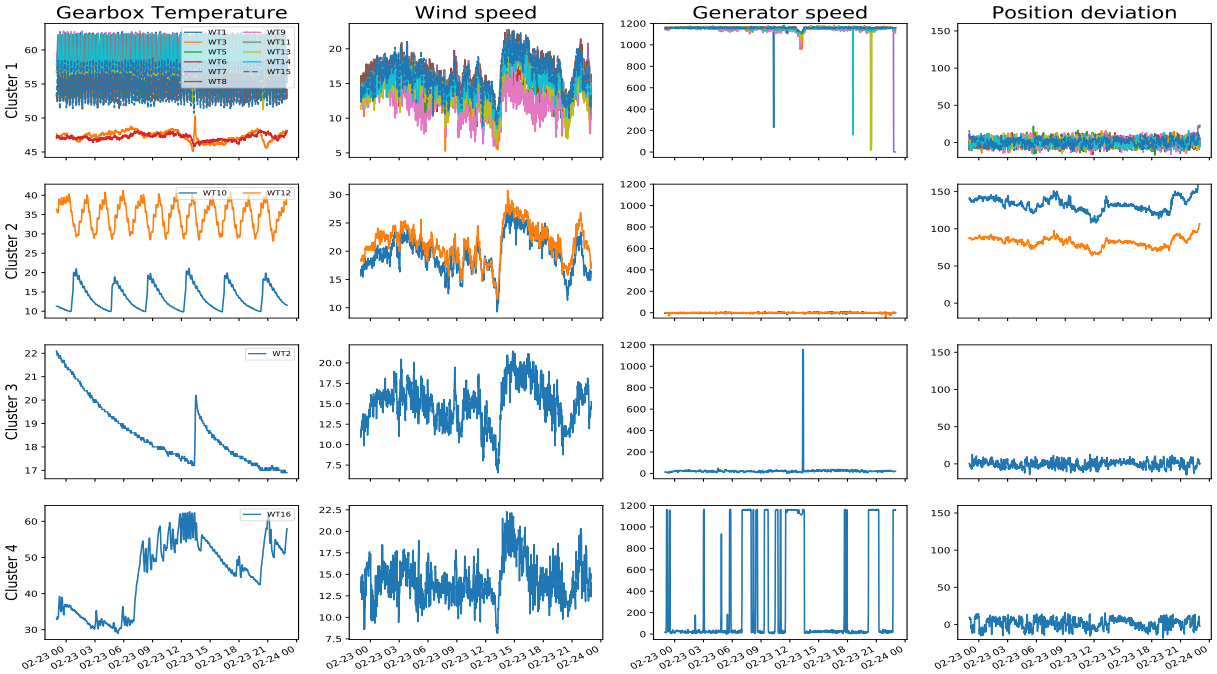


Figure B.3.3: Clustering results from the 'Multivariate_V1' data set with DTW distance. Summary plot of a cut of four to the dendrogram with the linkage criterion set as 'average'.

one could increase the corresponding alpha value associated with it. This is illustrated in Appendix B.2 and shows an optimal configuration which cuts the dendrogram into five partitions; this splits turbine 3, 6 and 13 from the current cluster 1 to an own separate cluster. Nonetheless, the physical interpretation of the underlying turbines within each cluster for the current cluster assignment seen in Figure B.3.3 will be further analysed in the next paragraphs.

Cluster 1: Gearbox temperature oscillates quite heavily with much higher frequency than the rest. These time series are characterised by high generator speeds and a position deviation of zero. As the majority of the clustered

turbines is assigned to this cluster - and their behaviour is expected - these turbines are assumed to be under normal operational conditions. However, there are three outliers which can be observed in the plots for the gearbox temperature: turbine 3, 6, and 13. These are very similar to the others with respect to all traits but the gearbox temperature; the time series for the gearbox does not experience similar oscillations. Oscillations are caused by the activation and deactivation of the cooling pump. The lack of oscillations might indicate some fault in the cooling system. However, the temperature levels are not critical: Based on this, the settings for the cooling system might be different from the others.

Cluster 2: This cluster contains both turbine 10 and 12. These turbines are characterised by a gearbox temperature with a much lower frequency than those of cluster 1. The position deviation is nonzero - which means that the turbine is not directed against the wind - and the time series associated with the generator speed shows that the turbine is not rotating. Further inspection shows that the generator is not rotating because of the extreme wind conditions (and of course its alignment which is not directed against the wind direction) it experiences.

Cluster 3: Contains only itself because the time series for the gearbox temperature is very different from the rest. Wind speed, generator speed and position deviation have similar behaviours. Reviewing the generator speed it becomes clear that this turbine is not rotating. Note that this turbine is under maintenance as stated previously.

Cluster 4: Contains only itself because the time series for the gearbox temperature and the generator speed is very different from the rest. The gearbox speed is varying quite heavily between the minimum and the maximum value. This results in an equally varying gearbox temperature. It can also be observed that the oscillations for the gearbox temperature occur only when the turbine is rotating.

B.3.3: Similarity in shape - DTW is constrained and the alpha corresponding to the gearbox temperature is modified

For this analysis, the same experiment as in Section 7.1.3 is repeated with a modification to the alpha values. The alphas are now all equal to one except for the alpha corresponding to the gearbox temperature which is set to eighth. This has the effect that it weights the dissimilarities for the gearbox temperature eighth time as much as for the others. For instance, if the dissimilarity between turbine 1 and turbine 2 was 30 (with respect to the gearbox temperature), it now would be $8 * 30 = 240$. Other than the modification of the alpha value, all other parameters are identical to that of the analysis in Section 7.1.3; the analysis is included with constraints to the DTW algorithm. The summary plot and table for the internal indexes of cutting each dendrogram into k partitions - each with different linkage criterion - can be seen in Figure B.3.4 and Table B.3.2. The cophenetic correlation coefficient can be seen in the upper right corner in the aforementioned figure (Figure B.3.4). It must be noted that the internal indexes are all standardised such that the comparison between different models is possible.

Table B.3.2: Summary table with the linkage criteria: single, complete, average and ward. The data set which was clustered is 'Multivariate_V2' with the objective of similarity in time. The silhouette indexes and MSSSE values for a cut K - from 2 to 13 - are presented. The alphas are all equal to one except for the alpha corresponding to the gearbox temperature which is set to 8.

Method	Internal index	Number of cuts to the dendrogram											
		K = 2	K = 3	K = 4	K = 5	K = 6	K = 7	K = 8	K = 9	K = 10	K = 11	K = 12	K = 13
Single	Silhouette	0.33	0.29	0.37	0.37	0.33	0.32	0.21	0.14	0.13	0.14	0.14	0.11
	MSSSE	7.70	6.62	5.25	3.87	3.26	2.69	2.19	1.86	1.27	0.88	0.59	0.36
Complete	Silhouette	0.37	0.39	0.37	0.37	0.35	0.32	0.21	0.19	0.15	0.09	0.14	0.11
	MSSSE	7.23	6.15	4.77	3.87	3.30	2.69	2.19	1.60	1.31	0.98	0.59	0.36
Average	Silhouette	0.39	0.39	0.37	**0.37**	0.35	0.32	0.21	0.19	0.13	0.09	0.14	0.11
	MSSSE	7.07	6.15	5.25	**3.87**	3.30	2.69	2.19	1.60	1.27	0.98	0.59	0.36
Ward	Silhouette	0.34	0.37	0.37	0.37	0.35	0.23	0.20	0.19	0.20	0.20	0.14	0.11
	MSSSE	6.90	5.63	4.77	3.87	3.30	2.63	2.01	1.60	1.21	0.93	0.59	0.36

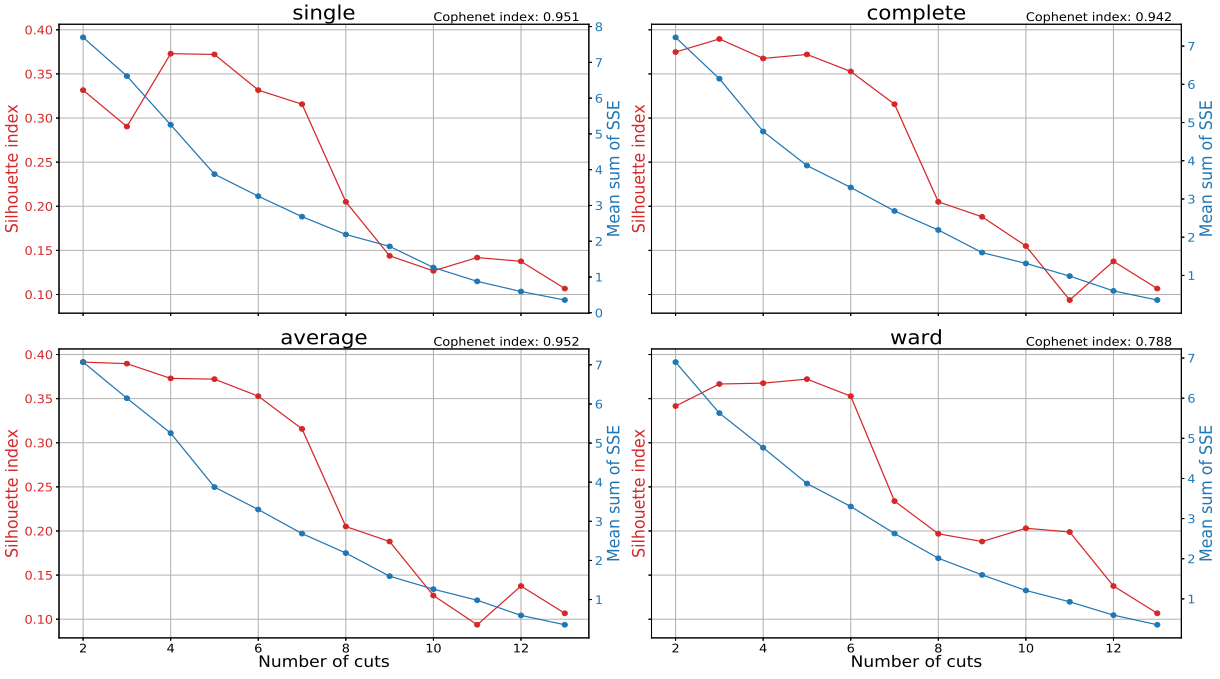


Figure B.3.4: Summary plots with the linkage criteria: single, complete, average and ward. The data set which was clustered is 'Multivariate_V1' with the objective of similarity in time. The silhouette indexes and MSSSE values for a cut K - from 2 to 13 - is presented along with the cophenetic correlation coefficient in the upper right corner. The alphas are all equal to one except for the alpha corresponding to the gearbox temperature which is set to 8.

First of all, reviewing the cophenetic correlation coefficient in Figure B.3.4, it can be seen that all dendrograms

experience similar high values; two of them have values which are above 0.95. The cophenetic correlation coefficient is close to 1 for all dendrogram; values close to 1 indicate that all the dendrograms represent a high-quality solution. The dendrogram with the highest cophenetic index is the dendrogram constructed with the linkage criterion set as *average*. The dendrogram built with the linkage criterion set as *average* will be cut and its assignment will be analysed. Secondly, reviewing the summary table and the corresponding summary plot, it can be observed that the silhouette index is largest for a cut of two and three and slightly decreases until a cut of five where it starts to decrease more rapidly. The corresponding MSSSE value from a cut of two to five decreases significantly for all these cuts. No significant reduction in the MSSSE value is observed for larger cuts; larger cuts also experience a quite significant decrease in its Silhouette index value. A cut of five to the dendrogram results in a large silhouette index and a relatively low MSSSE value. A cut of five seems to be the optimal configuration where the compromise between separation and intra-cluster similarity is at its best. The corresponding dendrogram - with *average* as linkage criterion - can be seen in Figure B.3.5. In Figure B.3.6, the cluster assignment and the time series are visualised for a cut of five.

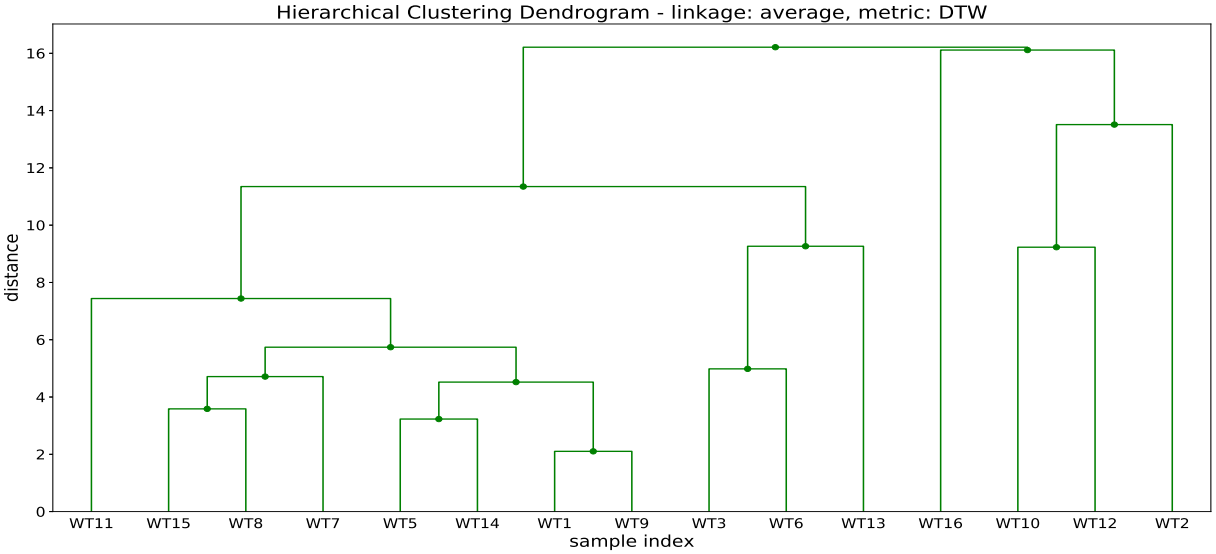


Figure B.3.5: Dendrogram with *average* as the linkage criterion and distance as Euclidean distance implemented on 'Multivariate_V2' data set. The alphas are all equal to one except for the alpha corresponding to the gearbox temperature which is set to 8.

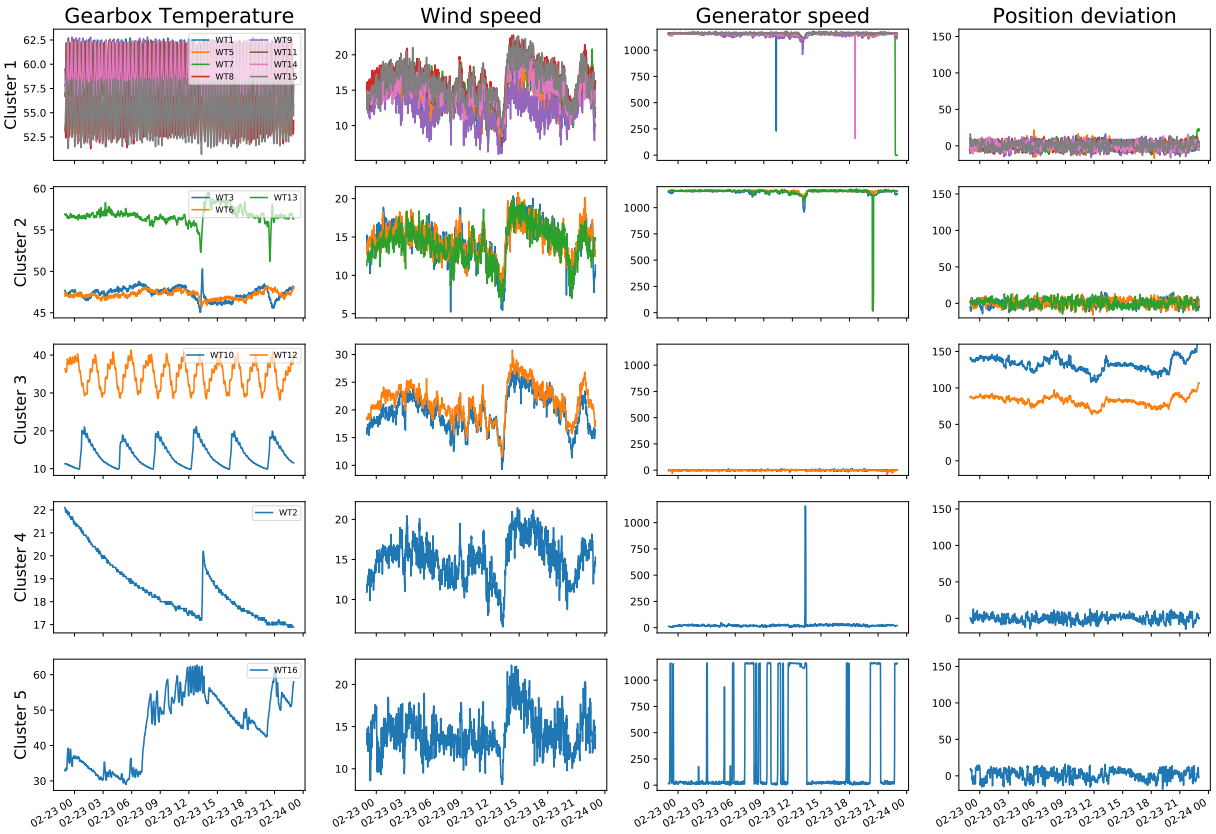


Figure B.3.6: Clustering results from the 'Multivariate_V1' data set with Euclidean distance. Summary plot of a cut of four to the dendrogram with the linkage criterion set as 'average' (from Figure B.3.5)

B.4: Additional clustering results from clustering the 'Multivariate_v2' data set

B.4.1: Similarity in shape - DTW algorithm not constrained and a dimension of 5

The objective is now to cluster the time series with the objective of similarity in shape and not the absolute difference and offset. Therefore, the clustering analysis is performed on the normalised data set. Because of the invalid measurements of the external temperature, the corresponding univariate data set will be excluded from cluster analysis in this section. The other time series associated with 'Multivariate_V2', post-filtering, can be observed in Appendix A.3. None of the time series for any of the univariate data sets experiences flat and noisy behaviour (as in Section 7.1.3 for the generator speed). The time series can, therefore, be normalised prior to clustering with the objective of similarity in shape. The similarity measure used between time series is the dynamic time warping (DTW) measurement presented in Section 3.1. The DTW algorithm is unconstrained. DTW has much higher complexity then Euclidean distance and the calculation of the condensed distance matrix takes now over 16 minutes (over 8 minutes when using parallelization with two cores). The condensed distance matrix is then fed to the hierarchical clustering algorithm. The optimal hierarchical clustering method and the optimal number of clusters will be found with the same internal indexes used so far: Cophenetic correlation coefficient, Silhouette index and MSSSE. More information about these specific indexes can be found in Chapter 3. The summary plot and table for the internal indexes of cutting each dendrogram - each with different linkage criterion - into *k* cuts can be seen in Figure B.4.1 and Table B.4.1. The cophenetic correlation coefficient can be seen in the upper right corner in the aforementioned figure (Figure B.4.1). Because we do not wish to include the external temperature (obviously invalid/wrong measurement) the corresponding alpha is set to zero, disregarding the measurements completely. The alpha corresponding to the wind direction is kept as 1, as it still might be similar in shape.

Table B.4.1: Summary table with the linkage criteria: single, complete, average and ward. The data set which was clustered is 'Multivariate_V2' with the objective of similarity in shape. The silhouette indexes and MSSSE values for a cut K - from 2 to 13 - are presented. The alphas are all equal to one except for the alpha corresponding to the external (outside) temperature which is set to zero.

Method	**Internal index**	**Number of cuts to the dendrogram**											
		K = 2	K = 3	K = 4	K = 5	K = 6	K = 7	K = 8	K = 9	K = 10	K = 11	K = 12	K = 13
Single	Silhouette	0.52	0.36	0.27	0.21	0.16	0.08	0.05	0.04	0.01	0.01	0.05	0.07
	MSSSE	5.38	4.36	3.76	3.25	2.87	2.50	2.11	1.78	1.45	1.04	0.67	0.39
Complete	Silhouette	0.52	0.28	0.26	0.15	0.13	0.15	0.15	0.15	0.13	0.08	0.08	0.07
	MSSSE	5.38	4.47	3.75	3.34	2.85	2.44	2.05	1.72	1.39	1.01	0.69	0.39
Average	Silhouette	**0.52**	0.36	0.26	0.21	0.13	0.10	0.15	0.10	0.08	0.08	0.08	0.07
	MSSSE	**5.38**	4.36	3.75	3.25	2.88	2.49	2.05	1.67	1.34	1.01	0.69	0.39
Ward	Silhouette	0.52	0.36	0.23	0.15	0.15	0.16	0.16	0.15	0.10	0.08	0.08	0.07
	MSSSE	5.38	4.36	3.81	3.38	2.88	2.51	2.12	1.72	1.34	1.01	0.69	0.39

First of all, reviewing the cophenetic correlation coefficient in Figure B.4.1, it can be seen that all dendrograms experience similar high values. Both the dendrogram constructed with single and average linkage criteria have high values above 0.95. A cophenetic correlation coefficient is very close to 1 and well above 0.75 which indicates that the dendrograms represent a high-quality solution. The dendrogram with the largest cophenetic index is the dendrogram constructed with the linkage criterion set as *average*. Therefore, the dendrogram built with the linkage criterion set as *average* will be cut and its assignment will be analysed. Secondly, reviewing the summary table and the corresponding summary plot, it can be observed that the silhouette index is largest for a cut of two and decreases exponentially after; the corresponding MSSSE value has a relatively steep descend from a cut of two to a cut of three (with over 20% reduction in the within-cluster variance), but afterwards decreases more slowly. As the silhouette index decreases exponentially after a cut of two and a reduction in the MSSSE is not significant, a cut of two is arguably best cut for the dendrogram. A cut of two seems to be the optimal configuration where the compromise between separation and intra-cluster similarity is at its best. Therefore, a cut of two to the dendrogram will be further analysed in the consecutive section. The corresponding dendrogram - with *average* as the linkage criterion and the external temperature excluded - can be seen in Figure B.4.2. As the Silhouette index and the MSSSE values are identical for $K = 2$ for all dendrograms, one could assume that the different dendrograms produce the same cluster assignments.

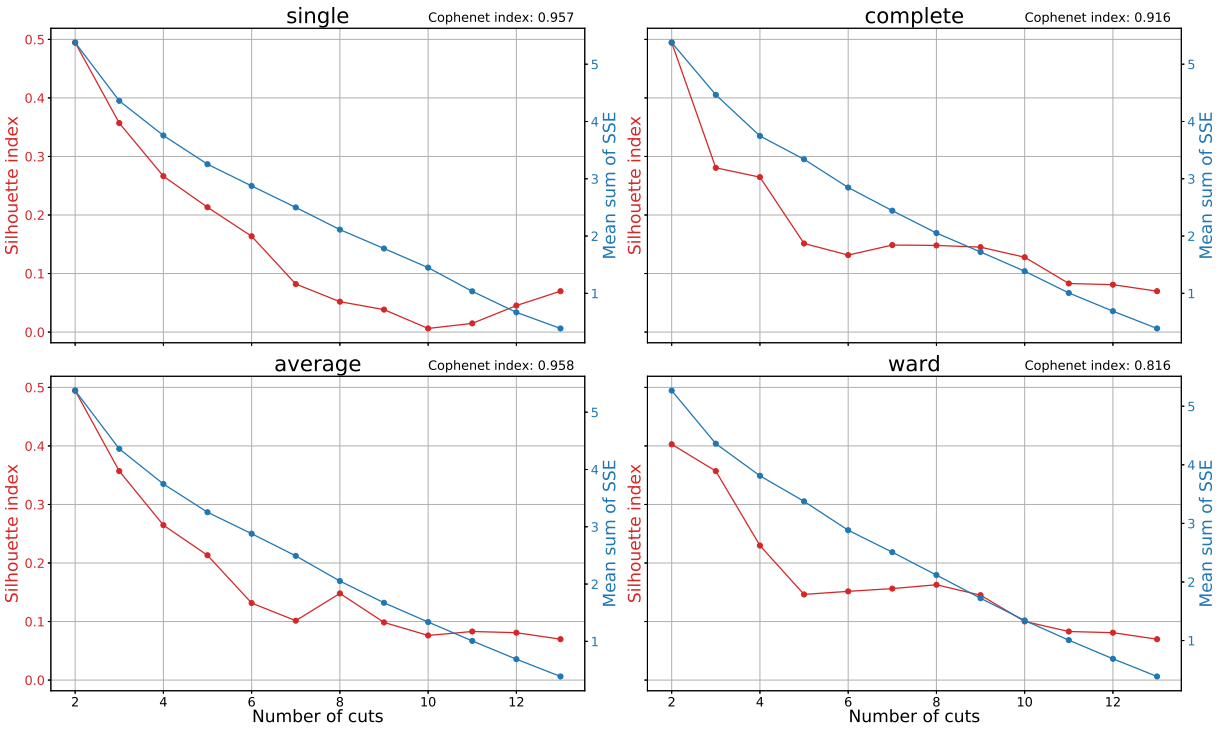


Figure B.4.1: Summary plots with the linkage criteria: single, complete, average and ward. The data set which was clustered is 'Multivariate_V2' with the objective of similarity in shape. The silhouette indexes and MSSSE values for a cut K - from 2 to 13 - is presented along with the cophenetic correlation coefficient in the upper right corner. The alphas are all equal to one except for the alpha corresponding to the external (outside) temperature which is set to zero.

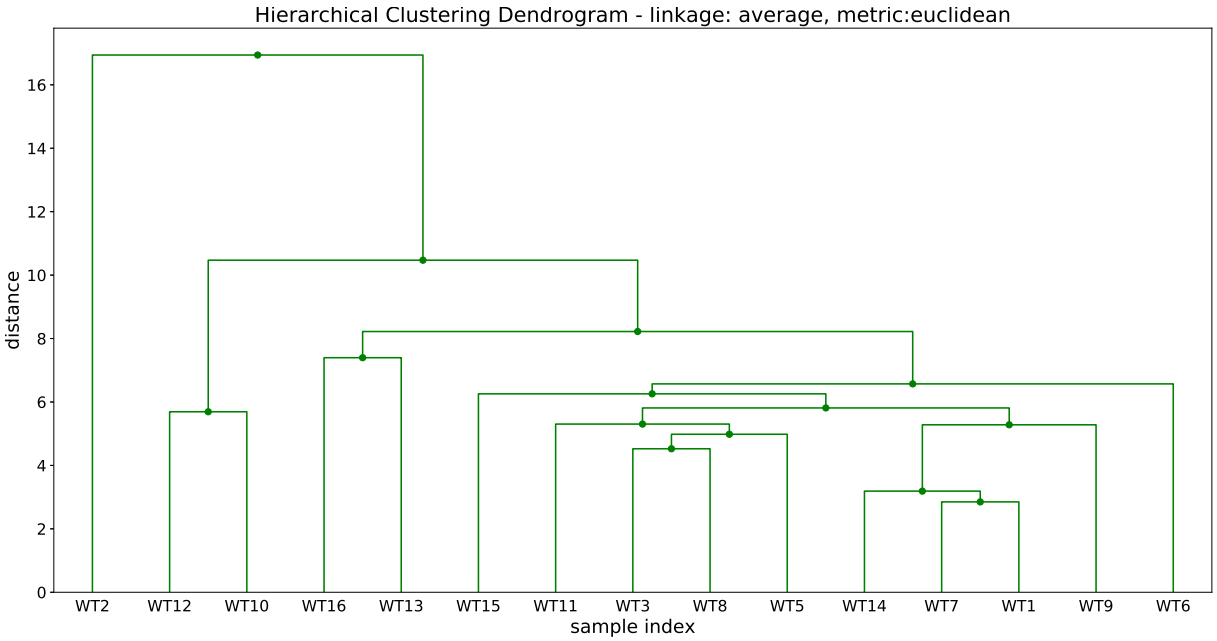


Figure B.4.2: Dendrogram with *average* as linkage criterion and distance DTW distance implemented on 'Multivariate_V2' data set. The alphas are all equal to one except for the alpha corresponding to the external (outside) temperature which is set to zero.

B.4.2: Analysing the clustering results from similarity in shape - DTW algorithm not constrained and the dimension of 5

In this section, the clustering results from clustering 'Multivariate_V2' data set in Appendix B.4.1 - where the external temperature is excluded - will be analysed. The corresponding dendrogram which will be cut can be seen in Figure B.4.2. In Figure B.4.3 the cluster assignment and the time series are visualised for a cut of two.

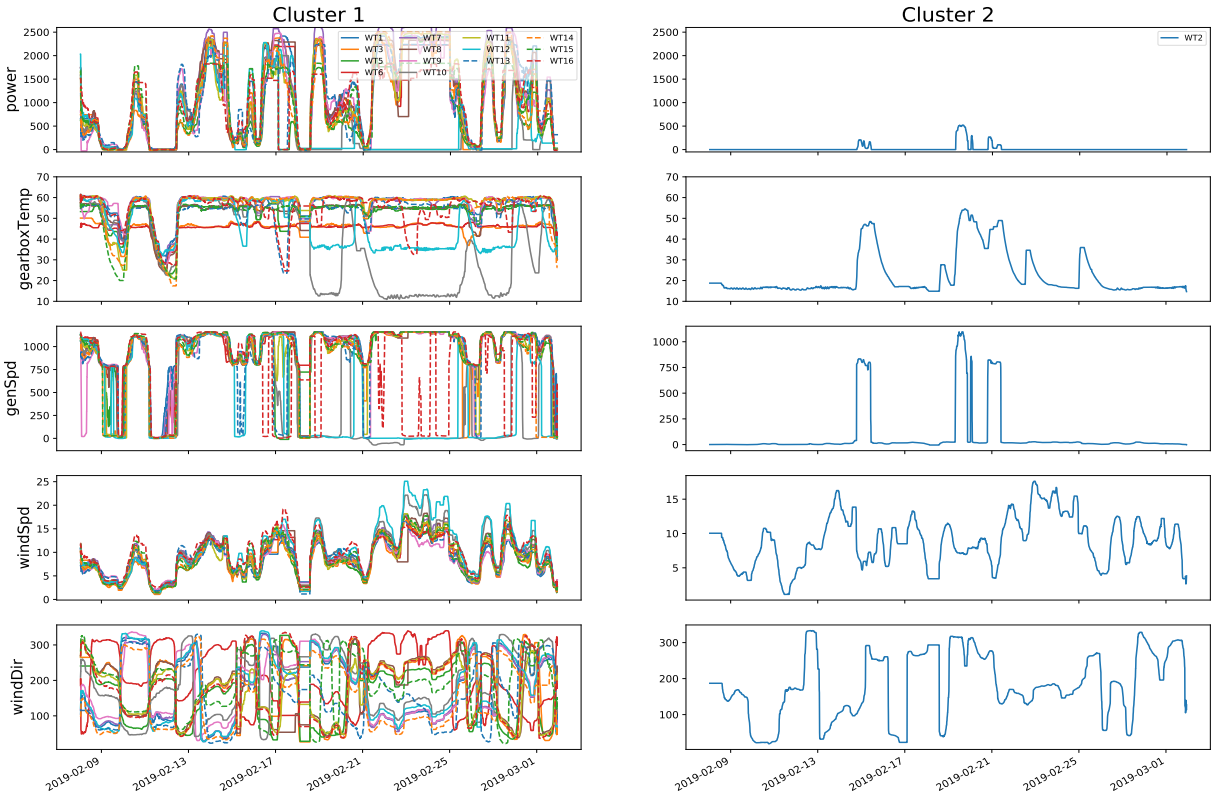


Figure B.4.3: Clustering results from the 'Multivariate_V2' data set with DTW distance. Summary plot of a cut of two to the dendrogram with the linkage criteron set as 'average' (from Figure B.4.2)

A cut of two to the dendrogram separates only turbine 2 from the remaining turbines. This indicates that turbine 2 is vastly different from the rest. This can easily be observed by reviewing the corresponding time series for the power, gearbox temperature and generator speed. More interestingly, a cut of three to the current dendrogram (Figure B.4.2 would result in the exact same cluster assignment as for similarity in time. Because the cluster assignment is the same for both objectives, the clusters formed are both similar in time and in shape. Additional information from clustering with the objective of similarity in shape is not transparent. Nonetheless, the physical interpretation of the cluster assignment will be interpreted in the following paragraphs.

Cluster 1: This cluster contains all of the turbines except for turbine 2. The turbines associated with cluster 1 are characterised by medium to high power production caused by a medium to high generator speed. From the objective of similarity in time, there are two time series which are, to a degree, outliers in this plot: Turbine 10 and 12 which experience low to no generator speed during the second half of the interval. This is caused by elevated wind speed conditions. This is further backed up by reviewing the gearbox temperature which also experiences similar drops as the power and generator speed. That is because the power production, gearbox temperature and generator speed are highly correlated to each other. Regarding the wind speed, no benefit is observed by including this in the analysis.

Cluster 2: Contains only turbine 2 and is one of the turbines which is vastly different from the rest. Turbine 2 have very small power production; where the power production is zero so is the generator speed followed by a low gearbox temperature. The wind conditions do not indicate high wind speed, or at least elevated from the others. With the acquired information, no justification can be made for the low generator speeds observed. However, after reviewing the maintenance plan, turbine two are in fact under maintenance during this interval.

B.4.3: Similarity in shape - DTW algorithm constrained and the dimension of 4

The objective is now to cluster the time series with the objective of similarity in shape and not the absolute difference and offset. Therefore, the clustering analysis is performed on the normalised data set. The dimension of the data set is now four where the alpha corresponding to the external temperature and the wind direction is set to zero. The DTW algorithm is run with the same settings as in Section 7.2.3, *Sakeo-Chiba* band width of 10% of the length of the time series. DTW has much higher complexity then Euclidean distance and the calculation of the condensed distance matrix took over one hour to complete. The condensed distance matrix is then fed to the hierarchical clustering algorithm. The optimal hierarchical clustering method and the optimal number of clusters will be found with the same internal indexes used so far: Cophenetic correlation coefficient, Silhouette index and MSSSE. More information about these specific indexes can be found in Chapter 3. The summary plot and table for the internal indexes of cutting each dendrogram - each with different linkage criterion - into *k* cuts can be seen in Figure B.4.4 and Table B.4.2. The cophenetic correlation coefficient can be seen in the upper right corner in the aforementioned figure (Figure B.4.4).

Table B.4.2: Summary table with the linkage criteria: single, complete, average and ward. The data set which was clustered is 'Multivariate_V2' with the objective of similarity in shape. The silhouette indexes and MSSSE values for a cut K - from 2 to 13 - are presented. The alphas are all equal to one except for the alpha corresponding to the external (outside) temperature which is set to zero.

Method	**Internal index**	**Number of cuts to the dendrogram**											
		K = 2	K = 3	K = 4	K = 5	K = 6	K = 7	K = 8	K = 9	K = 10	K = 11	K = 12	K = 13
Single	Silhouette	0.57	**0.65**	0.44	0.37	0.26	0.12	0.08	0.05	0.05	0.04	0.05	0.01
	MSSSE	7.02	**4.26**	3.47	2.94	2.42	2.05	1.74	1.42	1.14	0.87	0.58	0.37
Complete	Silhouette	0.67	0.65	0.43	0.33	0.26	0.12	0.06	0.06	0.06	0.06	0.06	0.03
	MSSSE	5.69	4.26	3.52	2.96	2.42	2.09	1.84	1.54	1.20	0.93	0.63	0.38
Average	Silhouette	0.67	0.65	0.44	0.33	0.26	0.12	0.08	0.09	0.07	0.05	0.05	0.03
	MSSSE	5.69	4.26	3.47	2.96	2.42	2.05	1.74	1.44	1.15	0.85	0.58	0.38
Ward	Silhouette	0.67	0.65	0.43	0.33	0.26	0.12	0.11	0.06	0.06	0.06	0.06	0.03
	MSSSE	5.69	4.26	3.52	2.96	2.42	2.09	1.78	1.54	1.20	0.93	0.63	0.38

First of all, reviewing the cophenetic correlation coefficient in Figure B.4.4, it can be seen that all dendrograms experience similar high values. Both the dendrogram constructed with single and average have high values above 0.95. A cophenetic correlation coefficient is very close to 1 and well above 0.75 which indicates that the dendrograms represent a high-quality solution. The dendrogram with the largest cophenetic index is the dendrogram constructed with the linkage criterion set as *single*. Therefore, the dendrogram built with the linkage criterion set as *single* will be cut and its assignment will be analysed. Secondly, reviewing the summary table and the corresponding summary plot, it can be observed that the silhouette index is largest for a cut of three and decreases exponentially after; the corresponding MSSSE value has a relatively steep descend from a cut of two to a cut of three but afterwards, decreases more slowly. As the silhouette index decreases exponentially after a cut of three and a reduction in the MSSSE is not significant, a cut of three is arguably best cut for the dendrogram. A cut of three seems to be the optimal configuration where the compromise between separation and intra-cluster similarity is at its best. Therefore, a cut of three to the dendrogram will be further analysed in the consecutive section. The corresponding dendrogram - with *single* as linkage criterion and the external temperature and wind direction is excluded - can be seen in Figure B.4.5.

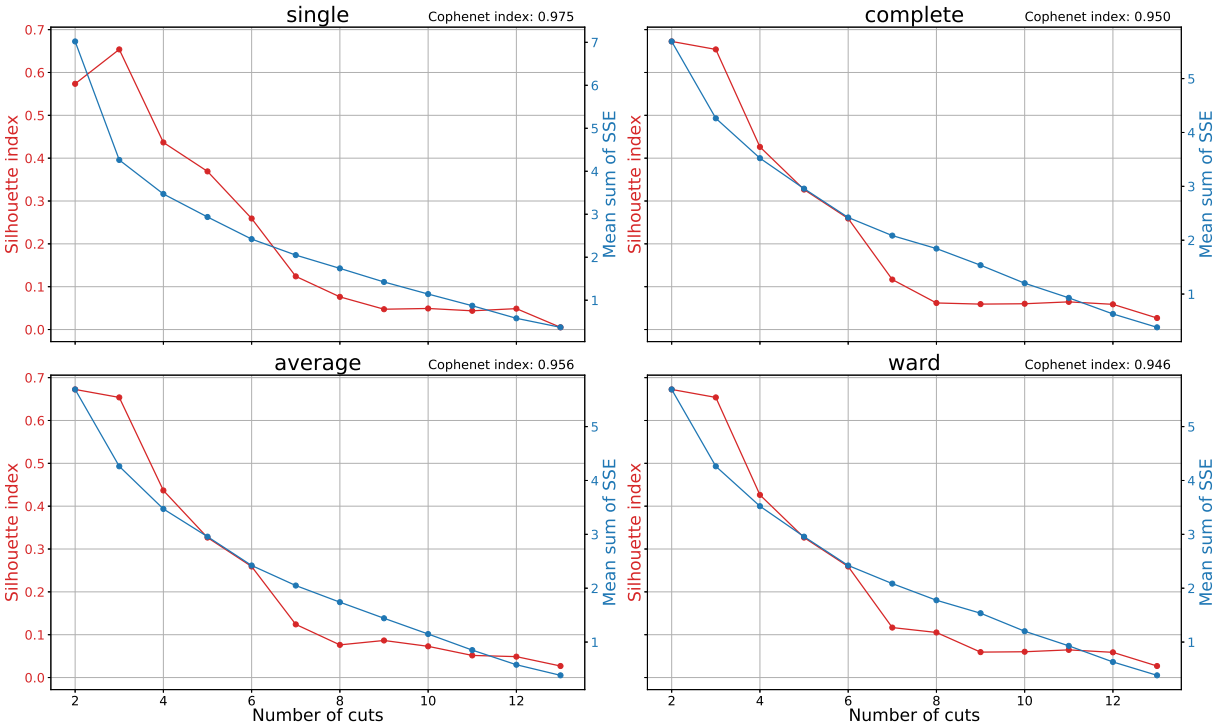


Figure B.4.4: Summary plots with the linkage criteria: single, complete, average and ward. The data set which was clustered is 'Multivariate_V2' with the objective of similarity in shape. The silhouette indexes and MSSSE values for a cut K - from 2 to 13 - is presented along with the cophenetic correlation coefficient in the upper right corner. The alphas are all equal to one except for the alpha corresponding to the external (outside) temperature and wind direction which is set to zero.

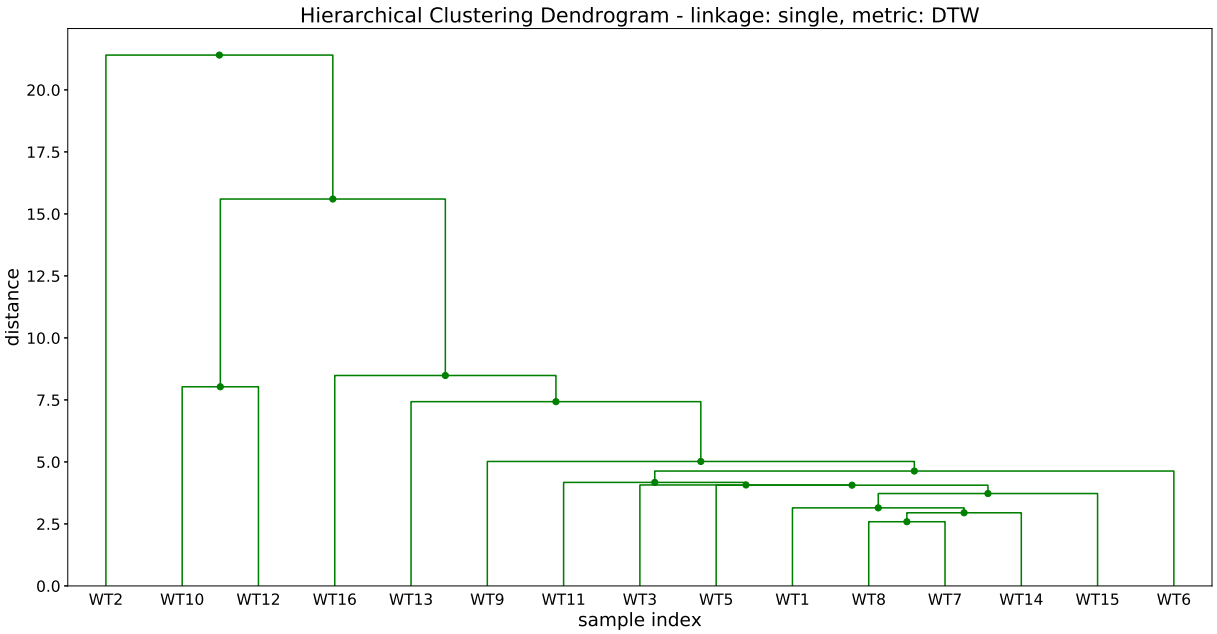


Figure B.4.5: Dendrogram with *average* as the linkage criterion and distance DTW distance implemented on 'Multivariate_V2' data set. The alphas are all equal to one except for the alpha corresponding to the external (outside) temperature and wind direction which is set to zero.

B.4.4: Analysing the clustering results from similarity in shape - DTW algorithm constrained and the dimension of 4

In this section, the clustering results from clustering 'Multivariate_V2' data set in Appendix B.4.3 - where the external temperature and the wind direction are excluded - will be analysed. The corresponding dendrogram which will be cut can be seen in Figure B.4.5. In Figure B.4.6 the cluster assignment and the time series are visualised for a cut of two.

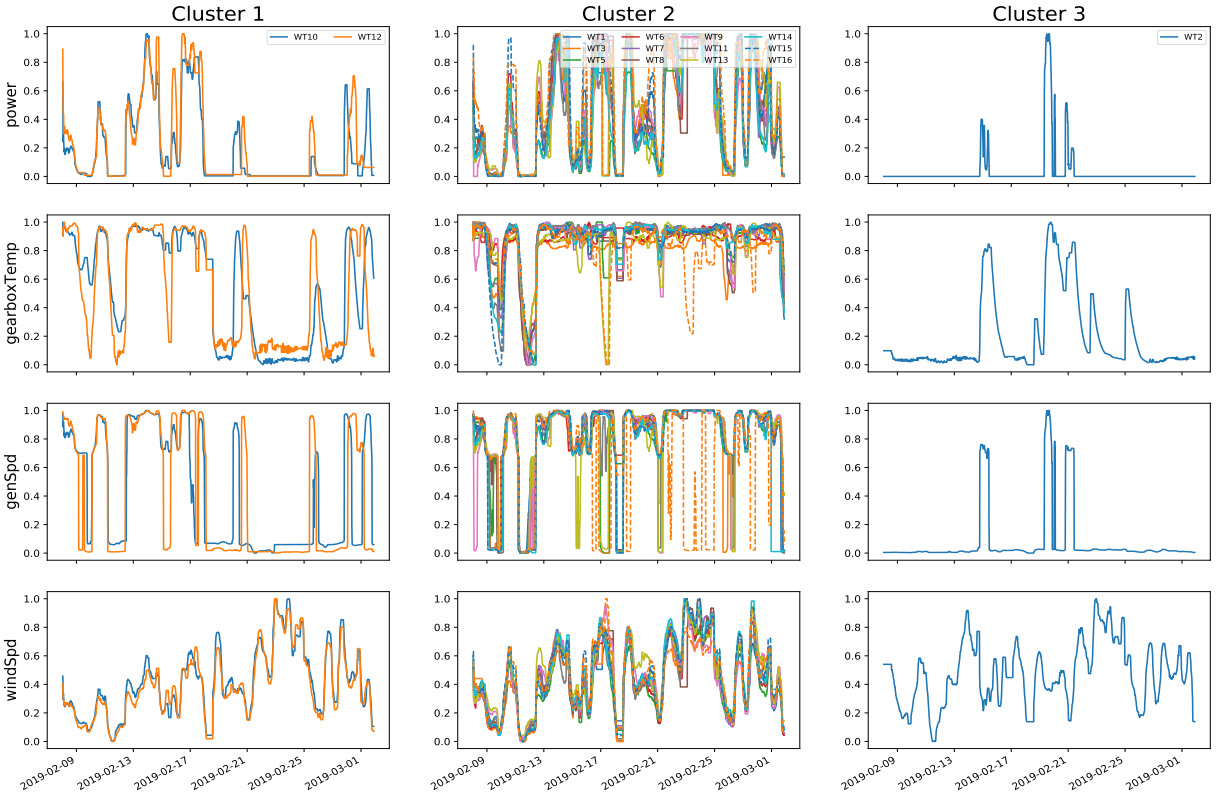


Figure B.4.6: Clustering results from the 'Multivariate_V2' data set with DTW distance. Summary plot of a cut of three to the dendrogram with the linkage criterion set as 'average' (from Figure B.4.5). Note the normalised time series is shown, rather than the time series after filtering.

Similar to clustering with respect to the objective of similarity in time in Section 7.2.1, the clustering results from the objective of similarity in shape for dimension equal to 4 and 5 are identical. The same cluster assignments can be observed by comparing Figure 7.18 and B.4.6. The corresponding dendrograms can also be observed to be close to identical for both analyses. The only difference can be observed in the internal indexes where a large silhouette index and a lower MSSSE value (i.e. within-cluster variance) can be observed for the current analysis (dimension of 4).

B.4.5: Comparison between similarity in time and similarity in shape for the longer time series

In this section, the aim is to compare the clustering results acquired from the objective of similarity in time with the objective of similarity in shape. The clustering results and analysis of similarity in shape has already been done in Section 7.2.3. In this section, the normalised data set - not the scaled data set as with the objective of similarity in time analysis - will be clustered with the Euclidean distance metric. Similar to previous analyses, the best linkage criterion for hierarchical clustering will be found by comparing the *Cophenetic correlation coefficient* (3.10) and the number of cuts to the dendrogram will be determined by viewing the internal performance indexes: *Silhouette index* (3.14) and *MSSSE* (3.17) for multivariate analysis. More information about these specific indexes can be found in Chapter 3. The summary plot and table for the internal indexes of cutting each dendrogram into *k* partitions - each with different linkage criterion - can be seen in Figure B.4.7 and Table B.4.3. The cophenetic correlation coefficient can be seen in the upper right corner in the aforementioned figure (Figure B.4.7). In the current run, the alphas are all equal to one, except for the alphas associated with the external temperature and the wind direction, which is set to zero.

Table B.4.3: Summary table with the linkage criteria: single, complete, average and ward. The data set which was clustered is 'Multivariate_V2' with the objective of similarity in time. The silhouette indexes and MSSSE values for a cut K - from 2 to 13 - is presented. The alphas are all equal to one, except for the alphas associated with the external temperature and the wind direction, which is set to zero.

Method	Internal index	Number of cuts to the dendrogram											
		K = 2	K = 3	K = 4	K = 5	K = 6	K = 7	K = 8	K = 9	K = 10	K = 11	K = 12	K = 13
Single	Silhouette	0.52	**0.58**	0.39	0.32	0.17	0.10	0.07	0.03	0.04	0.04	0.05	0.02
	MSSSE	220.56	**82.55**	57.57	45.61	36.03	28.71	23.24	19.26	13.71	9.80	6.09	3.72
Complete	Silhouette	0.59	0.58	0.39	0.32	0.15	0.10	0.10	0.08	0.07	0.05	0.05	0.02
	MSSSE	138.77	82.55	57.57	45.61	34.43	26.74	21.02	16.24	12.54	9.12	6.09	3.72
Average	Silhouette	0.59	0.58	0.39	0.32	0.17	0.10	0.10	0.08	0.07	0.05	0.05	0.02
	MSSSE	138.77	82.55	57.57	45.61	36.03	28.71	21.02	16.24	12.54	9.12	6.09	3.72
Ward	Silhouette	0.59	0.58	0.39	0.22	0.15	0.10	0.10	0.08	0.07	0.05	0.05	0.02
	MSSSE	138.77	82.55	57.57	46.40	34.43	26.74	21.02	16.24	12.54	9.12	6.09	3.72

First of all, reviewing the cophenetic correlation coefficient in Figure B.4.7, it can be seen that all of the dendrograms experience similar high values; all are above 0.95. The cophenetic correlation coefficient is close to 1 for all dendrograms; values close to 1 indicate that the dendrograms represent a high-quality solution. The dendrogram with the highest cophenetic index is the dendrogram constructed with the linkage criterion set as *single*. Therefore, the dendrogram built with the linkage criterion set as *single* will be cut and its assignment will be analysed. Secondly, reviewing the summary table and the corresponding summary plot, it can be observed that the silhouette index is largest for a cut of three and decreases exponentially for deeper cuts; the corresponding MSSSE value has a relatively steep descend from a cut of two to a cut of three, but for deeper cuts it decreases more slowly. A *cut of three* seems to be the optimal configuration where the compromise between separation and intra-cluster similarity is at its best. Therefore, a cut of three to the dendrogram will be further analysed in the consecutive section. The corresponding dendrogram - with *single* as linkage criterion and the external temperature excluded - can be seen in Figure B.4.8.

Cutting the dendrogram into three partitions results in the exact same cluster assignment as for scenario 2 for both similarity in time done in Section 7.2.1 and similarity in shape analysis presented in Appendix B.4.3. Now we will compare the similarity in time analysis (done in this section) to the similarity in shape analysis presented in Appendix B.4.3. The only difference between the analysis is the similarity measure: Euclidean distance is used in this section and DTW distance is used in Appendix B.4.3. Initially, the internal indexes can be observed to recommend the exact same dendrogram as well as the same number of cuts to the dendrograms. No significant differences can be observed by reviewing the silhouette index. But reviewing the MSSSE value (i.e. the within-cluster variance), the internal indexes are quite different: 4.26 for the similarity in shape analysis and 73.91 for the similarity in time analysis. From reviewing the time series assigned to each cluster in Figure B.4.6 some differences can be observed within each cluster. By looking at the first cluster, we can observe that the time series for the first half of the interval are quite similar in time and shape. Recall, that similarity in shape is a more general case of similarity, where the time of occurrence of patterns is not important (Aghabozorgi et al., 2015; Zhang et al., 2011). But for the second half of the interval, some delays in the signal can be observed. These delays will results in a very large pairwise distance between the time series if the Euclidean distance is used, which in the end, will

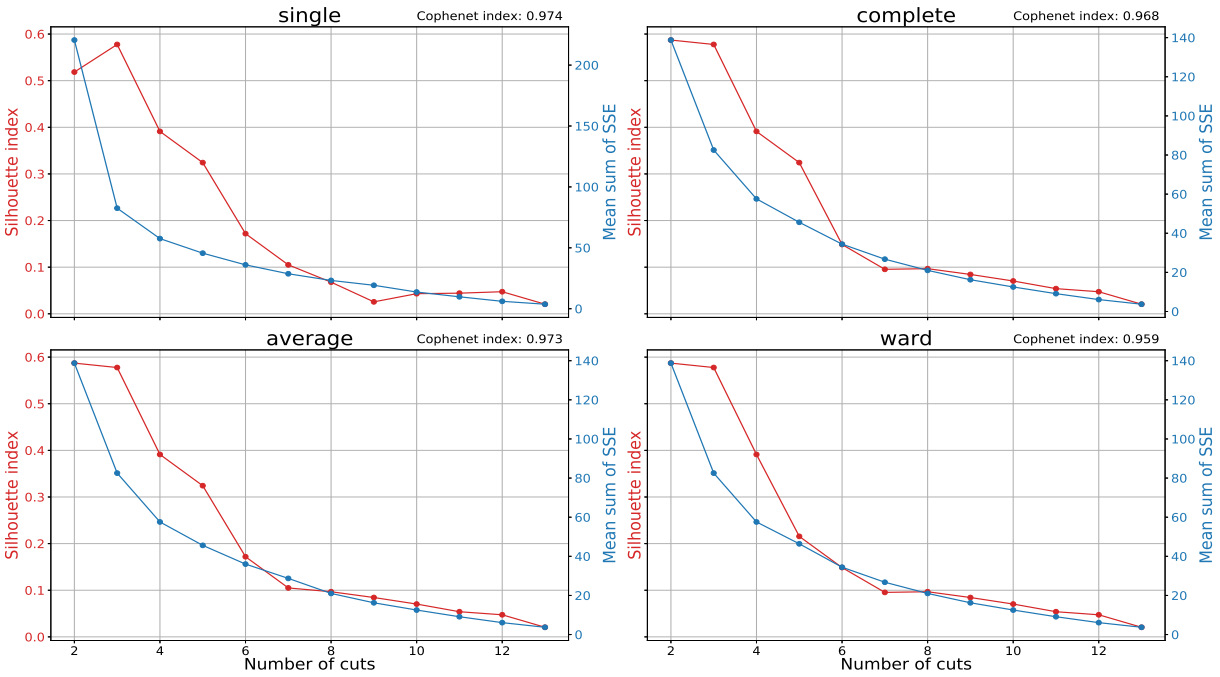


Figure B.4.7: Summary plots with the linkage criteria: single, complete, average and ward. The data set which was clustered is 'Multivariate_V2' with the objective of similarity in time. The silhouette indexes and MSSSE values for a cut K - from 2 to 13 - is presented along with the cophenetic correlation coefficient in the upper right corner. The alphas are all equal to one, except for the alphas associated with the external temperature and the wind direction, which is set to zero.

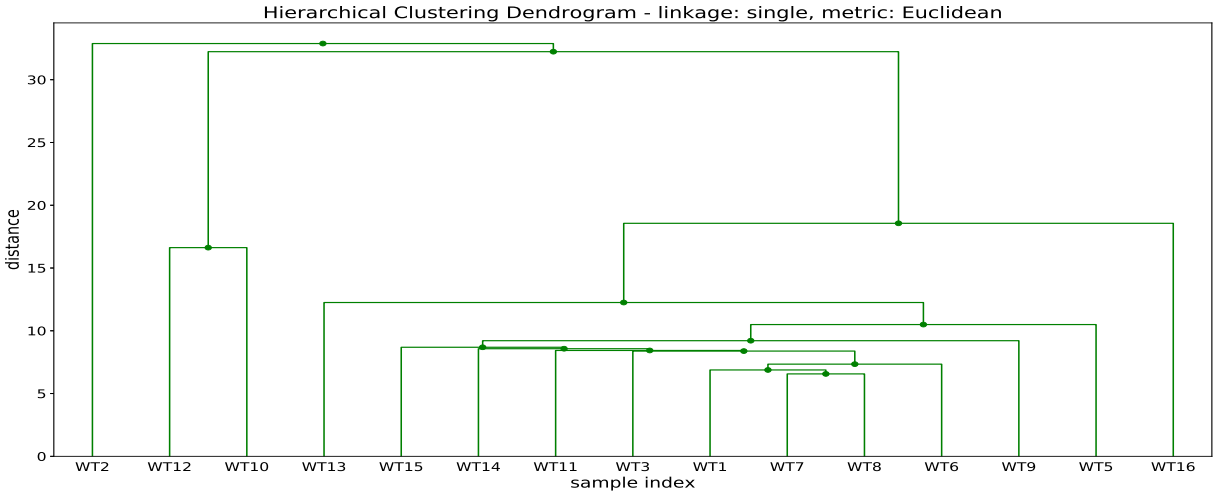


Figure B.4.8: Dendrogram with *average* as the linkage criterion and distance as Euclidean distance implemented on 'Multivariate_V2' data set. The alphas are all equal to one, except for the alphas associated with the external temperature and the wind direction, which is set to zero.

result in a significant increase in the within-cluster variance (i.e. MSSSE value).

Now, let us compare the dendrograms for both analyses with each other. The dendrograms, with DTW and Euclidean distance metric, can be seen in Figure B.4.5 and B.4.8, respectively. Initially, these dendrograms might look quite dissimilar, but they result in the exact same cluster assignment for all cuts up to a cut of six. The results show that the Euclidean distance is quite competitive to DTW, in terms of the cluster "accuracy", as shifts in time are less dominating for the longer time series. The results from clustering the normalised data set could be achieved by implementing either the Euclidean or the DTW distance; DTW comes with an increased accuracy and a high running time. Calculation the dissimilarity matrix with Euclidean distance took only 8.4ms compared the close to two hours for calculation the dissimilarity matrix with the DTW distance. Then, if the cluster accuracy is more important than the running speed, the DTW similarity measure should be used; and if the running time is more

important than the accuracy, the Euclidean distance should be used. But is the longer time series less affected by these time shifts? It is at least obvious that the oscillations observed in the 'Multivariate_V1' data sets could cause some serious damage if the Euclidean distance is used; very small shifts in time could be the difference between two time series being very similar to each other or vastly different. Consider the Euclidean distance and the DTW distance between a sinus wave. Now consider the following example where two cosine waves which have very high-frequency oscillations. If one of these is subjected to a relatively small time delay - which results in a phase lag of close to 90 degrees - the corresponding Euclidean distance between these to signals would become vastly different. This time delay does not need to be large, as long as it causes a significant shift in the phase. Longer time series does not contain such high-frequency components - at least not with the current extraction settings and preprocessing - and is, therefore, less affected by small time shifts (or delays).

Appendix C - Additional calculation and remarks

C.1: Making a comparison between different widths of the Sakeo-Chiba band constraint of the DTW algorithm.

In some cases, DTW can provide undesired effects, such as the construction of a path which is far from the diagonal. This might map a large number of points to a single point. In Figure C.1.1, the DTW is calculated without constraints and finds the optimal and expected path between the two time series. In many cases, this might not be what we wanted and can be overcome by restricting the warping path close to the diagonal (Cassisi et al., 2012).

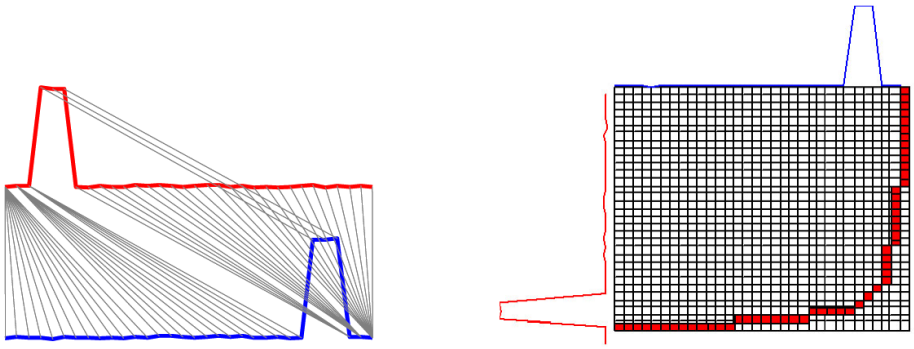
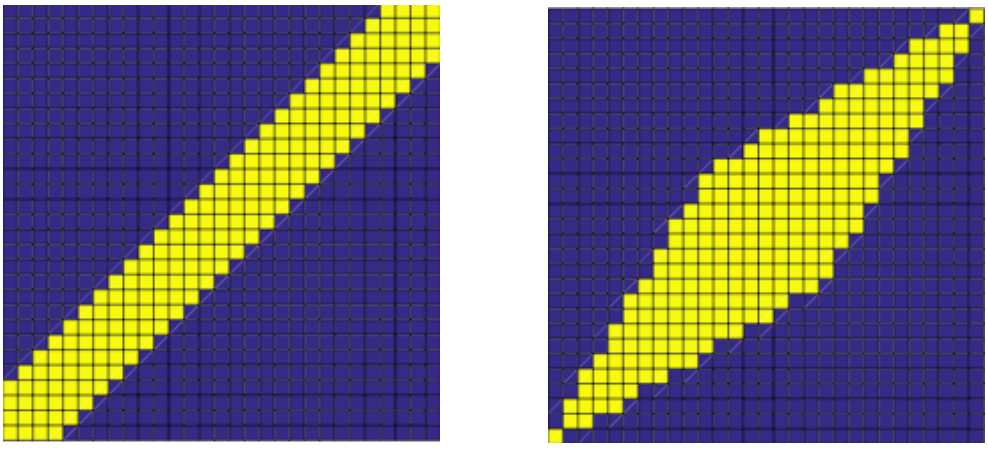


Figure C.1.1: DTW without global constraint. Taken from (Niennattrakul and Ratanamahatana, 2006)

Two popular global restrictions which are normally implemented are called the *Sakeo-Chiba band* and *Itakura parallelogram* (Cassisi et al., 2012; Elsworth, 2017). Both of these methods restricts the path along the yellow squares as illustrated in Figure C.1.2.



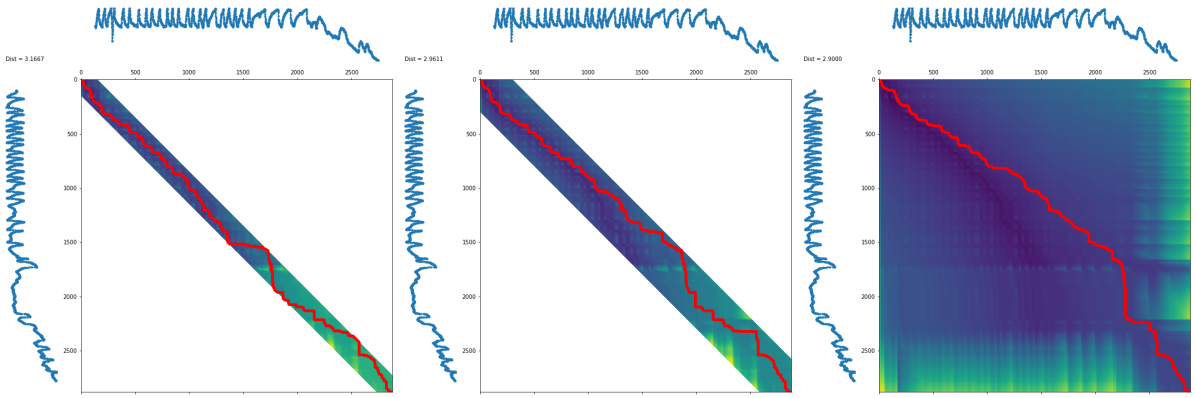
(a) Sakeo-Chiba band

(b) Itakura parallelogram

Figure C.1.2: Yellow cubes indicates the restricted warping path. Figures are taken from Elsworth (2017)

In this section, the effect of different Sakeo-Chiba band widths will be discussed. Ratanamahatana and Keogh (2004); Niennattrakul and Ratanamahatana (2006) states that the vast majority of the data mining community used a width of 10%, which is believed to be superior to other widths. In Ratanamahatana and Keogh (2004) Ratanamahatana and Keogh (2004), an empirical experiment was performed on seven data sets with varying widths between 0-100% of the length of the data set. They found that adjusting the warping window size did indeed affect the accuracy of the clustering, but for each data set, the optimal width was located between 1-10% of the length of the time series. The paper also showed that with a decreasing size of the database (the number of turbines in our case), the classification accuracy decreases with it and the corresponding optimal width was larger. For databases containing 12-24 objects, the width was closer to 10%. In addition to Ratanamahatana and Keogh (2004), Xi et al. (2006) also states that narrower constraints are necessary for accurate DTW and that a too large warping window

could actually deteriorate the accuracy. Finding the optimal value is specific to the application and hard to find. Because the ground truth in the thesis is not known, the clustering accuracy cannot be measured and finding the optimal width of the Sakeo-Chiba band cannot be done by comparing the classification accuracy. The effects of different widths will be done through a visual approach by comparing the different warped paths achieved from finding the DTW distance between a couple of time series. A comparison between a width of $5\%, 10\%$ and 100% (or no constraint) will be performed on several combinations of time series. The warped paths are presented in Figure C.1.5, C.1.6, C.1.3 and C.1.4. The darker (blue) colour indicates a low weighing value and lighter colours indicates a high value, and vice versa.

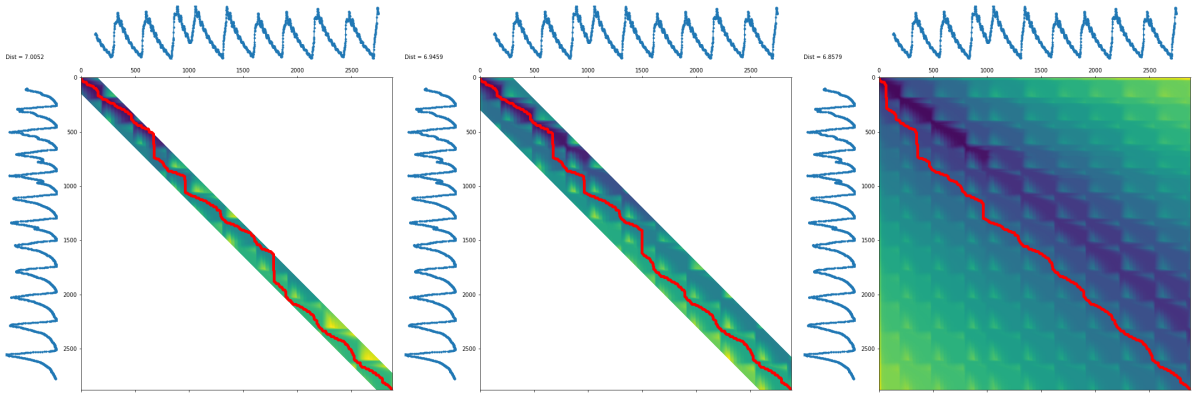


(a) Sakeo-Chiba band width of 5% and DTW distance of 3.1667.

(b) Sakeo-Chiba band width of 10% and DTW distance of 2.9611.

(c) No constraints and DTW distance of 2.9.

Figure C.1.3: Plotting of the warped path between time series of the gearbox temperature for turbine 1 and turbine 6 in the 'Univariate_V1' data set.



(a) Sakeo-Chiba band width of 5% and DTW distance of 7.00.

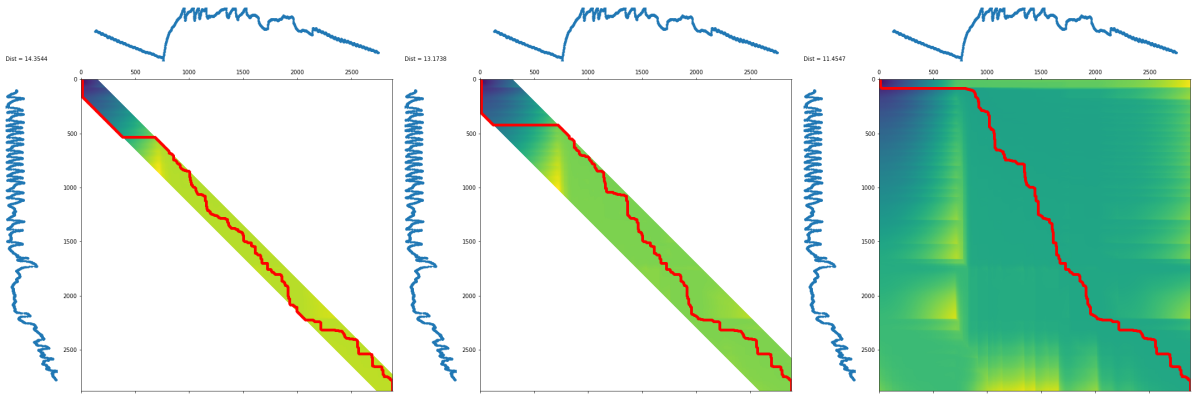
(b) Sakeo-Chiba band width of 10% and DTW distance of 6.94.

(c) No constraints and DTW distance of 6.86.

Figure C.1.4: Plotting of the warped path between time series of the gearbox temperature for turbine 2 and turbine 16 in the 'Univariate_V1' data set.

First of all, lets review Figure C.1.3 and C.1.4. For both examples, the two time series compared are relatively similar in terms of shape. Because these time series are very similar to each other, the corresponding warped path from the DTW algorithm, between different widths, are also very similar to each other. This can be further verified by reviewing the calculated DTW distance for the warped path. Note that the warped path starts at the upper left corner and ends at the bottom right. From comparing the weights for these two plots, a deterioration of the accuracy cannot be observed and the choice between the different widths seems to be insignificant. Now let us look at the warped path between time series which are both slightly and very different to each other. In Figure C.1.5 and C.1.6, the warped path between time series which are obviously different to each other are presented. The two time series analysed in Figure C.1.5 can be observed to be different during the first part of the interval and have very close similarity for the remainder of the interval. The effect of no constraints to the DTW algorithm results

in a warping path which maps a large number of points to a single point during the first part of the interval; the warped path, in this case, is therefore quite different from other widths. The corresponding DTW distance can be observed to decrease with the increase of the band width, which is expected. The large DTW distance still reflects a relatively large dissimilarity between the two time series. Similar results can be observed in the other figure (Figure C.1.6), where the warped path has been found between two very different time series. A quite narrow width results in the quite large DTW distance. The DTW distance can be observed to decrease as the width of the band increases. The reduction of the DTW distance can be observed to be slightly more significant than for those two time series in Figure C.1.5. Furthermore, for the warped path which was found without any constraints (Figure C.1.6), a very large number of points can be observed to be mapped to a single point. This case is an unwanted case as the (dis)similarity in shape between the two time series is lost; it distorts the distance. It can be observed that the DTW distance between turbine 2 and 12 is quite similar to the DTW distance between turbine 1 and 12, which does make sense visually.

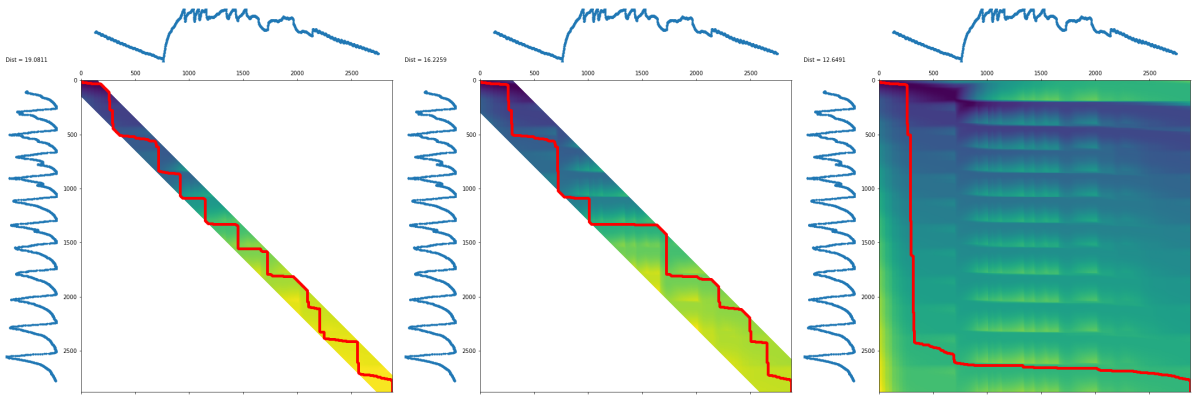


(a) Sakeo-Chiba band width of 5% and DTW distance of 14.35.

(b) Sakeo-Chiba band width of 10% and DTW distance of 13.17.

(c) No constraints and DTW distance of 11.45.

Figure C.1.5: Plotting of the warped path between time series of the gearbox temperature for turbine 1 and turbine 12 in the 'Univariate_V1' data set.



(a) Sakeo-Chiba band width of 5% and DTW distance of 19.08.

(b) Sakeo-Chiba band width of 10% and DTW distance of 16.23.

(c) No constraints and DTW distance of 12.65

Figure C.1.6: Plotting of the warped path between time series of the gearbox temperature for turbine 2 and turbine 12 in the 'Univariate_V1' data set.

Based on the results in Figure C.1.3, C.1.4 and C.1.5, any of the band widths results in a clear indication of the similarity and no significant distortion or warping, which cannot be explained, is observed. However, reviewing Figure C.1.6, a very large number of points can be observed to be mapped to a single point. This case is an unwanted case as the similarity in shape between the two time series is lost: The initial time points of the first time series (horizontal one) are compared to pretty much the entire second time series; the last percentage of the second time series (vertical) is compared to the majority of the first time series. This is avoided by applying the

global constraint, Sakeo-chiba band, to the calculations. Therefore, the DTW algorithm will be included with Sakeo-Chiba band constraint. As most of the literature uses a width of 10%, the difference between 5% and 10% was insignificant, and no detailed analysis of the clustering accuracy can be performed, a width of 10% is chosen for all cluster analysis with DTW as the similarity measure.

C.2: Comparison between different DTW implementations

In this appendix, several different python libraries will be implemented on a test data sets and the running time will be compared to one another. The comparison between libraries for the unconstrained case can be seen in Table C.1.1 and the constrained case (i.e. included a Sakeo-chiba band of width 10% of length of the time series) is presented in Table C.1.2. All computations and cluster analysis are performed on a HP EliteBook with Intel Core i7-3520M processor and 8GB of RAM.

Table C.1.1: Running time of different implementations of the DTW distance metric for the unconstrained case. Width = 10% means 10% of the length of the time series.

Running time different libraries unconstrained			
Library	**Arguments**	**Distance**	**Time [s]**
from cdtw import pydtw	dist = 'euclid', step = 'dp2', window = 'nowindow', compute_path = False	18.42	1.441
from fastdtw import fastdtw	dist=euclidean	913.00	3.577
from tslearn.metrics import dtw_path	-	18.42	21.63
from tslearn.metrics import cdist_dtw	-	18.42	21.60
from dtaidistance import dtw.distance_matrix	-	18.42	98.70
from dtaidistance import dtw.distance_matrix	use_c = True	18.42	57.68
from dtaidistance import dtw.distance_matrix	use_c = True, parallel = True	18.42	59.63
from dtaidistance import dtw.distance_matrix	use_nogil = True	18.42	97.04
from dtaidistance import dtw.distance_matrix	use_nogil = True, parallel = True	18.42	82.00
from dtaidistance import dtw.distance_matrix_fast	-	18.42	**1.25**

Table C.1.2: Running time of different implementations of the DTW distance metric for the constrained case. Width = 10% means 10% of the length of the time series.

Running time different libraries constrained			
Library	**Arguments**	**Distance**	**Time [s]**
from cdtw import pydtw	N/A	N/A	N/A
from fastdtw import fastdtw	N/A	N/A	N/A
from tslearn.metrics import dtw_path	global_constraint='sakoe_chiba', sakoe_chiba_radius=width	39.48	22.36
from tslearn.metrics import cdist_dtw	global_constraint='sakoe_chiba', sakoe_chiba_radius=width	39.48	22.54
from dtaidistance import dtw.distance_matrix	width = 10%	39.57	15.24
from dtaidistance import dtw.distance_matrix	use_c = True, width = 10%	39.57	**11.19**
from dtaidistance import dtw.distance_matrix	use_c = True, parallel = True, width = 10%	39.57	11.88
from dtaidistance import dtw.distance_matrix	use_nogil = True, width = 10%	39.57	19.09
from dtaidistance import dtw.distance_matrix	use_nogil = True, parallel = True, width = 10%	39.57	16.21
from dtaidistance import dtw.distance_matrix_fast	width = 10%	NOT WORKING!	NOT WORKING!

For the unconstrained case, the clear winner is the function call '*dtw.distance_matrix_fast*' from the library '*dtaidistance*' with running time of only 1.25 seconds. For the constrained case, the winner is the function call '*dtw.distance_matrix*' from the library '*dtaidistance*' with the listed arguments in Table C.1.2. Including the *Sakoe-chiba band* for the winner in unconstrained case, '*dtw.distance_matrix_fast*', caused the script to crash. In the documentations it is stated that it support these arguments, but for some reason it does not seem to work.

Appendix D - Python code

D.1: Python script for clustering the data set with hierarchical clustering - both objectives.

In this appendix, the python scripts for clustering the data set with hierarchical clustering - both objectives - is presented. In Appendix D.1.1, the libraries which need to be imported prior to running the script is presented. In Appendix D.1.2, the class for importing and preprocessing of the time series is presented. The path to the .csv and the names of the different files must be changed according to the path chosen for anyone trying to run this code. Then, in Appendix D.1.3, the class for clustering the extracted time series according to the hierarchical clustering algorithm is presented. The class allows for two similarity measures: Euclidean or DTW distance. The procedure for clustering with the objectives of similarity in time or similarity in shape is outlined in the class description. In Appendix D.1.4, the class for calculating and displaying the internal indexes is provided. The procedure for calculation and presenting the internal indexes is included in the class description. Finally, the main function of running the clustering techniques for all data sets is provided in Appendix D.1.5. The different implementation details are outlined in this script. This includes the length of the moving average filter, the size of the window used for the *Sakoe-Chiba band* and the alpha values. The procedure for clustering the time series are:

1. Import the different libraries outlined in the script in Appendix D.1.1. If these are not installed prior to running the script, do so first.
2. Run the following scripts in the following order: D.1.2, D.1.3 and D.1.4 (appendices).
3. In script in Appendix D.1.5, uncomment the data set which is going to be clustered, and comment out the rest. The example shown in the script in Appendix D.1.5, illustrates what to uncomment and what to comment out for clustering the data set called 'Univariate_V1'.
4. Modifications can be made to the length of the moving average filter, the size of the window used for the *Sakoe-Chiba band* and the alpha values.

D.1.1: The libraries which needs to be imported in order for the hierarchical clustering algorithm for both objectives of similarity in time and similarity in shape.

```
#!/usr/bin/env python
# coding: utf-8

# In[ ]:


import pandas as pd
from datetime import datetime
import time
import numpy as np
import matplotlib.pyplot as plt

# Filtering
from collections import deque
from bisect import insort, bisect_left
from itertools import islice


# Clustering
import timeit
from scipy.spatial.distance import euclidean
import scipy.spatial.distance as ssd
from dtaidistance import dtw
from joblib import Parallel, delayed
import multiprocessing
from cycler import cycler


# Internal indexes
from scipy.cluster.hierarchy import cophenet
from sklearn.metrics import silhouette_score
from scipy.spatial.distance import squareform
from scipy.cluster.hierarchy import fcluster
from scipy.cluster.hierarchy import dendrogram
import scipy.cluster.hierarchy as hac
```

D.1.2: Class for the importing and preprocessing of the time series.

```
#!/usr/bin/env python
# coding: utf-8

# In[ ]:


class Dataframe_extraction_preprocessing:
    """ Class used to the extraction of data from .csv files
        The extraction includes setting the time, handling of missing values and
        removing turbines which does not contain measurements.
        dataset_name us required because for one data set the filled values are wrongly filled."""
    def __init__(self, path_csv, filenames, dataset_name, skip_rows):
        self.dataframes = []
        self.dataframes_norm = []
        self.dataframes_scaled = []

        # Importing into dataframes
        for idx, filename in enumerate(filenames):
            self.dataframes.append(pd.read_csv(path_csv+filename+'.csv', delimiter=',', skiprows = skip_rows[idx]))

        self.setting_time_dataframe()       # Setting time for dataframes
        self.filling_missing_values(dataset_name)        # Handling missing values
        self.remove_turbines_with_no_measurement()

    def setting_time_dataframe(self):
        """ Setting date time for the dataframe if not already done
            Also renames the columns to the correct turbine number """
        for idx, df in enumerate(self.dataframes):
            try:
                df['Time'] = pd.to_datetime(df['Time'], format='%Y-%m-%dT%H:%M:%S.%fZ')
                df.set_index('Time', inplace=True)
                df.columns = ['WT1', 'WT2', 'WT3','WT5', 'WT6', 'WT7', 'WT8', 'WT9',
                              'WT10', 'WT11', 'WT12', 'WT13', 'WT14', 'WT15', 'WT16']
            except:
                try:
                    df.columns = ['WT1', 'WT2', 'WT3', 'WT4', 'WT5', 'WT6', 'WT7', 'WT8', 'WT9',
                                  'WT10', 'WT11', 'WT12', 'WT13', 'WT14', 'WT15', 'WT16']
                except:
                    print("Dataframe {} does not fit column names".format(idx+1))
                    print(df.columns)


    def filling_missing_values(self, dataset_name):
        # The function fills in wrong for this timeseries associated with 'Mulitvariate_v2' data set
        if dataset_name == 'Multivariate_v2':
            self.dataframes[0]['WT2'].fillna(0, inplace=True)
        for df in self.dataframes:
            number_of_NaN_before = df.isna().sum().sum()
            df.fillna(method='ffill', inplace=True) # Filling inn all NaN with forward filling
            df.fillna(method='bfill', inplace=True) # In case the first values are NaN, do backwards filling.
            number_of_NaN_after = df.isna().sum().sum()
            print("Number of NA fields in dataset", number_of_NaN_before)
            print("Number of NA fields in dataset after filling: ", number_of_NaN_after,"/",number_of_NaN_before)


    def remove_turbines_with_no_measurement(self):
        """ Removes turbine with no measurments
            and corrects the shape of the timeseries"""
        # Finding the turbine which contains missing values
        list_df = []
        for df in self.dataframes:
            number_of_NaN = df.isna().sum()
            for idx in np.where(number_of_NaN > 0)[0]:
                if df.columns[idx][:6] not in list_df:
                    list_df.append(df.columns[idx][:6])

        # Removing the column containing nan from all dataframes
        for idx, df in enumerate(self.dataframes):
            for index_name in list_df:
                try:
                    df.drop([index_name+df.columns[0][6:]], axis=1, inplace=True)
                except:
                    print(index_name+df.columns[0][6:]+" Does not exist in df", idx + 1)

        # Fixing timestamps such that they are equal
        self.print_shape_time_interval()
        dataframes_new = []
        for idx, df in enumerate(self.dataframes):
            dataframes_new.append(df[:self.dataframes[0].shape[0]])
        self.dataframes = dataframes_new
        self.print_shape_time_interval()


```

```
    def print_shape_time_interval(self):
        for idx, df in enumerate(self.dataframes):
            print("Shape of dataframe{0}: {1},".format(idx+1, df.shape), end='        ')
            print("Start/stop", df.index[0] ,"/", df.index[-1])


    def running_median_insort(self,seq, window_size): # Faster approach to simple moving median filter
        """Contributed by Peter Otten"""
        seq = iter(seq)
        d = deque()
        s = []
        result = []
        for item in islice(seq, window_size):
            d.append(item)
            insort(s, item)
            result.append(s[len(d)//2])
        m = window_size // 2
        for item in seq:
            old = d.popleft()
            d.append(item)
            del s[bisect_left(s, old)]
            insort(s, item)
            result.append(s[m])
        return np.asarray(result)


    # Filters the dataset in the object with a preent lenght of the moving window
    def filter_median(self, length_moving_median):
        for df in self.dataframes:
            for i in range(df.shape[1]):
                df.values[:,i] = self.running_median_insort(df.values[:,i], length_moving_median)


    def normalization(self):
        for df in self.dataframes:
            self.dataframes_norm.append((df - np.min(df.values, axis=0)) / (np.max(df.values, axis=0) -
                                                                            np.min(df.values, axis=0)))


    def scaling(self):
        for df in self.dataframes:
            temp = (df - np.min(np.min(df, axis=0)))
            temp /= np.max(np.max(temp, axis=0))
            self.dataframes_scaled.append(temp)

    # Helper function for plotting the different data sets
    # initial, filtered, scaled, normalised data set
    def plotting_dataframe(self, dataframe, title="No title"):
            style_code = ['-', '-', '-', '-', '-', '-', '-', '-', '-', '-',
                  '--','--','--','--','--','--','--','--','--','--',]
            print("\n",title,"\n")
            for df in dataframe:
                df.plot(figsize=(18,10), subplots=True, layout=(4,4), sharex=True, sharey=True, grid=False,
                        style=style_code, x_compat=True, fontsize= 10)
                plt.gca()
                plt.show()


```

D.1.3: Class for doing the hierarchical clustering with the Euclidean distance or the DTW distance.

```
#!/usr/bin/env python
# coding: utf-8

# In[ ]:


class Clustering_hierarchical:
    """ Clustering is performed on the dissimilarity matrix chosen.
        Similarity in time: Uses the scaled data set and the calculation of the euclidean distance matrix
            1) Initilise the dataframes_scaled
            2) Call the distance matrix function 'distance_matrix_euc' with the corresponding weigthing vector
            3) Cluster and plot the dendrogram with 'plot_dendrogram' function
            4) Cut the dendrogram and plot the clusters with 'print_clusters' function

        Similarity in shape: Uses the normalised data set and the calculation of the DTW distance matrix.
            1) Initilise the dataframes_norm
            2) Call the distance matrix function 'distance_matrix_dtw' with the corresponding weigthing vector
            3) Same as similarity in time
            4) Same as similarity in time
            """
    def __init__(self, dataframes_norm = None, dataframes_scaled = None):
        self.dataframes_norm = dataframes_norm
        self.dataframes_scaled = dataframes_scaled
        self.dist_cond = None
        self.dist_cond_original = None



    def distance_matrix_euc(self, alphas):
        """ Alphas is the weighting vector of length equal to the number of dimension in the data set """
        # Initializing an empty condensed distance matrix
        condensed_matrix_size = 0
        for i in range(1,self.dataframes_scaled[0].shape[1]):
            condensed_matrix_size += i
        dist_cond = np.zeros([condensed_matrix_size,])
        dist_cond_original = []

        # Calculating the distance
        start = timeit.default_timer()
        for idx, df_s in enumerate(self.dataframes_scaled):
            dist_cond_original.append(ssd.pdist(df_s.T.values, metric='euclidean'))
            if alphas[idx] != 0:
                dist_cond = np.add(dist_cond, alphas[idx]*dist_cond_original[idx])

        end = timeit.default_timer()
        print("Time: ",end-start)
        print("Shape of dist: ", dist_cond.shape)

        # Mean condenced distance vector
        sum_alphas = sum(alphas)
        dist_cond /= sum_alphas

        self.dist_cond = dist_cond
        self.dist_cond_original = dist_cond_original



    def distance_matrix_dtw(self, alphas, window_size):
        """ Alphas is the weighting vector of length equal to the number of dimension in the data set """
        num_cores = multiprocessing.cpu_count()

        start = timeit.default_timer()
        temp = Parallel(n_jobs=num_cores)( delayed(dtw.distance_matrix)(df.T.values,
                use_c=True, show_progress=True, window=window_size) for df in self.dataframes_norm )
        end = timeit.default_timer()
        print("Time in seconds for calculating the dtw distance matrix: ",end-start)


        # Constructing the original untampered distance matrix; not including where alphas is zero.
        count_nonzero = np.count_nonzero(alphas)
        iterations = [1 for v in range(count_nonzero - 1)]
        dist_squared_original = temp[0]
        for i in iterations:
            dist_squared_original = np.add(dist_squared_original, temp[i])
        print(dist_squared_original.shape)

        # Transfoming to condensed distance matrix (n,)
        listing = []
        for i in range(dist_squared_original.shape[0]):
            for j in range(i+1,dist_squared_original.shape[1]):
                listing.append(dist_squared_original[i,j])

        dist_cond_original = np.asarray(listing)
```

```
        print(dist_cond_original.shape)


        # Combining the dissimilarity matrices: Used in the cluster analysis
        dist_squared = alphas[0]*temp[0]

        for i in range(1,len(temp)):
            if alphas[i] != 0:
                dist_squared = np.add(dist_squared, alphas[i]*temp[i])
        print(dist_squared.shape)

        # Transfoming to condensed distance matrix (n,)
        listing = []
        for i in range(dist_squared.shape[0]):
            for j in range(i+1,dist_squared.shape[1]):
                listing.append(dist_squared[i,j])

        dist_cond = np.asarray(listing)
        dist_cond.shape

        # Mean condenced distance vector
        sum_alphas = sum(alphas)
        dist_cond /= sum_alphas
        dist_cond_original /= count_nonzero

        self.dist_cond = dist_cond
        self.dist_cond_original = dist_cond_original



    # Helper function for plotting the dendrogram
    def fancy_dendrogram(self, *args, **kwargs):
        max_d = kwargs.pop('max_d', None)
        if max_d and 'color_threshold' not in kwargs:
            kwargs['color_threshold'] = max_d
        annotate_above = kwargs.pop('annotate_above', 0)

        ddata = dendrogram(*args, **kwargs)

        if not kwargs.get('no_plot', False):
            plt.title(r'Hierarchical Clustering Dendrogram - linkage: '+method+', metric: Euclidean', fontsize=20)
            plt.xlabel(r'sample index', fontsize=18)
            plt.ylabel(r'distance', fontsize=18)
            plt.yticks(fontsize=15)
            for i, d, c in zip(ddata['icoord'], ddata['dcoord'], ddata['color_list']):
                x = 0.5 * sum(i[1:3])
                y = d[1]
                if y > annotate_above:
                    plt.plot(x, y, 'o', c=c)
                    #plt.annotate("%.3g" % y, (x, y), xytext=(0, -5),
                                 #textcoords='offset points',
                                 #va='top', ha='center')
            if max_d:
                plt.axhline(y=max_d, linestyle='--', c='k')
        return ddata


    # Doing hierarchical clustering and plotting the dendrogram
    def plot_dendrogram(self, method, metric, cut_distance, dataframes,  path_pictures, file_ending):
        """ This method performes hierarchical clustering by using the hierarchical clustering
            algorithm of SciPy.
            method: The linkage criteria of the hierarchical clustering algorithm. See SciPy doc.
            metric: The internal distance metric. See SciPy doc.
            cut_distance: Plotting a horizontal line which indicates the cut of the dendrogram.
                          If no cut is wanted increase the number above the larges plotted distance
                          value in the plot
            dataframes: Only used to get the columns of the dataframes which will be the sample index
                        of the dendrogram.
            path_pictrure: Path where the dendrogram will be saved.
            file_ending: file ending of the saved dendrogram"""
        Z = hac.linkage(self.dist_cond, method=method, metric=metric,optimal_ordering =True)
        # Plot dendogram
        fig = plt.figure(figsize=(20, 10))
        self.fancy_dendrogram(
            Z,
            leaf_rotation=0.,  # rotates the x axis labels
            leaf_font_size=15.,  # font size for the x axis labels
            labels = dataframes[0].columns,
            color_threshold = cut_distance,
            max_d=cut_distance,  # plot a horizontal cut-off line
        )


        fig.savefig(path_pictures+'dendrogram_'+method+'_'+metric+file_ending+'.eps', bbox_inches='tight',
                    pad_inches=0)
```

```
        plt.show()


    def print_clusters(self, dataframes, method, metric, k, count_nonzero, path_pictures, save_plot=False):
        """ This method performes hierarchical clustering by using the hierarchical clustering algorithm of
            SciPy and printing the different clusters.
            dataframes: For plotting of the time series assigned to each cluster
            method: The linkage criteria of the hierarchical clustering algorithm. See SciPy doc.
            metric: The internal distance metric. See SciPy doc.
            k: number of clusters.
            count_nonzero: Number of nonzero elements in the weigthing vector alpha.
            path_pictrure: Path where the dendrogram will be saved."""
        style_code = ['-', '-', '-', '-', '-', '-', '-', '-', '-', '-',
                      '--', '--', '--', '--', '--', '--', '--', '--', '--', '--']
        style_color = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd',
                       '#8c564b', '#e377c2', '#7f7f7f', '#bcbd22', '#17becf',
                       '#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd',
                       '#8c564b', '#e377c2', '#7f7f7f', '#bcbd22', '#17becf']
        # k Number of clusters I'd like to extract
        Z = hac.linkage(self.dist_cond, method=method, metric=metric, optimal_ordering=True)
        results = fcluster(Z, k, criterion='maxclust')

        # check the results
        s = pd.Series(results)
        clusters = s.unique()
        clusters.sort()
        print(clusters)
        #fig, axarr = plt.subplots(num_cuts, length_dataframe,figsize=(20,15))
        fig, axarr = plt.subplots(count_nonzero, num_cuts, figsize=(20,15))
        if count_nonzero > 1:
            for idx, c in enumerate(clusters):
                cluster_indeces = s[s==c].index
                print("Cluster %d number of entries %d" % (c, len(cluster_indeces)))
                for num, df in enumerate(dataframes):
                    custom_cycle = (cycler(color=style_color) +
                                    cycler(linestyle=style_code))

                    axarr[num,idx].set_prop_cycle(custom_cycle)
                    axarr[num,idx].plot(df.index, df.iloc[:,cluster_indeces])
                    plt.gcf().autofmt_xdate()


                axarr[0,idx].legend(dataframes[0].columns[cluster_indeces], loc='upper right', ncol=4,
                                    fontsize=8, framealpha = 0.7)
                axarr[0, idx].set_title("Cluster "+str(idx+1), fontsize=20)
                # axarr[0, idx].set_ylim((-50,2600)) # Used for power time series
                # axarr[1, idx].set_ylim((5,70))     # Generator speed
                # axarr[3, idx].set_ylim((0,27))     # Wind speed
                #axarr[5, idx].set_ylim((-200,200))  # Position deviation or wind direction
            for num, df in enumerate(dataframes):
                axarr[num,0].set_ylabel(df_names[num], fontsize=15)

        elif count_nonzero == 1:
            for idx, c in enumerate(clusters):
                cluster_indeces = s[s==c].index
                print("Cluster %d number of entries %d" % (c, len(cluster_indeces)))
                df = dataframes[0]
                custom_cycle = (cycler(color=style_color) +
                                cycler(linestyle=style_code))
                axarr[idx].set_prop_cycle(custom_cycle)
                axarr[idx].plot(df.index, df.iloc[:,cluster_indeces])
                plt.gcf().autofmt_xdate()

                axarr[idx].legend(df.columns[cluster_indeces], loc='upper right', ncol=4, fontsize=8,
                                  framealpha = 0.7)
                axarr[idx].set_title("Cluster "+str(idx+1), fontsize=20)
                # axarr[idx].set_ylim((-50,2600)) # Used for power time series
        else:
            print("Do nothing")

        if save_plot:
            plt.savefig(path_pictures+'subplots_'+method+'_numCut'+str(k)+'_dim4.pdf', bbox_inches='tight',
                        pad_inches=0)
```

D.1.4: Class for calculating and displaying the internal indexes.

```
#!/usr/bin/env python
# coding: utf-8

# In[ ]:


class Internal_indexes:
    """ Class for calculating and plotting of the internal indexes.
        dist_cond: 1D condensed distance matrix calculated with the the weighting vector, as used in SciPy.
        dist_cond_original: 1D condensed distance matrix calculated without the weighting vector (all alphas equal to 1)
        count_nonzero: Number of nonzero elements in Alpha vector.

        Procedure for calculating and presenting the itnernal indexes
        1) First initate the class witht du corresponding distance matrices, count_nonzeroes and dataframes_scaled
           for the euclidean distance or random dataframe for the dtw distance
        2) Call the 'calculation_internal_indexes' with the list of methods allowed in the SciPy notation and valid metric
           'euclidean' or 'dtw'.
        3) Call 'plotting_internal_indexes' for potting of the internal indexs
        """

    def __init__(self, dist_cond, dist_cond_original, count_nonzero, dataframes):
        self.dist_cond = dist_cond
        self.dist_cond_original = dist_cond_original
        self.count_nonzero = count_nonzero
        self.dataframes = dataframes
        self.correlations = {}
        self.silhouettes = {}
        self.mean_sum_squared = {}

    def cophenet_index(self, Z):
        """Calculates the cophenet correlation coefficent for a given dendrogram, Z"""
        c = cophenet(Z)
        corr_value = np.corrcoef(c,self.dist_cond)[0,1]
        return corr_value



    def silhouette_index(self, Z):
        """ Calculates the silhouette index for a given dendrogram, Z
            for cuts from 2 to number of (turbines - 2) """
        dist_squared = squareform(self.dist_cond)
        silhouette_scores = []
        for k in range(2,dist_squared.shape[0] - 1):
            labels = fcluster(Z, k, criterion='maxclust')
            silhouette_scores.append(silhouette_score(X=dist_squared, labels=labels, metric="precomputed"))
        return silhouette_scores


    # Helper function for sum_squared_error_euc and sum_squared_error_dtw functions
    def mssse(self, columns, SSE):
        """ Calculates the average sum of SSE for all clusters formed. """
        MSSSE = 0
        for index_name in columns:
            MSSSE += SSE[index_name]
        MSSSE /= len(columns)
        return MSSSE



    def sum_squared_error_euc(self, Z, current_scaled_dataframes):
        dist_squared_list = []
        # Converting list of 1d condensed matrices to square form
        for idx, element in enumerate(self.dist_cond_original):
            dist_squared_list.append(squareform(element))

        mean_sum_of_SSE = []
        for num_cuts in range(2,dist_squared_list[0].shape[0] - 1):
            # Getting labels
            labels = fcluster(Z, num_cuts, criterion='maxclust')
            s = pd.Series(labels)
            clusters = s.unique()
            clusters.sort()

            cluster_vectors = pd.DataFrame()
            sum_squared_error = {}  # dict for storing the SSE
            for c in clusters:
                cluster_indeces = s[s==c].index
                # Caluclating the mean vector for df in scaled_data = [df1, df2, df3, ...]
                for idx, df in enumerate(current_scaled_dataframes):
                    cluster_vectors['Cluster{0}_{1}'.format(c,idx)] = df[df.columns[cluster_indeces]].sum(axis=1)/
                                                                      len(cluster_indeces)

                # Getting the sum of squared error between all timeseries in the cluster to its cluster centre
```

```
                # This is repeated for each dimension scaled_data = [df1, df2, df3, ...] and than the mean
                # is extracted
                squared_error = {}
                for idx, df in enumerate(current_scaled_dataframes): # For [df1,df2,df3]
                    temp = 0
                    for timeseries in cluster_indeces: #Indces for the timeseres within the current cluster
                        temp2 = np.sum( (df[df.columns[timeseries]] -
                                         cluster_vectors['Cluster{0}_{1}'.format(c,idx)])**2 )
                        squared_error[df.columns[timeseries]+'_cluster_'+str(idx)] = temp2

                # Initializing each sum to zero
                for timeseries in cluster_indeces:  # indces for the timeseres within the current cluster
                    sum_squared_error[df.columns[timeseries]] = 0

                # Summing each squared error associated with the same turbine (f.eks WT2)
                # from all dimension in scaled_dataframes: df1, df2, df3, ...
                for idx, df in enumerate(current_scaled_dataframes): # For [df1,df2,df3]
                    for timeseries in cluster_indeces:
                        sum_squared_error[df.columns[timeseries]] += squared_error[df.columns[timeseries]+
                                                                                   '_cluster_'+str(idx)]


            # Dividing all errors by the number of dimension to get a comparable value
            for idx in range(len(sum_squared_error)):
                sum_squared_error[df.columns[idx]] /= len(current_scaled_dataframes)


            #Mean sum of SSE - as above, to get a comparable value
            temp = self.mssse(current_scaled_dataframes[0].columns, sum_squared_error)
            mean_sum_of_SSE.append(temp)

        return mean_sum_of_SSE


    def get_centroid(self, cluster_indeces, dist_squared):
        # Cluster_indeces is all timeseries in that cluster with index values. Same as the function returns.
        centroid_distances = []
        for ts in cluster_indeces:
            average_distance = dist_squared[ts,cluster_indeces.drop(ts)].sum(axis=0)
            centroid_distances.append(average_distance)

        # Returns index from [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14]
        # Indexes associated ['WT1','WT2','WT3','WT5','WT6','WT7','WT8','WT9','WT10','WT11','WT12','WT13',
        # 'WT14','WT15','WT16']
        centroid_index = cluster_indeces[centroid_distances.index(min(centroid_distances))]
        return centroid_index



    def sum_squared_error_dtw(self, Z, columns):
        dist_squared = squareform(self.dist_cond)
        mean_sum_of_SSE = []
        for num_cuts in range(2,dist_squared.shape[0] - 1):
            # Getting labels
            labels = fcluster(Z, num_cuts, criterion='maxclust')
            s = pd.Series(labels)
            clusters = s.unique()
            clusters.sort()

            centroid_indices = []
            sum_squared_error = {}  # dict for storing the SSE
            for c in clusters:
                cluster_indeces = s[s==c].index

                # Finding the centroid for each cluster
                centroid_index = self.get_centroid(cluster_indeces, dist_squared)
                centroid_indices.append(centroid_index)

                # Calculating the sum of squared error for each timeseries to its centroid
                for ts1 in cluster_indeces:
                    sum_squared_error[columns[ts1]] = dist_squared[centroid_index, ts1]
                    #print(centroid_index, ts1, dist_squared[centroid_index, ts1])

            # Overall Mean sum of SSE
            temp = self.mssse(columns, sum_squared_error)
            mean_sum_of_SSE.append(temp)

        return mean_sum_of_SSE



    def calculation_internal_indexes(self, methods, metric, similarity_measure = "euclidean"):
        """ metric and methods limited to those avaiable in the SciPy hac library.
            methods: should be a list containing the names of linkage criterias.
                    Example: methods = ['single', 'complete'] or methods = ['single']
```

```
            metric: internal distance, measure used in the linakge criteria
            similarity_measure: can either be the 'euclidean' or the 'dtw'"""
        for method in methods:
            Z = hac.linkage(self.dist_cond, method=method, metric=metric, optimal_ordering=True)
            self.correlations[method] = self.cophenet_index(Z)
            self.silhouettes[method] = self.silhouette_index(Z)
            if similarity_measure == "euclidean":
                self.mean_sum_squared[method] = self.sum_squared_error_euc(Z, self.dataframes[:self.count_nonzero])
            elif similarity_measure == "dtw":
                self.mean_sum_squared[method] = self.sum_squared_error_dtw(Z, self.dataframes[0].columns)
            else:
                print("Invalid similarity measure. Should be either 'euclidean' or 'dtw'. MSSSE is not calculated")

            print("Method: "+method)
            print("Correlation: {:.3f}".format(self.correlations[method]))
            print("Silhouettes:   \t\t", end='')
            print(' [' + ';'.join('%6.2f' % v for v in self.silhouettes[method]) + ']')
            if self.mean_sum_squared:
                print("Mean sum of SSE: \t", end='')
                print(' [' + ';'.join('%6.2f' % v for v in self.mean_sum_squared[method]) + ']\n')

        for method in methods:
            print(';'.join('%6.2f' % v for v in self.silhouettes[method]))
            if self.mean_sum_squared:
                print(';'.join('%6.2f' % v for v in self.mean_sum_squared[method]))



    def plotting_internal_indexes(self, similarity_measure, dataset_name, cophenet_yaxis,
                                  file_ending, save_figure=False):
        """ similarity_measure, dataset_name and file_ending are all used to determine the
            name of the figure saved. cophenet_yaxis: number which indicates where the
            cophenet index values should be located in the plot (y-axis)."""
        methods = ['single', 'complete', 'average', 'ward']
        fig, ax = plt.subplots(2, 2,figsize=(20,12), sharex=True, sharey=True)

        x_data = [] # X-axis for the coefficents
        dist_squared = squareform(self.dist_cond)
        [x_data.append(cuts) for cuts in range(2,dist_squared.shape[0] - 1)]
        temp = -1 # Used to iterate the subplots
        for idx, method in enumerate(methods):
            if idx%2 == 0:
                temp += 1
            color = 'tab:red'
            ax[temp,idx%2].grid(True)
            ax[temp,idx%2].set_title(method, fontsize=25)
            ax[temp,idx%2].plot(x_data, self.silhouettes[method],'-o', color=color, label = 'Silhouette index')
            ax[temp,idx%2].tick_params(axis='y', labelcolor=color, labelsize=15)
            ax[temp,idx%2].text(10,cophenet_yaxis,"Cophenet index: {:.3f}".format(self.correlations[method]),
                                fontsize=15)
            ax[temp,idx%2].tick_params(axis='x', labelsize=15)
            ax[temp,idx%2].set_ylabel('Silhouette index', color=color, fontsize=20)
            ax2 = ax[temp,idx%2].twinx()  # instantiate a second axes that shares the same x-axis

            color = 'tab:blue'
            ax2.plot(x_data, self.mean_sum_squared[method], '-o', color=color, label = 'Mean sum of SSE')
            ax2.tick_params(axis='y', labelcolor=color, labelsize=15)
            ax2.set_ylabel('Mean sum of SSE', color=color, fontsize=20)


        ax[1,0].set_xlabel('Number of cuts', labelpad=0, fontsize=20)
        ax[1,1].set_xlabel('Number of cuts', labelpad=0, fontsize=20)

        fig.tight_layout()  # otherwise the right y-label is slightly clipped
        plt.show()

        if save_figure:
            path_pictures = 'C:/Users/ewaaga/OneDrive - KONGSBERG MARITIME AS/Pictures/Performance indicators/'
            fig.savefig(path_pictures+'Internal_indices_'+similarity_measure+'_'+dataset_name+file_ending+'.eps',
                        bbox_inches='tight', pad_inches=0)
```

D.1.5: Main function which is called after the former scripts has been run.

For running the 'Univariate_V1' cluster analysis, leave it as it is and run the scripts. For running the 'Univariate_V2' comment out line number 11 to 29 and uncomment line 33 to 51. For running the 'Multivariate_V1' cluster analysis, uncomment line 54 to 76 and comment out the other data sets. For running the 'Multivariate_V2' cluster analysis, uncomment line 79 to 102 and comment out the other data sets. Make sure that the path the the data sets is correct.

```
#!/usr/bin/env python
# coding: utf-8

# In[ ]:


if __name__ == "__main__":
    """ For running the different cluster analysis, uncomment the different sections.
        For example, if I want to run the analysis on the UNIVARIATE_v1, I would uncomment
        line number 11 to 29. """
    #### UNIVARIATE_v1 ####
    #Paths
    path_csv = 'C:/Users/ewaaga/OneDrive - KONGSBERG MARITIME AS/Finished_python_code/Data sets/Univariate_v1/'
    path_pictures_euc = 'C:/Users/ewaaga/OneDrive - KONGSBERG MARITIME AS/Pictures/'
    path_pictures_dtw = 'C:/Users/ewaaga/OneDrive - KONGSBERG MARITIME AS/Pictures/'

    # Filenames
    dataset_name = 'Univariate_v1'

    filenames = ['Gearbox_temp_1day_30A_R']

    df_names = ['gearboxTemp']

    skip_rows = [1]
    alphas = [1]
    file_ending = "_alphas_1"
    lenght_moving_average = 5
    window_size_dtw = 280
    ########################



#     #### UNIVARIATE_v2 ####
#     #Paths
#     path_csv = 'C:/Users/ewaaga/OneDrive - KONGSBERG MARITIME AS/Finished_python_code/Data sets/Univariate_v2/'
#     path_pictures_euc = 'C:/Users/ewaaga/OneDrive - KONGSBERG MARITIME AS/Pictures/'
#     path_pictures_dtw = 'C:/Users/ewaaga/OneDrive - KONGSBERG MARITIME AS/Pictures/'

#     # Filenames
#     dataset_name = 'Univariate_v2'

#     filenames = ['Gearbox_temp_1day_30A_R_v2']

#     df_names = ['gearboxTemp']

#     skip_rows = [1]
#     alphas = [1]
#     file_ending = "_alphas_1"
#     lenght_moving_average = 5
#     window_size_dtw = 280
#     ########################


#    #### MULTIVARIATE_V1 #####
#     #Paths
#     path_csv = 'C:/Users/ewaaga/OneDrive - KONGSBERG MARITIME AS/Finished_python_code/Data sets/Multivariate_v1/'
#     path_pictures_euc = 'C:/Users/ewaaga/OneDrive - KONGSBERG MARITIME AS/Pictures/'
#     path_pictures_dtw = 'C:/Users/ewaaga/OneDrive - KONGSBERG MARITIME AS/Pictures/'

#     # Filenames
#     dataset_name = 'Multivariate_v1'


#     filenames = ['Gearbox_temp_1day_30A_R_v2',
#                  'Wind_speed_nascell_v2',
#                  'Generator_speed_v2',
#                  'Positional_derivative_v2_unwrapped']

#     df_names = ['gearboxTemp', 'windSpd','genSpd','posDeviation']

#     skip_rows = [1,1,1,0]
#     alphas = [8,1,1,1]
#     file_ending = "_alphas_8111"
#     lenght_moving_average = 5
```

```
#     window_size_dtw = 280
#     ########################


#     #### MULTIVARIATE_V2 #####
#     # Paths
#     path_csv = 'C:/Users/ewaaga/OneDrive - KONGSBERG MARITIME AS/Finished_python_code/Data sets/Multivariate_v2/'
#     path_pictures_euc = 'C:/Users/ewaaga/OneDrive - KONGSBERG MARITIME AS/Pictures/'
#     path_pictures_dtw = 'C:/Users/ewaaga/OneDrive - KONGSBERG MARITIME AS/Pictures/'

#     # Filenames
#     dataset_name = 'Multivariate_v2'

#     filenames = ['power_mv2',
#                  'gearboxTemp_mv2',
#                  'generatorSpeed_mv2',
#                  'windSpeed_mv2',
#                  'windDir_mv2',
#                  'outsideTemp_mv2']

#     skip_rows = [0,0,0,0,0,0]
#     alphas = [1,1,1,1,0,0]
#     file_ending = "_alphas_111100"
#     lenght_moving_average = 100

#     df_names = ['power', 'gearboxTemp','genSpd', 'windSpd', 'windDir', 'extTemp']
#     window_size_dtw = 620
#     ########################


    # Initiate dataframes
    dataframes = Dataframe_extraction_preprocessing(path_csv, filenames, dataset_name, skip_rows)
    #dataframes.plotting_dataframe(dataframes.dataframes, "Initial plots")

    # Applying filter
    dataframes.filter_median(lenght_moving_average)
    #dataframes.plotting_dataframe(dataframes.dataframes, "Post-filtering plots")

    # Normalization
    dataframes.normalization()
    #dataframes.plotting_dataframe(dataframes.dataframes_norm, "Normalized timeseries plots")

    # Scaling
    dataframes.scaling()
    #dataframes.plotting_dataframe(dataframes.dataframes_scaled, "Scaled timeseries plots")

    # Reformed dataset
    if dataset_name == 'Multivariate_v1':
        dataframes_reformed = [dataframes.dataframes_norm[0], dataframes.dataframes_norm[1],
                               dataframes.dataframes_scaled[2], dataframes.dataframes_norm[3]]


    ### SIMILARITY IN TIME CLUSTER ANALYSIS ###
    ## Distance matrix
    count_nonzero = np.count_nonzero(alphas)

    # Euclidean
    model_euc = Clustering_hierarchical(dataframes_scaled = dataframes.dataframes_scaled)
    model_euc.distance_matrix_euc(alphas)

    ## Internal indexes
    methods = ['single', 'complete', 'average', 'ward']
    metric = 'euclidean' # internal distance, NOT the similarity measure

    print("Euclidean internal indexes")
    similarity_measure = 'euclidean'
    cophenet_yaxis = 0.63  # Adjust this for the location of the Cophenet index in the plot of the dendrogram
    internal_indexes = Internal_indexes(model_euc.dist_cond, model_euc.dist_cond_original,
                                        count_nonzero, dataframes.dataframes_scaled)
    internal_indexes.calculation_internal_indexes(methods, metric, similarity_measure)
    internal_indexes.plotting_internal_indexes(similarity_measure, dataset_name, cophenet_yaxis, file_ending,
                                               save_figure = False)

    # Plotting the dendrogram with the given method
    method = 'single'
    metric = 'euclidean'
    cut_distance = 250000  # Only used for visualising the cut in the dendrogram
    model_euc.plot_dendrogram(method, metric, cut_distance, dataframes.dataframes, path_pictures_euc,
                              file_ending)

    num_cuts = 3 # Cuts the dendrogram into 'num_cuts' partitions
    print(count_nonzero)
    df_names = ['power', 'gearboxTemp','genSpd', 'windSpd', 'windDir', 'extTemp']
    model_euc.print_clusters(dataframes.dataframes_norm[:count_nonzero], method, metric, num_cuts,
                             count_nonzero, path_pictures_euc, save_plot=False)
```

```
160
161
162
163     ### SIMILARITY IN SHAPE CLUSTER ANALYSIS ###
164     ## DTW distance
165     if dataset_name == 'Multivariate_v1':
166         model_dtw = Clustering_hierarchical(dataframes_norm = dataframes_reformed)
167     else:
168         model_dtw = Clustering_hierarchical(dataframes_norm = dataframes.dataframes_norm)
169
170     model_dtw.distance_matrix_dtw(alphas, window_size_dtw)
171
172     ## Internal indexes
173     methods = ['single', 'complete', 'average', 'ward']
174     metric = 'euclidean' # internal distance, NOT the similarity measure
175
176     print("DTW internal indexes")
177     similarity_measure = 'dtw'
178     cophenet_yaxis = 0.72  # Adjust this for the location of the Cophenet index in the plot of the dendrogram
179     internal_indexes = Internal_indexes(model_dtw.dist_cond, model_dtw.dist_cond_original,
180                                         count_nonzero, dataframes.dataframes_scaled)
181     internal_indexes.calculation_internal_indexes(methods, metric, similarity_measure)
182     internal_indexes.plotting_internal_indexes(similarity_measure, dataset_name, cophenet_yaxis, file_ending,
183                                                save_figure=True)
184
185     # Plotting the dendrogram with the given method
186     method = 'single'
187     metric = 'euclidean'
188     cut_distance = 250000  # Only used for visualising the cut in the dendrogram
189     model_dtw.plot_dendrogram(method, metric, cut_distance, dataframes.dataframes, path_pictures_dtw,
190                               file_ending)
191
192     # Plotting the timeseries and the cluster assignment
193     num_cuts = 3 # Cuts the dendrogram into 'num_cuts' partitions
194     print(count_nonzero)
195     df_names = ['power', 'gearboxTemp','genSpd', 'windSpd', 'windDir', 'extTemp']
196     model_dtw.print_clusters(dataframes.dataframes_norm[:count_nonzero], method, metric, num_cuts,
197                              count_nonzero, path_pictures_dtw, save_plot=True)
```

D.2: Python script for the K-means implementation

In this appendix, the Python script for the K-means implementation is presented. This script assumes that the time series are stored in dataframes, which is the results from running the scripts in Appendix D.1. In this implementation, the scaled dataframes are clustered (*dataframes.dataframes_scale[0].T* is called in the script). This is only run for the first data set, 'Univariate_V1', where the time series for the gearbox temperature is clustered.

```
#!/usr/bin/env python
# coding: utf-8

# ## K-means
# Implementation from scipy library
#
# https://docs.scipy.org/doc/scipy/reference/cluster.vq.html#module-scipy.cluster.vq

# In[ ]:


### K-means from scipy.cluster.vq.kmeans

from scipy.cluster.vq import vq, kmeans, kmeans2, whiten
import timeit

num_clusters = 4
num_iterations = 5000

# Codebook: A k by N array of k centroids,
# Distrotion: The mean (non-squared) Euclidean distance between the observations passed and the centroids
#             generated.
# codebook, distortion = kmeans(timeSeries, num_clusters, iter=500)
start = timeit.default_timer()
centroid, _ = kmeans(dataframes.dataframes_scaled[0].T, num_clusters, iter= num_iterations, thresh=1e-08,
                     check_finite=False)
end = timeit.default_timer()

print("Time: ",(end-start)/num_iterations)

# Code: A length M array holding the code book index for each observation.
# dist: The distortion (distance) between the observation and its nearest code.
label, dist = vq(timeSeries, centroid)

print(label, dist)

df_centroid = pd.DataFrame(centroid.transpose())
df_centroid.set_index(time_axis.values, inplace=True)


# In[ ]:


# Getting the cluster centers and plotting them

import matplotlib.pyplot as plt
df_centroid.plot(figsize=(18,10), style=style_code, fontsize=15)
plt.title(r'Gearbox oil temperature', fontsize=20)
plt.ylabel(r'Temperature [C]', fontsize=18)

# Attaching lengeds
legends = []
for i in range(centroid.shape[0]):
    legends.append('Cluster '+str(i+1))
plt.legend(legends, fontsize=13)
plt.savefig(path_pictures+'Cluster_prototypes_numClusters_'+str(num_clusters)+'v2.eps',
            bbox_inches='tight', pad_inches=0)
plt.show()
#bottom, top = [0,1]
#plt.ylim(bottom, top)

# Cluster assignment
print("Cluster assignment: ",label)


# In[ ]:


# Plotting the cluster center along with the time series assigned to that cluster

for idx, cluster in enumerate(centroid):
    # Transforming to panda frames
    labels = pd.Series(label)
    cluster_plot = pd.DataFrame(cluster)
    cluster_plot.set_index(time_axis.values, inplace=True)
    cluster_plot.columns = ['Cluster '+str(idx+1)]
```

```
77
78      # Assigning time-series to correct cluster
79      cluster_indeces = labels[labels==idx].index
80      print("Cluster %d number of entries %d" % (idx+1, len(cluster_indeces)))
81
82      # Plotting the results
83      title = r"Cluster "+str(idx+1)+" - Number of time-series: "+str(len(cluster_indeces))
84      ax = cluster_plot.plot(figsize=(18,8), title = title, style=style_code, fontsize = 20)
85      timeSeries.T.iloc[:,cluster_indeces].plot(ax=ax, style=style_code, alpha=0.3, fontsize = 20)
86      ax.legend(loc='upper right', fontsize=20)
87      ax.set_ylim(bottom, top)
88      fig = ax.get_figure()
```

D.3: Additional python scripts

In this appendix, additional Python scripts are provided. In Appendix D.3.1, the Python script used for calculating and visualising the warping path of the DTW algorithm in Appendix C.1 is provided. In Appendix D.3.2, the Python script for plotting of the dendrogram with the associated time series. Examples of these resulting plots can be seen in for example Figure 6.17 or 6.21. In Appendix D.3.3, the Python script for comparing the different DTW implementations is provided.

D.3.1: Python script used for calculating and visualising the warping path of the DTW algorithm in Appendix C.1.

```
#!/usr/bin/env python
# coding: utf-8

# # Script for visualising the warping path

# In[ ]:


# Plotting of the warping path
from dtaidistance import dtw
from dtaidistance import dtw_visualisation as dtwvis
import numpy as np
t1 = 1  # time sereis number in the dataframe
t2 = 4  # time sereis number in the dataframe

# The dataframes_norm is the same as acquired from the main function.
# Should be run after the dataframes extarction and preprocessing
x = dataframes.dataframes_norm[0].values[:,t1]
y = dataframes.dataframes_norm[0].values[:,t2]


window_size = 300
d, paths = dtw.warping_paths(x, y, window=window_size)
best_path = dtw.best_path(paths)
# dtwvis.plot_warpingpaths(x, y, paths, best_path, filename = "turbine_{}vs{}_window_{}".format(t1,t2,window_size))
dtwvis.plot_warpingpaths(x, y, paths, best_path)


# In[ ]:


window_size = 600
d, paths = dtw.warping_paths(x, y, window=window_size)
best_path = dtw.best_path(paths)
dtwvis.plot_warpingpaths(x, y, paths, best_path)


# In[ ]:


d, paths = dtw.warping_paths(x, y)
best_path = dtw.best_path(paths)
dtwvis.plot_warpingpaths(x, y, paths, best_path)
```

D.3.2: Script for plotting of the dendrogram with the associated time series

```
#!/usr/bin/env python
# coding: utf-8

# # Helper function for visualising thedendrograms and the time series

# In[ ]:


dataframes.dataframes_norm[0].values.shape
timeSeries = dataframes.dataframes_norm[0].T
print(timeSeries.shape)
print(type(timeSeries.values))
print(timeSeries.values.shape)



import timeit
from dtaidistance import clustering
from dtaidistance import dtw
# Custom Hierarchical clustering
model1 = clustering.Hierarchical(dtw.distance_matrix,  {})
# Augment Hierarchical object to keep track of the full tree
model2 = clustering.HierarchicalTree(model1)
# SciPy linkage clustering
model3 = clustering.LinkageTree(dtw.distance_matrix, dists_options={"use_c":True, "window":window_size_dtw})
# use_c =True, window=100
start = timeit.default_timer()
method = 'average'
cluster_idx = model3.fit(timeSeries.values, method = method)
end = timeit.default_timer()

print("Time: ",end-start)
print(model3)

fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(20, 15))
show_ts_label = lambda idx: dataframes.dataframes_norm[0].columns[idx]
model3.plot(path_pictures_dtw+"dendrogram_dtaidistance_2"+method+".png", axes=ax, show_ts_label=show_ts_label,
            show_tr_label=True, ts_label_margin=-200, ts_height = 100,
            ts_left_margin=-50, ts_sample_length=1, tr_label_margin=1, tr_left_margin=0)
plt.show()
```

D.3.3: Script for comparing the different DTW implementations

```
#!/usr/bin/env python
# coding: utf-8

# # Comparison between different DTW implementations

# In[1]:


import timeit
import numpy as np
import matplotlib.pyplot as plt
from dtaidistance import dtw
from dtaidistance import dtw_visualisation as dtwvis
import numpy as np


# In[2]:


idx = np.linspace(0, 10, 5000)
x = 1.4*np.sin(idx)
y = np.cos(idx)

width = len(x)//10
print("Width:", width)
plt.figure(figsize=(15,10))
plt.plot(idx,x,idx,y)


# # DTW unconstrained

# In[3]:


from cdtw import pydtw

start = timeit.default_timer()
d = pydtw.dtw(x,y,pydtw.Settings(dist = 'euclid', step  = 'dp2', window = 'nowindow',
                                 compute_path = False))
end = timeit.default_timer()
print("Time pydtw from cdtw: ",end-start)
print("Distance pydtw from cdtw: ", d.get_dist())


# In[4]:


from scipy.spatial.distance import euclidean
from fastdtw import fastdtw

start = timeit.default_timer()
distance, path = fastdtw(x, y, dist=euclidean)
end = timeit.default_timer()

print("Time: ",end-start)
print("Distance: ", distance)


# In[5]:


from tslearn.metrics import dtw_path

start = timeit.default_timer()
path, distance = dtw_path(x, y) # support
end = timeit.default_timer()
#
print("Time: ",end-start)
print("Distance: ", distance)


# In[6]:


from tslearn.metrics import cdist_dtw

start = timeit.default_timer()
path, distance = cdist_dtw([x,y])
end = timeit.default_timer()
#
print("Time: ",end-start)
print("Distance: ", distance)

```

```
84
85  # In[7]:
86
87
88  from dtaidistance import dtw
89
90  start = timeit.default_timer()
91  distance = dtw.distance_matrix([x,y])
92  end = timeit.default_timer()
93
94  print("Time: ",end-start)
95  print("Distance: ", distance)
96
97  start = timeit.default_timer()
98  distance = dtw.distance_matrix([x,y], use_c = True)
99  end = timeit.default_timer()
100
101 print("Time: ",end-start)
102 print("Distance: ", distance)
103
104 start = timeit.default_timer()
105 distance = dtw.distance_matrix([x,y], use_nogil  = True)
106 end = timeit.default_timer()
107
108 print("Time: ",end-start)
109 print("Distance: ", distance)
110
111 start = timeit.default_timer()
112 distance = dtw.distance_matrix([x,y], use_nogil  = True, parallel = True)
113 end = timeit.default_timer()
114
115 print("Time: ",end-start)
116 print("Distance: ", distance)
117
118 start = timeit.default_timer()
119 distance = dtw.distance_matrix([x,y], use_c  = True, parallel = True)
120 end = timeit.default_timer()
121
122 print("Time: ",end-start)
123 print("Distance: ", distance)
124
125 start = timeit.default_timer()
126 distance = dtw.distance_matrix_fast([x,y])
127 end = timeit.default_timer()
128
129 print("Time: ",end-start)
130 print("Distance: ", distance)
131
132
133 # # DTW contrained, Sakoe-band
134
135 # In[8]:
136
137
138 from tslearn.metrics import dtw_path
139
140 start = timeit.default_timer()
141 path, distance = dtw_path(x, y, global_constraint='sakoe_chiba', sakoe_chiba_radius=width) # support
142 end = timeit.default_timer()
143 #
144 print("Time: ",end-start)
145 print("Distance: ", distance)
146
147
148 # In[9]:
149
150
151 from tslearn.metrics import cdist_dtw
152
153 start = timeit.default_timer()
154 path, distance = cdist_dtw([x,y], global_constraint='sakoe_chiba', sakoe_chiba_radius=width)
155 end = timeit.default_timer()
156 #
157 print("Time: ",end-start)
158 print("Distance: ", distance)
159
160
161 # In[3]:
162
163
164 from dtaidistance import dtw
165
166 start = timeit.default_timer()
167 distance = dtw.distance_matrix([x,y], window=width)
168 end = timeit.default_timer()
```

```
169
170 print("Time: ",end-start)
171 print("Distance: ", distance)
172
173 start = timeit.default_timer()
174 distance = dtw.distance_matrix([x,y], use_c = True, window=width)
175 end = timeit.default_timer()
176
177 print("Time: ",end-start)
178 print("Distance: ", distance)
179
180 start = timeit.default_timer()
181 distance = dtw.distance_matrix([x,y], use_nogil  = True, window=width)
182 end = timeit.default_timer()
183
184 print("Time: ",end-start)
185 print("Distance: ", distance)
186
187 start = timeit.default_timer()
188 distance = dtw.distance_matrix([x,y], use_nogil  = True, parallel = True, window=width)
189 end = timeit.default_timer()
190
191 print("Time: ",end-start)
192 print("Distance: ", distance)
193
194 start = timeit.default_timer()
195 distance = dtw.distance_matrix([x,y], use_c  = True, parallel = True, window=width)
196 end = timeit.default_timer()
197
198 print("Time: ",end-start)
199 print("Distance: ", distance)
200
201
202 # # IMPLEMENTATION DOES NOT WORK!
203 # start = timeit.default_timer()
204 # distance = dtw.distance_matrix_fast([x,y], window=width)
205 # end = timeit.default_timer()
206
207 # # IMPLEMENTATION DOES NOT WORK!
208 # start = timeit.default_timer()
209 # distance = dtw.distance_fast(x,y, window=width)
210 # end = timeit.default_timer()
```

Espen Waaga

Machine Learning for Automatic Classification of Wind Turbines

NTNU
Norwegian University of
Science and Technology

KONGSBERG