UiA University of Agder

Master

Exploration and Performance Analysis of Clustering Algorithms for Time-Series Data with Dimension Reduction

Exploration of pattern detection using dimensionality reduction on time-series data to simplify clustering algorithms, then compare the results to the state-of-the-art.

DANIEL NGUYEN HANSEN,
ALEKSANDER MARKUS LINGSTAD &
MARIUS SUVATNE

SUPERVISORS
LEI JIAO &
REBEKKA OLSSON OMSLANDSETER

University of Agder, 2022
Faculty of Engineering and Science
Department of Information and Communication Technology

Obligatorisk gruppeerklæring

Den enkelte student er selv ansvarlig for å sette seg inn i hva som er lovlige hjelpemidler, retningslinjer for bruk av disse og regler om kildebruk. Erklæringen skal bevisstgjøre studentene på deres ansvar og hvilke konsekvenser fusk kan medføre. Manglende erklæring fritar ikke studentene fra sitt ansvar.

1.	Vi erklærer herved at vår besvarelse er vårt eget arbeid, og at vi ikke har brukt andre kilder eller har mottatt annen hjelp enn det som er nevnt i besvarelsen.	Ja
2.	**Vi erklærer videre at denne besvarelsen:** • Ikke har vært brukt til annen eksamen ved annen avdeling/universitet/høgskole innenlands eller utenlands. • Ikke refererer til andres arbeid uten at det er oppgitt. • Ikke refererer til eget tidligere arbeid uten at det er oppgitt. • Har alle referansene oppgitt i litteraturlisten. • Ikke er en kopi, duplikat eller avskrift av andres arbeid eller besvarelse.	Ja
3.	Vi er kjent med at brudd på ovennevnte er å betrakte som fusk og kan medføre annullering av eksamen og utestengelse fra universiteter og høgskoler i Norge, jf. Universitets- og høgskoleloven §§4-7 og 4-8 og Forskrift om eksamen §§ 31.	Ja
4.	Vi er kjent med at alle innleverte oppgaver kan bli plagiatkontrollert.	Ja
5.	Vi er kjent med at Universitetet i Agder vil behandle alle saker hvor det forligger mistanke om fusk etter høgskolens retningslinjer for behandling av saker om fusk.	Ja
6.	Vi har satt oss inn i regler og retningslinjer i bruk av kilder og referanser på biblioteket sine nettsider.	Ja
7.	Vi har i flertall blitt enige om at innsatsen innad i gruppen er merkbart forskjellig og ønsker dermed å vurderes individuelt. Ordinært vurderes alle deltakere i prosjektet samlet.	Nei

Publiseringsavtale

Fullmakt til elektronisk publisering av oppgaven Forfatter(ne) har opphavsrett til oppgaven. Det betyr blant annet enerett til å gjøre verket tilgjengelig for allmennheten (Åndsverkloven. §2).
Oppgaver som er unntatt offentlighet eller taushetsbelagt/konfidensiell vil ikke bli publisert.

Vi gir herved Universitetet i Agder en vederlagsfri rett til å gjøre oppgaven tilgjengelig for elektronisk publisering:	Ja
Er oppgaven båndlagt (konfidensiell)?	Nei
Er oppgaven unntatt offentlighet?	Nei

Acknowledgments

This thesis is written as the final part of a two-year Master of Information Communication Technology program from the Department of ICT, Faculty of Engineering and Science, University of Agder, Grimstad.

Bitmesh AS created the basis of the thesis, set constraints, and provided resources and supervision. This thesis is the results of a close collaboration with and for Bitmesh AS.

We want to express our deepest gratitude to our advisors, Associate Professor Lei Jiao & Ph.D. Research Fellow Rebekka Olsson Omslandseter for their guidance and help. In addition to our advisors, we want to thank Arnt E. Berge and Tahani S. Berge, and the rest of the Bitmesh AS team for providing a challenging and comprehensive assignment and for lending us their time and resources.

We would also like to thank the University of Agder for providing an excellent learning environment and for providing us with tools that proved useful in the completion of this Master's Thesis.

Lastly, we would like to thank our families and friends for their unwavering support.

Daniel Nguyen Hansen
Aleksander Markus Lingstad
Marius Suvatne

Grimstad
June 3rd, 2022

for providing

Abstract

Clustering is an attempt to form groups of similar objects, and it is a powerful tool for discovering valuable underlying patterns in the data. When clustering on high dimensional data, the algorithms can suffer from the *curse of dimensionality*. This is a problem that occurs when data becomes sparse due to many dimensions, and can lead to poor clustering performance. Dimensionality reduction methods (DRMs) are thus designed to help alleviate this issue. For a time-series that is a temporal set of points, each consecutive point in time can be considered a dimension and therefore it belongs to high dimensional data. Time-Series K-Means (TSK-Means) with Dynamic Time Warping (DTW) is an algorithm that has been proven successful for clustering time-series. However, TSK-Means is computationally complex and might require substantial training time due to the potentially high dimensionality of time-series.

This thesis studies the clustering of time-series data, provided by temperature sensors installed in refrigerators, trying to make it less computationally complex by the use of the DRMs Principal Component Analysis (PCA), Time-Series Autoencoder (TSA), and Self-Organizing Maps (SOM). We utilize these methods in combination with three clustering algorithms, namely, K-Means, Density-Based Spatial Clustering of Applications with Noise (DBSCAN), and Agglomerative Hierarchical Clustering (AHC), to potentially find valuable patterns in the provided data. The clusters and patterns were evaluated on a theoretical and practical level regarding the application of pattern recognition and detection in the domain of refrigerator temperature monitoring and logging. This is an effort to improve refrigerator maintenance and quality assurance, deviation management, and to potentially reduce food loss.

The results indicate that TSK-Means outperforms any other combination of DRMs and clustering algorithms when it comes to detecting patterns in the data, despite being more computationally complex. Regardless, the use of DRMs simplified the clustering process of time-series, and allowed the K-Means algorithm to detect patterns more efficiently than the TSK-Means algorithm. The clusters and patterns that were discovered seem promising for the application of deviation management and refrigerator quality assurance.

Contents

List of Figures

List of Tables

Terminology

Measurement - A temperature value paired with a timestamp

Sample - Smaller collection of consecutive measurements from a signal

Signal - Stream of measurements from one temperature sensor

Acronyms

AE - Autoencoder

AHC - Agglomerative Hierarchical Clustering

AI - Artificial Intelligence

DBSCAN - Density-based Spatial Clustering of Applications with Noise

DRD - Dimensionality Reduced Data

DRM - Dimensionality Reduction Method

LSTM - Long Short-Term Memory

MKP - Manufactured Known Patterns

ML - Machine Learning

PCA - Principal Component Analysis

SOM - Self-Organizing Map

TSA - Time-Series Autoencoder

Table of Notations

Notation	Description
A and B	Time-Series
α	Input value
C	LSTM cell state
CH	Caliniski-Harabasz Index
D	Distance matrix
DB	Davies-Bouldin Index
d_0	Mean distance from a sample and all other points in the same cluster
d_1	Mean distance from a sample and all other points in the next nearest cluster
$\Delta(X_i, X_j)$	Distance between X_i and X_j
ϵ	Radius extending from a data point
ζ	Single cluster
η	Learning rate
f	LSTM forget gate output
θ	Cluster centroid
$h_{w(z),i}$	Scalar multiplier (neighborhood function)
i and j	Denotes index and iteration
K	Amount of clusters
L	Rank of matrix
Λ	Diagonal Matrix
M	Matrix
m	Dimensions/features
μ	SOM model
n	Amount of points/samples
R_{ij}	Similarity measure
r_i and r_w	Vectorial locations on display grid
S	Set of samples
$SC(d_0, d_1)$	Silhouette Coefficient for a single sample s
$\overline{SC}$	Silhouette Coefficient for a set of samples S
s	One single sample/point
σ	Width of neighborhood function
t	Time
V	Trace matrix of inter-cluster dispersion
W	Trace matrix of intra-cluster dispersion
w	SOM winner node
X	Dataset
Z	Set of clusters

Table 1: Table of notations.

Chapter 1

Introduction

Faulty refrigerators can have devastating effects on the food contained within. In order to handle food properly and safely, it is important to understand how fluctuating temperatures can affect the food quality. Traditionally, temperatures are manually measured daily, but over the last years, continuous digital monitoring of temperature has become more widespread [25]. This has opened up a world of possibilities for monitoring and anomaly detection.

Bitmesh AS is an IoT company specializing in deploying solutions related to temperature sensors and other sensors located all around Norway [4]. They intend to use this data to aid temperature control and anomaly management in businesses. Establishments that store food in temperature-controlled units are legislated by law to document temperatures and ensure that their products are safe for consumption [25]. Variations in these environments happen frequently. It is therefore desirable to identify patterns in order to evaluate the food quality and enhance routines preventing food loss. Abnormal patterns in temperature data are likely to suggest faulty thermostats, open doors, improper usage, food delivery, etc. Being able to detect and differentiate such behavior is beneficial for businesses to understand when and why their food has spoiled, and ideally help to prevent it in the first place.

Bitmesh AS provided a large dataset of temporal temperature data, in which we will attempt to discover and identify patterns. The provided data consists of continuous individual measurements from multiple sensors, and comes with no labels or known truths, making it challenging to identify and confirm potential patterns. Thorough organization, curation, and processing of data is crucial for the success of this thesis.

Briefly explained, we will explore pattern detection, also known as clustering, algorithms that can be applied to predict faulty refrigerators and automatic deviation reporting. We will, in addition, utilize methods for extracting important features in an attempt to reduce the computation, and evaluate if such methods can aid in this detection.

We will, prior to explaining our methodology, present theory and background related to established methods and algorithms. This includes the dimensionality reduction methods (DRMs) Principle Component Analysis (PCA), Time-Series Autoencoder (TSA), and Self-Organizing Maps (SOM), the clustering algorithms K-Means, Density-based Spatial Clustering of Applications with Noise (DBSCAN), Agglomerative Hierarchical Clustering (AHC), and Time-Series K-Means (TSK-Means), and the evaluation metric methods Silhouette Score, Davies-Bouldin Index, and Calinski-Harabasz Index. Furthermore, the implementation, application, and testing of the aforementioned algorithms will be presented. We will

discuss the process of implementation and the rationale behind these choices, before an evaluation and comparison of collected results are conducted. It is important that results are evaluated from multiple points of view to obtain the best possible impression. This is done via visual inspection with the supervision and opinion of a domain expert, combined with the different performance metrics provided.

1.1 Motivation

According to Bitmesh AS, classification of change in temperatures can be helpful in anomaly detection and management. As there are no known truths to the data at this point, traditional classification methods with known classes are not applicable. However, finding underlying patterns in the data is a step in the right direction for Bitmesh AS and their customers to understand more about temperature behavior and the equipment (e.g., refrigerators). Thus, the clustering provides a starting point for further analysis.

We want to research and explore methods for dimensionality reduction, due to the *curse of dimensionality*, see Section 2.4 for more information. Clustering high dimensional data is also computationally complex, and reducing dimensionality might prove helpful in optimization. For reduction, we want to explore methods that can extract information from the original data, such that it can be expressed with less dimensionality while maintaining its original characteristics, so-called feature extraction methods.

Successful classification of temperature patterns can prove beneficial, and has multiple applications. Classification can help predict faulty refrigerators before they break. In this way, preventative measures can be taken in order to reduce food waste. Pattern recognition and classification can also guide in evaluating the food quality due to temperature changes on a more precise level than traditional methods, which relies on one measurement alone. Different patterns can also be used to detect valuable refrigerator properties, such as energy consumption and insulation.

1.2 Thesis Definition

There are clustering algorithms that can be applied to high dimensional data, but these are complex and time-consuming. In addition, the phenomenon *curse of dimensionality* appears when clustering algorithms struggle to define patterns when clustering, due to the wide variety in the data leading to sparseness. Thus DRM are applied to overcome these problems. This thesis can be defined by the following research questions and hypotheses.

1.2.1 Research Questions

1. Can methods for extracting features from the data and reducing its dimensionality aid the clustering algorithms?

2. To what extent can the results be applied in a practical sense?

3. Can dimensionality reduction, and clustering algorithms outperform the state-of-the-art TSK-Means algorithm?

1.2.2 Hypotheses

1. Dimensionality reduction methods will aid clustering algorithms in detecting patterns in time-series data.
2. Patterns detected in this thesis can be applied to improve deviation management.
3. Clustering algorithms on the DRD will outperform state-of-the-art TSK-Means.

1.3 Objectives

The data should be well organized and curated in order to discover temperature time-series patterns. Once the data is prepared, it should then be used to explore solutions. Hence, the objectives of this thesis can be outlined as follows:

- Analyze the provided data to gain insight. Knowing how to collect and organize the data to promote a good foundation.
- Curate the data to remove unwanted samples and handle overlapping, missing, and duplicated measurements.
- Pre-process the data so that the DRMs can use it. The data should be in a format that facilitates simple adaptation for the other methods used while keeping most of the details of the original data.
- Research and implement DRMs.
- Research and implement clustering algorithms.
- Install temperature sensors in a controlled test environment to provide some example patterns.
- Evaluate the results from the proposed DRMs.
- Evaluate the results from the proposed clustering algorithms.

1.4 Outline

The thesis is organized as follows:

Chapter 2 will examine the background and theory of already established methods for dimensionality reduction and clustering, as well as methods for preparing the data before such methods are applied. We will also present the state-of-the-art for time-series clustering.

Chapter 3 presents the approaches and methods used in this thesis and an explanation of their implementation. It will clarify the reasoning behind decisions made in this thesis.

Chapter 4 will examine the experiments and results from the proposed methods. This chapter presents the observed patterns and an analysis of them.

Chapter 5 will discuss the work and the results of this thesis, and the ideas regarding future work, potential improvements, and dataset limitations.

Chapter 6 is the final chapter where the thesis is concluded.

Chapter 2

Background

In this chapter, we present the background and relevant theory to the work carried out in this thesis. This will provide a theoretical background for the methodology and the thesis results, as well as introduce the concept of time-series and approaches for applying this data to the domain. Lastly, theory related to methods of dimensionality reduction will be presented, before introducing relevant clustering algorithms and belonging theory.

2.1 Time-Series

In mathematics, time-series is defined as a temporal set of points. Commonly a value is paired with an independent point in time [7].

Time-series is generally divided into discrete and continuous time-series. Discrete time-series consists of points equally spaced and of a known length, while for continuous, the spacing and length may vary [32].

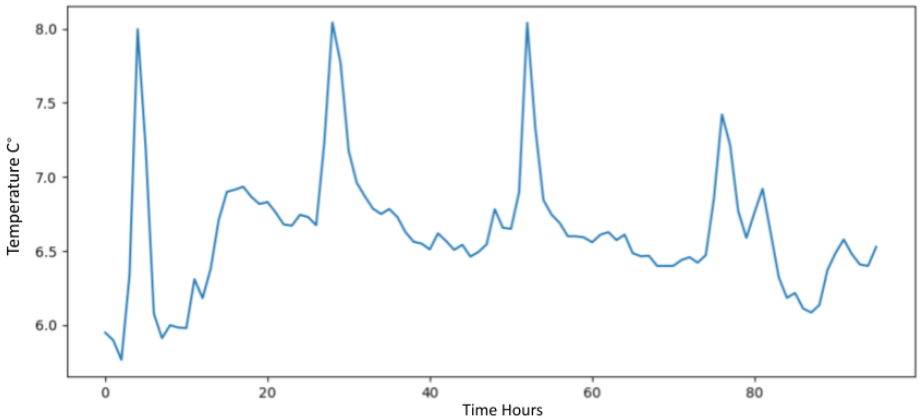


Figure 2.1: Example of a continuous time-series from data researched in this thesis.

Time-series provide a unique way of analyzing how values change over time, making it applicable for forecasting future values, tracking changes, and detecting patterns.

Time-series is considered n-dimensional, where n is the amount of points in time within the sample [18]. It is often simplified and presented as a graph, with two dimensions, time and value, see Figure 2.1.

2.1.1 Interpolation

Interpolation is a statistical method of extracting an estimated value given established values [17]. Broadly, it is a simple mathematical concept, where the general trend of the established values is used to estimate an unknown value.

Interpolation can be implemented in numerous ways, where the most common ones are piecewise, linear, or polynomial. Piecewise interpolation is the simplest, using the nearest known value as the estimated unknown value. For the linear approach, a straight line is made between two nearest established values, and the estimated value will be the crossing point of this line and the point of interest, as seen in Figure 3.6. The polynomial approach is based on *Weierstrass' approximation theorem*, saying that a polynomial $f(t)$ with n degrees can pass through a set of n points. The estimated value is the value for $f(t)$ at the desired value of t [8].

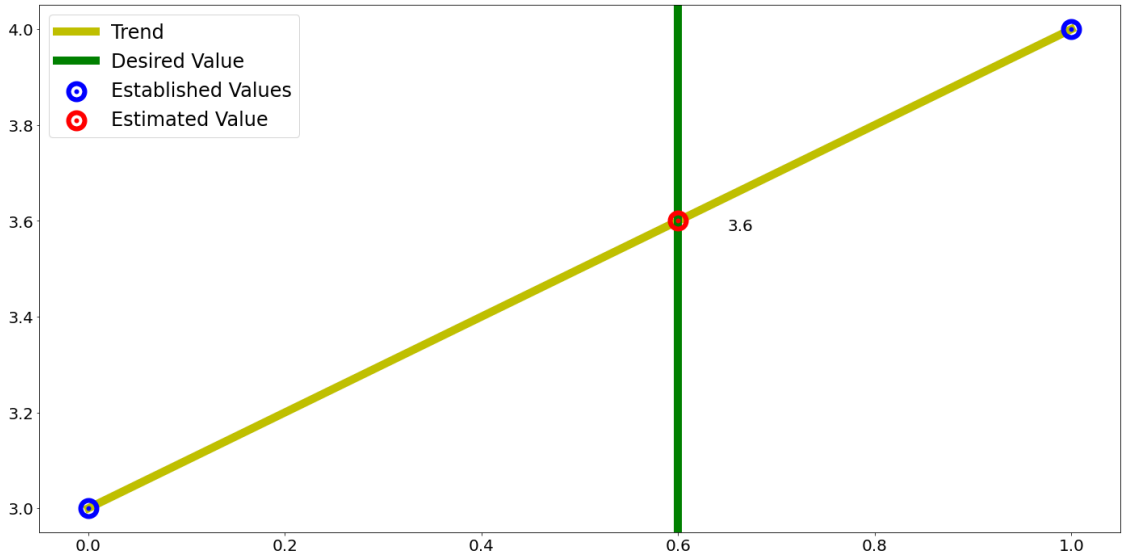


Figure 2.2: Illustration of a linear interpolation where y is estimated to be 3.6, when x is 0.6.

For time-series, interpolation can be a powerful tool to artificially distribute measurements uniformly. The estimated values from interpolation will capture the general trend, but can lose important details or add unwanted noise.

2.2 Normalization

Statistical data normalization is a pre-processing method of mapping values measured on different scales to a notionally common scale [31].

There are multiple ways of performing normalization depending on the wanted range of the resulting scale. Some common normalization methods are standard score, student's t-statistics, studentized residual, standardized moment, coefficient of variation, and min-max feature scaling [24].

The general formula for a min-max feature scaling, where the values are normalized within the range $[0, 1]$, is defined as follows,

$$S'(t) = \frac{S(t) - min(S)}{max(S) - min(S)}, \tag{2.1}$$

where S is a set of values, $S(t)$ is the original value and $S'(t)$ is the normalized value. Table 2.1 shows an example of normalization for a sample. The original sample contained values ranging from 2 to 32, while the normalized sample ranges from 0 to 1.

Before:	2	16	20	4	12	24	8	32
After:	0.00	0.47	0.60	0.07	0.33	0.73	0.20	1.00

Table 2.1: Normalization of data with range 2 to 32 to range 0 to 1.

2.3 Unsupervised Machine Learning

Unsupervised machine learning relates to the fact that a machine does not receive any supervised target outputs or feedback from its environment, and there is no human intervention. It might be difficult to imagine how and what a machine could possibly learn based off of this fact. A framework can, however, be developed based on the concept of a machine's goal to construct representations of the input space provided. This can further be used in decision-making and prediction of future inputs. Due to the fact that unsupervised learning does not receive any feedback from its environment, hidden patterns and groupings can be discovered in data that would be considered as unstructured noise [10] [15]. Generally, clustering methods such as K-Means, DBSCAN, and AHC are unsupervised forms of machine learning.

2.4 Dimensionality Reduction

Dimensionality reduction methods (DRMs) are methods applied to higher dimensional data in aims of de-noising and/or simplifying the data by reducing the amount of dimensions [28]. It is commonly applied before analyzing high dimensional data, and have proved to be beneficial in facing the *curse of dimensionality*. The *curse of dimensionality* is a term used for the problems occurring when analyzing data in higher dimensions. As the dimensionality of the data increases, the total volume of spaces increases, and the available data becomes sparse [45]. DRMs utilize a variety of methods for extracting what is considered important features of the given data.

2.4.1 Principal Component Analysis

Principal Component Analysis (PCA) is one method for dimensionality reduction which extract the most important features from the data [2]. These extracted features are called *principal components*. Principal components are linear combinations of the original data, in which the first component accounts for the largest variance of data [2]. The second component will account for the next most variance, under the constraint that it is uncorrelated to the first. Likewise the other components are calculated until there is an equal amount of

principal components as dimensions of the original data. Hence by removing the last principal components, which accounts for the least amount of variance in data, the dimensionality can be reduced, with minimal loss of data variation.

The data is comprised by n samples, and m features, creating an $n \times m$ matrix M. M has the *singular value decomposition* (SVD) $M = P\Lambda Q^T$. Where P is an $n \times L$ matrix, and Q is a $m \times L$, given L, the rank of the matrix M. Λ is the diagonal matrix of M [2]. In PCA, the principal components are obtained from the SVD to find the line F through the multidimensional data space M which minimizes the average distance between the line and data points [2]. F is denoted $F = P\Lambda$, using the equation from above gives,

$$F = P\Lambda = P\Lambda Q^T Q = MQ. \tag{2.2}$$

2.4.2 Autoencoder

Another method for reducing dimensionality of data is the application of an Autoencoder (AE), to extract features into a smaller latent-space. A latent-space is a compressed representation of data [42]. An AE is a type of unsupervised artificial neural network consisting of two subnetworks, an encoder and a decoder [27]. The encoder learns to generate a compressed representation of the input, while the decoder learns how to regenerate the input from this compressed latent-space. The algorithm is unsupervised, and applies backpropagation to reduce the loss between the input and output [27].

Figure 2.3 shows the basic structure of an AE. It consists of fully connected layers with a *bottle-neck* in the middle. This is the latent-space, also called the code. Commonly AEs are designed to be symmetrical on either side of the latent-space, but variations exist.

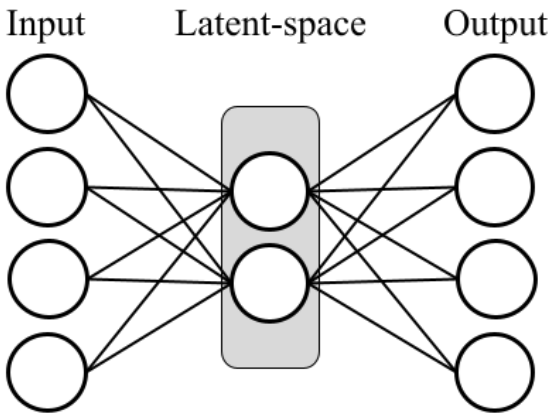


Figure 2.3: Simplification of the AE network structure.

AE trains on a given domain, hence, it learns to extract important features for the data within that domain. AE is, therefore, not a general compression method, but domain specific compression method [27]. It is considered to be *lossy*, meaning some details will be lost under compression. However, if implemented and trained properly, the latent-space can be a compressed, yet accurate, representation of the input data. Therefore the latent-space can be a useful DRM.

LSTM

Long Short-Term Memory (LSTM) is a variant of *Recurrent Neural Networks*, a sub-category of Neural Networks specifically designed for time-series data [14]. The idea behind the LSTM is to let every step extract important information for a larger collection of information while keeping the short-term relation between consecutive inputs [30].

The core concept of the LSTM lies within its cells [30]. Through three different gates, these cells process the information provided. The first gate is the *forget gate*, which decides what information is important or can be forgotten, giving the output f [30]. Next the *input gate* calculates the current cell state C_1. The current cell state is calculated as following,

$$C_1 = f \times C_0 + \alpha \times \hat{C}, \tag{2.3}$$

where C_0 is the previous cell state, α is the current input, and $\hat{C}$ is the new candidate values to be stored [30]. The final gate is the *output gate* which calculates the next hidden state, which contains relevant information from previous inputs [30].

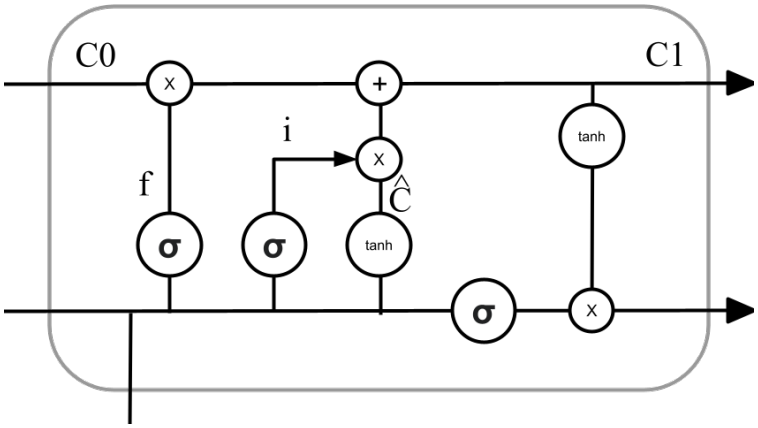


Figure 2.4: Illustration of an LSTM cell.

Section 2.4.2 illustrates how the current cell state is calculated using multiple gates and established values. LSTM is a powerful algorithm as it theoretically can carry relevant information from the entire data sample while processing it.

2.4.3 Self-Organizing Maps

A Self-Organizing Map (SOM) is an unsupervised neural network to visualize high-dimensional data for dimensionality reduction. The goal is to go from a high-dimensional representation to a low-dimensional (commonly two-dimensional) discretized representation of the input space, called a map [20] [33]. This DRM preserves topological properties of the input space through competitive learning [26].

The difference between SOMs and other neural networks is that a SOM applies competitive learning instead of error-correction learning. The most important topological properties of the input space are preserved by using a Gaussian neighborhood function [20] [33].

Mapping starts when weight vectors are initialized, and a sample vector is randomly selected. The map of these weight vectors is searched to find a weight representing the sample in the best way possible. In other words, training data is distributed in a data space, and SOM

nodes are arbitrarily positioned in the same space, before being pulled towards a fixed point in the training data, to match the training data. This can be seen in Figure 2.5. Each of these vectors has neighboring weights close to it. The chosen weight (winning node), along with neighboring weights are rewarded by being able to become more like that randomly selected sample vector. The closer the weights are to the winner, the more their weights will be altered. The further away, the less it learns. Repeating these steps will result in a decrease in the number of neighbors over time which enables the map to grow and form different shapes [20] [33].

The winning node, also called *best matching unit* is a technique for calculating the distance from each weight to the sample vector by running through all weight vectors. The winner is the weight with the shortest distance. The most commonly used method is the Euclidean Distance [20] [33].

The SOM computes the models and automatically organizes them into a meaningful representation to optimally describe the domain observations. In this representation, similar models in the grid are closer together than dissimilar ones. SOM can be viewed as a similarity graph, where the computation is a recursive regression process.

Regression of the set of model vectors are performed using the following equation,

$$\mu_i(t+1) = \mu_i(t) + h_{w(z),i}(z(t) - \mu_i(t)), \tag{2.4}$$

t represents the index of regression steps, the regression is performed for each presentation of z recursively, denoted $z(t)$. $h_{w(z),i}$ is the scalar multiplier, also known as the *neighborhood function* and acts as the smoothing kernel over the grid. The $w(z)$ subscript inside the neighborhood function is defined by a given condition,

$$\forall i, ||z(t) - \mu_w(t)|| \leq ||z(t) - \mu_i(t)||, \tag{2.5}$$

where $\mu_w(t)$ is the model that has the best match with $z(t)$, also known as the winner. Euclidean Distance is the most common comparison metric, and it is usually chosen for SOM. In the event of samples $z(t)$ being stochastic and having a continuous density function, the probability of multiple minima occurring is zero. In contrast, if the samples are of discrete values, minimums may occur. In the latter case, one of them should be randomly selected as the winner.

The scalar multiplier is given by,

$$h_{w(z),i} = \eta(t) \exp\left(-\frac{||r_i - r_w||^2}{2\sigma^2(t)}\right), \tag{2.6}$$

in which η denotes the learning rate factor, and $\sigma(t)$ is the width of the neighborhood function, both of which decrease for each regression step. r_i and r_w represent the vectorial locations on the display grid [20].

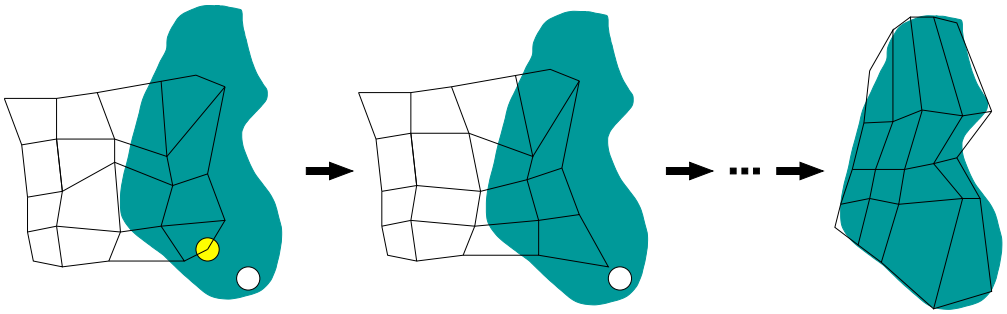


Figure 2.5: Illustration of SOM training. Blue blob represents training data and the yellow circle, the node nearest to the white fixed point in the data.

2.5 Clustering

In data science, clustering is a useful tool for finding structural groups in data, called clusters. Clusters are characterized by having similarity within one cluster, and dissimilarity between different clusters [37] [1].

Commonly, clustering algorithms are unsupervised machine learning algorithms using the distance between objects, the density of data, or statistical distributions to form clusters [1].

The primary goal of clustering is to explore and discover natural groupings of objects. There are several clustering methods, and convex clustering is one of them. In convex clustering, objects are said to belong to a cluster if they share a common cluster centroid. Density-based, distribution-based, and connectivity-based clustering are other commonly used clustering methods [16]. These algorithms go by many names depending on the domain of use and who is using them.

2.5.1 K-Means

One of the most commonly used and researched clustering algorithms is the convex clustering algorithm, K-Means [37]. Multiple adaptations to this algorithm have been proposed, but the fundamental mechanics remain. K-Means aims to form K clusters by assigning an observation to the nearest cluster mean, hence the name. The mean is represented as a cluster centroid. This algorithm updates and improves the cluster centroids iteratively until it converges to a solution. The K-Means algorithm works as follows:

1. Specify the number of clusters K.
2. Randomly select K data points to act as initial cluster centroids.
3. Assign data points to a cluster based on the closest cluster centroid.
4. Calculate a new cluster centroid for each cluster as the average of all data points within one cluster.
5. Repeat step 3. and 4. until the cluster centroids converge on one point.

Calculating the new cluster centroid is done with the following equation,

$$\theta_\zeta = \frac{1}{n_\zeta} \sum_{j=1}^{n_\zeta} s_j, \tag{2.7}$$

where n_ζ is the amount of data points s_j in cluster ζ.

Once the K-Means algorithm is complete, it will have formed clusters such that the sum of the squared distance between a data point and its assigned cluster centroid is minimized. The sum of the squared distances is calculated accordingly,

$$J(V) = \sum_{\zeta=1}^{Z} \sum_{j=1}^{n_\zeta} (||s_\zeta - \theta_j||)^2. \tag{2.8}$$

Where Z is number of cluster centroids, and n_ζ is the data points in cluster ζ.

One drawback worth mentioning, is that K must be specified. This can lead to sub-optimal assigning of clusters if K is either too small or large for the dataset [11]. In Figure 2.6 the chosen K is not appropriate for the data, leading to merging two different clusters into one, or splitting of clusters.

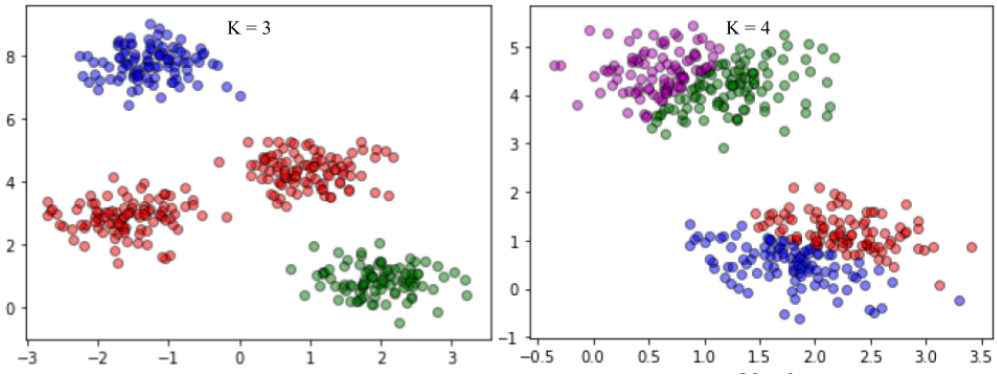


Figure 2.6: Two examples of when K is not appropriate.

This algorithm is considered to have an average runtime complexity of $O(tkn)$, where t, n and k is the iterations, number of clusters and number of data samples respectively [21].

2.5.2 K-Means for Time-Series

An adaptation for using K-means on time-series has been implemented, called *Time-Series K-Means* (TSK-Means). This algorithm forms clusters as *barycenters*, which is the mean sequence of multiple time-series [3], as shown in Figure 2.7. The TSK-Means algorithm is considered to be $O(tknm)$ runtime complex, where t is the iterations required to converge, k is the number of clusters, n is the number of samples to cluster, and m is the number of dimensions in the data [13].

Traditionally in TSK-Means, different time-series are compared at each instance of time using euclidean matching. While being useful in some domains, it has proven to be sub-optimal for others, especially dynamic data.

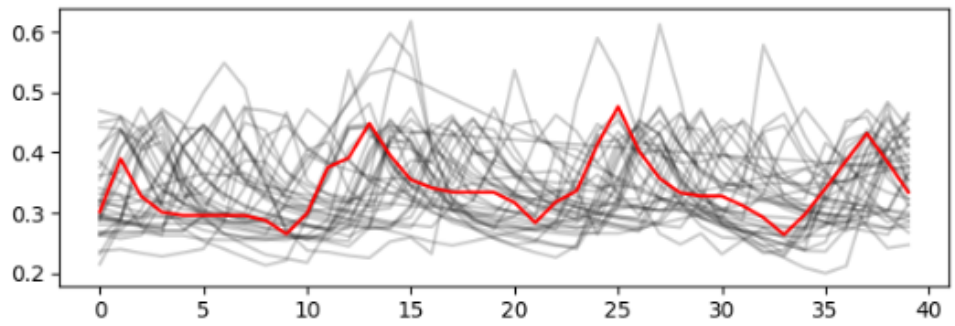


Figure 2.7: The red line represents the barycenter of the cluster, and an example of the samples within a cluster.

Dynamic Time Warping (DTW) is an algorithm commonly used to evaluate similarity in time-series [36]. The algorithm accounts for shift and distortion of the time in a time-series, hence useful for finding patterns in natural time-series data. The algorithm starts with a local distance matrix $D \in \mathbf{R}^{I,J} : c_{ij} = ||a_i - b_j||, i \in [1:I], j \in [1:J]$, containing a pairwise distance between each data point in time-series A and B. The algorithm finds the *alignment path*, which is the lowest cost path through D, as shown in Figure 2.8 [36].

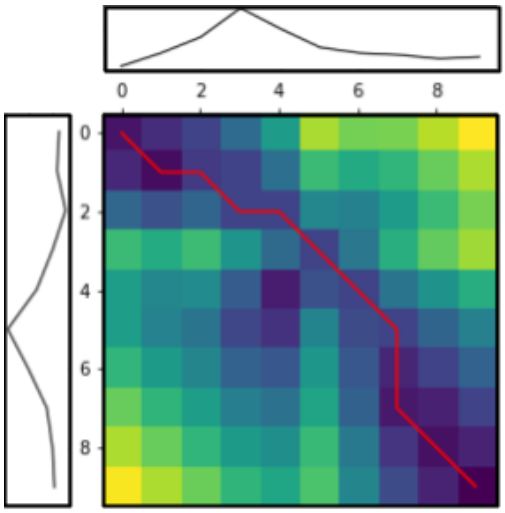


Figure 2.8: DTW finds the lowest cost through the distance matrix.

Figure 2.9 illustrates how two time-series can be compared with euclidean or DTW. DTW provides a natural comparison, and better results in some domains. Combined with DTW, TSK-Means can be a powerful method for analyzing time-series [36].

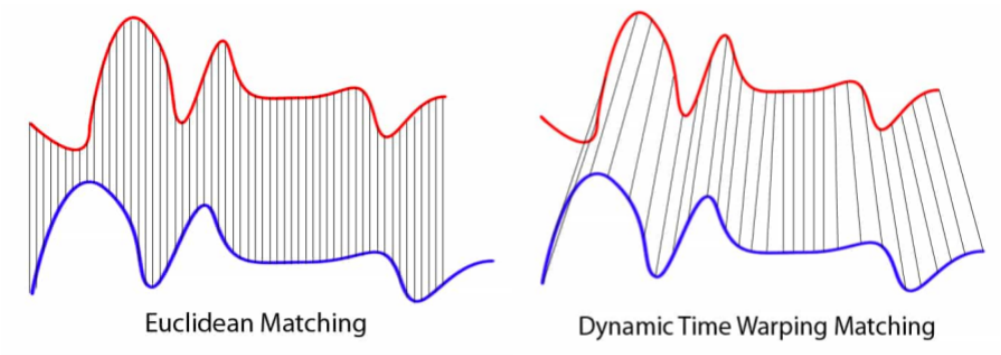


Figure 2.9: Example of how time-series can be compared with euclidean matching or DTW [44].

2.5.3 Elbow Method

Finding the optimal K for a given dataset can be complicated. Intuitively, more clusters will increase performance, but the problem is to find an optimal point where performance is high, and K is small. *Elbow method* is one proposed solution to this problem. This is a heuristic method aiming to find the optimal K while keeping K small [5].

The K-Means algorithm is tested for an increasing K, and evaluated using a relevant metric, e.g., Silhouette Score, see Section 2.6.1. Then the scores are graphed. The *elbow* or *knee* of the graph's curve indicate where an additional cluster will no longer increase the performance at a rate worth the additional cost [5]. This is considered the optimal K for the given dataset.

2.5.4 Density-Based Spatial Clustering of Applications with Noise

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is the first proposed clustering algorithm using the density of data points to perform clustering [19]. It is designed to form clusters with arbitrary shapes in the presence of noise in the data as seen in Figure 2.10.

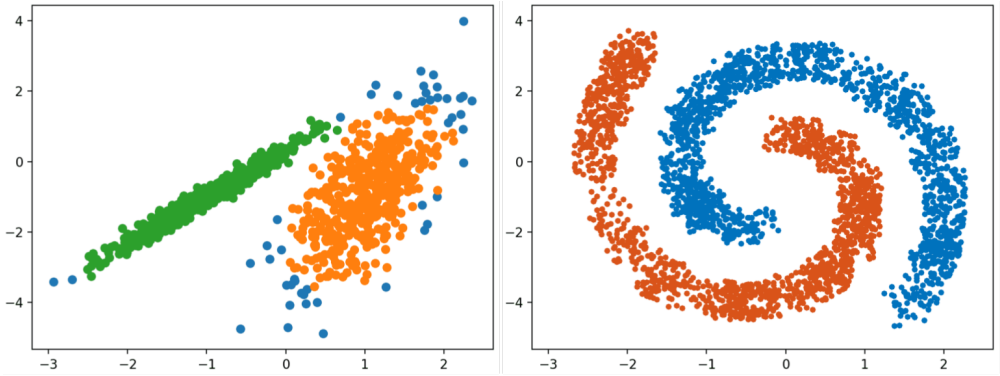


Figure 2.10: An example of a DBSCAN forming arbitrary shaped clusters [6].

DBSCAN treats data points as three different types of points, based on the amount, *min_samples*, of nearby neighbors. A neighbor is a data point within a given radius ϵ. *Core points* are data points with a defined amount of neighbors. *Border points* are data points that are neighbors with core points, but do not have enough neighbors to be considered a core point. *Noise points* are outliers, being data points that do not have any core points as neighbors. Knowing that the algorithm works as follows:

1. Define parameters radius ϵ and min_samples.
2. Define core points, border points, and noise points.
3. Connect all neighboring core points to clusters.
4. Assign all border points to the cluster of the closest core point.

DBSCAN relies on the density of data points for clustering, leading to some complications. The algorithm will not find the optimal clusters without a suitable ϵ and appropriate min_samples. The data could also vary its density, with sections of the data space being

sparser or denser than other sections. DBSCAN will not be able to take the difference in density within different clusters into an account [19].

The average runtime complexity of DBSCAN is $O(n \times log(n))$, where n is the number of samples [9].

2.5.5 Agglomerative Hierarchical Clustering

Agglomerative Hierarchical clustering (AHC) is a method of constructing a binary merge tree by merging two and two of the closest data points [29]. The algorithm treats every data point in the dataset X as a sub-cluster, and iteratively merges two sub-clusters until it reaches the root node, a cluster containing all other sub-clusters. This method is also referred to as *agglomerative hierarchical clustering* since it starts with each data point as a sub-cluster, and ends with one root cluster containing all data points [29]. This method can also be reversed, which is more complicated and is called *divisive hierarchical clustering*.

The distance between two sub-clusters, ζ_i and ζ_j, is denoted by $\Delta(\zeta_i, \zeta_j)$, and called the *linkage distance*.

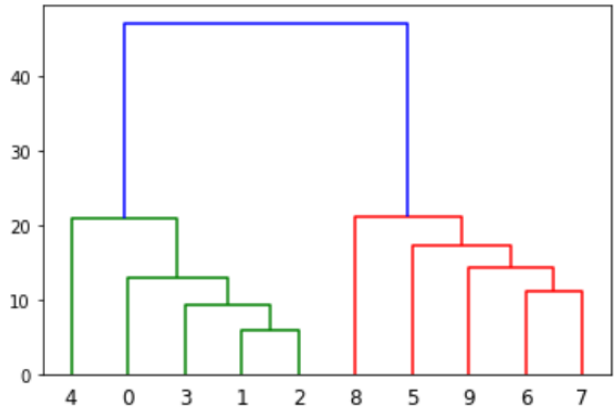


Figure 2.11: An example of a AHC dendrogram.

The binary merge tree can be represented graphically, called a dendrogram. The y-axis represents distance, and x-axis represents sub-clusters. Here, the connection between two sub-clusters indicates the shortest distance between any two sub-clusters and the distance between them.

One of the largest advantages of hierarchical clustering is that once the binary merge tree is complete, it can be cut at any given level, creating any desired amount of clusters [29].

For AHC methods, a runtime complexity of $O(n^3)$ where n is the number of samples [12].

2.6 Performance Criteria

A clustering algorithm is a form of unsupervised learning, which means that the data is not labeled in advance. Based on this, the outcome is uncertain, and may, in return, result in the discovery of hidden patterns or data groupings, since human intervention is absent [15].

Evaluating the clustering algorithms in terms of performance is not as straightforward as summing up errors, accuracy, loss, precision, or recall, as is the case for supervised classification. If a clustering algorithm can define separations of the data that somewhat corresponds to a set of ground truths, or can satisfy some assumptions such that members belonging to one particular class are similar than members of other classes, any evaluation metric should account for this, instead of taking into account the absolute values of cluster labels [35].

It is important to keep in mind that metric scores should not be trusted blindly. The calculation is data-dependent and there is a chance of incorrect understanding of output values. Performance score should only serve as an indicator, and it is up to the human eye to analyze and determine further. Outliers can lead to improper clustering, which would, in turn, affect how well an algorithm would perform. In that case, each cluster would have to be inspected to see how data was organized into clusters and how many members that cluster contained.

2.6.1 Silhouette Coefficient

The Silhouette Coefficient is one evaluation metric for evaluating in which ground truth labels are unknown. A higher score means a model with better defined clusters. The Silhouette Coefficient for a single sample is calculated as following,

$$SC(d_0, d_1) = \frac{d_1 - d_0}{\max(d_0, d_1)}. \tag{2.9}$$

The Silhouette Coefficient for a set of samples n is calculated as the mean of each Coefficient sample like,

$$\overline{SC} = \frac{1}{n} \sum_{i=1}^{n} SC(d_{0i}, d_{1i}), \tag{2.10}$$

where d_0 is the mean distance between a sample and the rest of the points in the same class, and d_1 is the mean distance between a sample and the rest of the points in the next nearest cluster. The advantage of using this method is that the score is between -1 and 1, where -1 means incorrect clustering, and 1 means highly dense and well separated clustering. A score around 0 indicate overlapping clusters. The drawback is that the score is generally higher for convex clusters than other types of clusters [35] [34].

2.6.2 Calinski-Harabasz Index

Calinski-Harabasz Index is the second metric that can be used for clustering where ground truth labels are unknown. The higher the index, the denser and well separated the clusters. This index indicates how similar an object is to its cluster compared to other clusters. The main advantage of this method is that it is fast to compute, and the drawback is that the score is generally higher for convex clusters than for other clusters.

Calinski-Harabasz score comprises of a more comprehensive calculation than for Silhouette Coefficient. Score CH is calculated as the ratio of inter-cluster dispersion mean and the

intra-cluster dispersion for a dataset X of size n clustered into Z clusters,

$$CH = \frac{tr(V)}{tr(W)} \times \frac{n-Z}{Z-1}, \tag{2.11}$$

where $tr(V)$ and $tr(W)$ is the trace of matrix V (inter-cluster dispersion), and W (intra-cluster dispersion), respectively. V and W is calculated as follows,

$$V = \sum_{\zeta=1}^{Z} n_\zeta (\theta_\zeta - \theta_X)(\theta_\zeta - \theta_X)^T, \tag{2.12}$$

$$W = \sum_{\zeta=1}^{Z} \sum_{s \in S_\zeta} (s - \theta_\zeta)(s - \theta_\zeta)^T, \tag{2.13}$$

where S_ζ represents the set of points in cluster ζ, θ_ζ and θ_X represents the center of ζ and X respectively, and n_ζ is the amount of points in ζ [35]. A larger V means higher inter-cluster dispersion, and a smaller W means that the relationship is close in that particular cluster. In other words, the higher the ratio, the better the effect of clustering, given that V is higher and W is lower [43].

2.6.3 Davies-Bouldin Index

The Davies-Bouldin Index is the third, and last, metric applicable to clustering where ground truth labels are unknown. The lower the index, the better the separation between the clusters. This index yields an average inter-cluster similarity, which means that it is a measure that compares the distance between clusters that are of the same size. A score of zero is the lowest possible score, while values close to zero mean better partitioning. The advantages are that computation is simpler than for Silhouette Coefficient, and the index is based on features that are inherent to the dataset since the computation only uses point-wise distances. The drawbacks are that the score is generally higher for convex clusters than for other types of clusters. Additionally, the distance metric to euclidean space is limited due to the use of centroid distance.

Davies-Bouldin Index is calculated as the average similarity between each cluster ζ_i and that cluster's most similar one ζ_j where $i = 1, ..., K$ and K is the amount of clusters. Similarity is a measure called R_{ij} that accommodates d_i which is cluster i's diameter (average distance between every point and its centroid), and d_{ij} which represents the distance between cluster centroids i and j [35]. R_{ij} and Davies-Bouldin Index are calculated respectively,

$$R_{ij} = \frac{d_i + d_j}{d_{ij}}, \tag{2.14}$$

$$DB = \frac{1}{K} \sum_{i=1}^{K} \max_{i \neq j} R_{ij}. \tag{2.15}$$

Chapter 3

Clustering and Detecting Patterns in Provided Sensor Data

In this thesis, we aimed to implement and test multiple methods to discover patterns in temporal temperature data. This was achieved by initially using three methods for extracting important information from data to reduce the dimensionality. The DRMs used in this thesis were PCA, TSA, and SOM. Subsequently, the compressed data was used to conduct clustering. Three different clustering algorithms were utilized to look for patterns, K-Means, DBSCAN, and AHC. All clustering algorithms were applied for the three DRDs and later compared. Results were also compared to the TSK-Means algorithm to investigate if dimensionality reduction would prove helpful given a high dimensional dataset. Three different performance scores Silhouette coefficient, Calinski-Harabasz index, and Davies-Bouldin index were applied to evaluate and compare results on a metric level. Additionally, the results were evaluated on a practical level, and to which degree the results could be applied within the domain.

3.1 The Dataset and its Characteristics

The data used in this thesis was provided by Bitmesh AS, and is normally used to supervise temperatures in commercial refrigerators in restaurants and stores. All signals used in this thesis can be seen in Appendix A.

A comprehensive analysis was performed to understand the data better. This included inspecting, cleaning, modeling, and other ways of gaining a better understanding of the domain. The dataset consisted of signals from temperature sensors distributed across 40 refrigerators, where each sensor measured and transmitted approximately every 15 minutes. However, the time between measurements can vary substantially. The temperature was measured in degrees Celsius and paired with a timestamp of when each measurement was conducted. Hence the dataset could be considered to consist of continuous time-series.

3.1.1 Temperature Sensors

The signals used for this thesis were collected from temperature sensors by Disruptive Technologies ASA. These sensors are wireless and marketed as the world's smallest sensors, with

a 19mm x 19mm footprint [40]. This allowed the sensor to be placed inside already existing refrigerators and freezers using an adhesive. The sensors had a temperature range from -40°C to 80 °C, a resolution of 0.05°C, and a maximum error of 0.4°C. Under the right conditions, a sensor is expected to have a lifespan of up to 15 years [40].

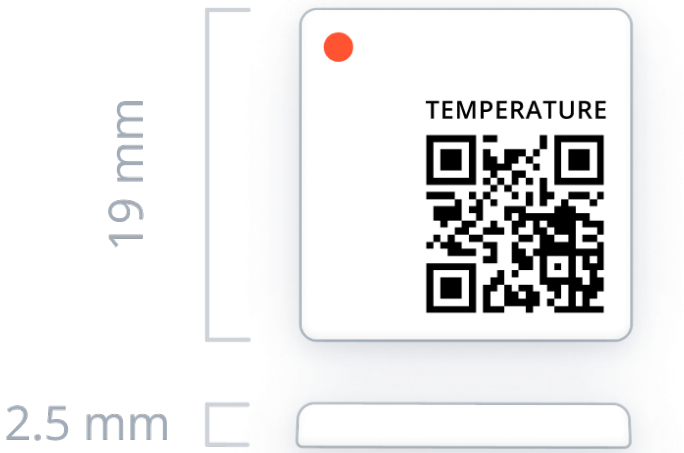


Figure 3.1: The data used in this thesis is collected using this type of sensor [39].

3.1.2 Accessing the Data

The data was accessed via a REST API to the Bitmesh AS signal API server. The signals had to be individually queried. There was no information regarding what type of unit the signal came from, freezer or refrigerator. Therefore, we manually had to investigate every signal. This was done using an internal tool from Bitmesh AS. Once the desired signal was identified, it was used to fetch the data from the API. The data contained a continuous stream of measurement events with the format of each event being observable in Table 3.1.

```
[{"id":"03792d3c-7c63-4df3-a7a9-776b46388405",
"sourceId":"dt/c43qdu1m4b6ce48o0mkg/devices/bjijoqe7gpvg00cjq1d0",
"timestamp":"2021-08-02T15:01:41Z",
"type":"VEvent",
"value":5.95}]
```

Table 3.1: Example of raw data from the API.

```
timestamp,value
2021-08-02T15:01:41,5.95
2021-08-02T15:16:27,5.9
2021-08-02T15:31:12,5.75
2021-08-02T15:45:58,6.25
```

Table 3.2: Example of data extracted from the API in CSV format where timestamp is datetime and value is in Celsius.

The tables Table 3.2 and Table 3.3 are examples of the data after processing the individual signal's API fetch. Here shown in CSV format, although the JSON format was also used in the implementation. Note that in Table 3.3, the data has been interpolated and normalized.

```
timestamp,value
2021-08-02T15:01:41,0.65421777191776
2021-08-02T15:16:41,0.6867227695362204
2021-08-02T15:31:41,0.7390646424197256
2021-08-02T15:46:41,0.7131721000763069
```

Table 3.3: Example of data after processing in CSV format where the timestamp is datetime and value is normalized temperature in Celsius.

3.1.3 Dataset Statistics

Key temperature statistics from the dataset are presented in Table 3.4. The metadata or data statistics provided a powerful insight into the expected behavior of the data.

Data Statistics	
Number of Measurements	1 076 243
Mean Temperature	4.61
Median Temperature	3.60
Maximum Temperature	28.40
Minimum Temperature	-14.30
Standard Deviation	3.80
Variance	14.46

Table 3.4: Table of key temperature statistics of the dataset.

The time-series is generally cyclic, with defrosting intervals, as seen in Figure 3.2.

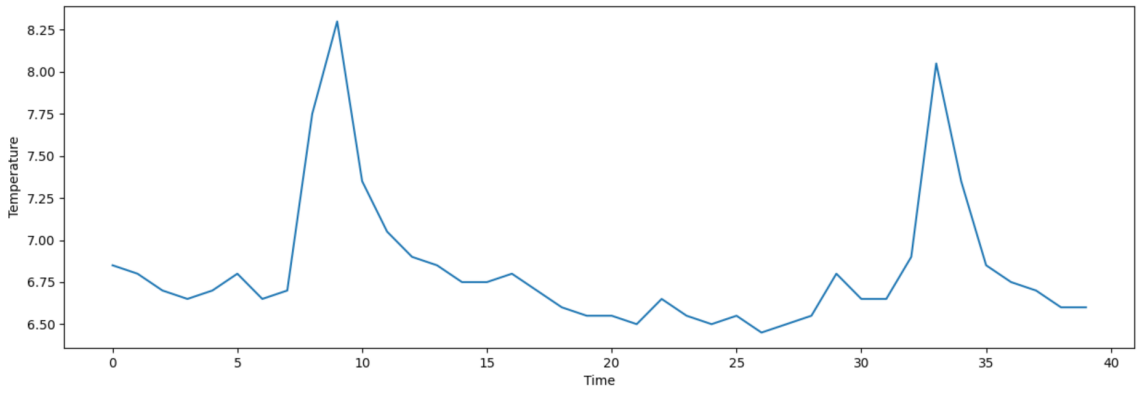


Figure 3.2: Visual example of the time-series data from the sensors.

As mentioned, the time between measurements vary. This is due to a natural transmission variation of the sensors, but can also be due to transmission errors, lost connections, frozen sensors, etc. Dataset analysis calculated the mean time between two consecutive measurements to be 932.298 seconds, or 15.5 minutes. As seen in Figure 3.3, the time between measurements is concentrated at 900 seconds, or 15 minutes. It is important to understand the effects of the varying time gaps to handle them appropriately.

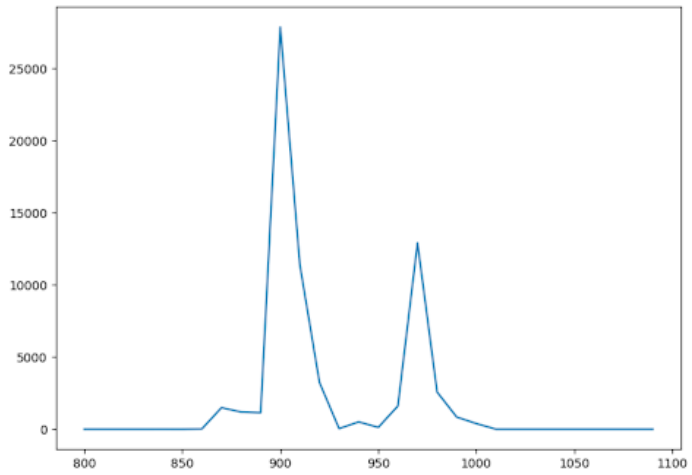


Figure 3.3: Distribution of time between measurements in seconds.

3.1.4 Challenges with the Provided Dataset

The provided dataset brought multiple complications. The first, and biggest problem regarding classification and pattern detection, is that there are no known truths in the dataset. Hence forming models to recognize known patterns and pattern classification is impossible. It also makes it hard to verify results in the end.

There is no known seasonality for the signals, except frequent defrosting, however, defrosting intervals is signal-dependent and not always present. Therefore we found no obvious sample size. A sample size too small could have had problems capturing larger patterns, while a sample size too large could have had multiple patterns within.

The signals are continuous time-series, with variation in the intervals between measurements. Commonly, the interval between each sample was approximately 15 minutes. However, some measurements had shorter and longer intervals as mentioned. One signal had an interval of 21 months, see Figure 3.4. This is due to the sensor or refrigerator being temporarily disabled.

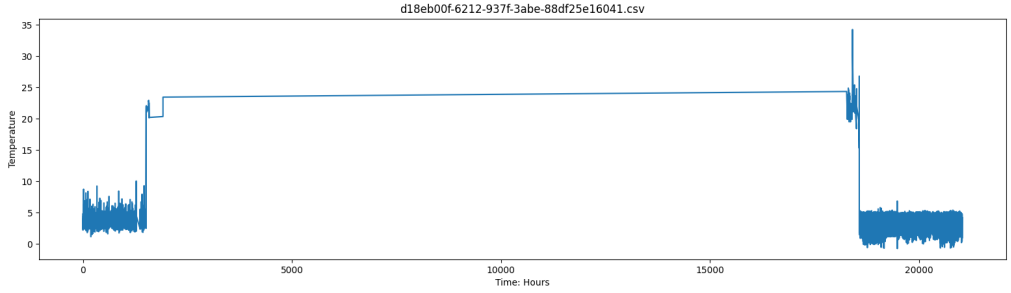


Figure 3.4: The graph from the signal is missing 21 months of data.

For an unknown reason, signals occasionally contained duplicate measurements. Here both timestamp and value were entered twice. One signal had over 4000 duplicated measurements. All duplicated measurements were identified and removed.

Some signals contained measurements from two different units. Figure 3.5 show how mea-

surements from a refrigerator and a freezer are intertwined as a single signal. This is troublesome for this thesis because only refrigerator temperatures are applicable, and intersecting temperatures from different refrigerators will create misleading patterns. We manually had to look through the signals for overlapping measurements and remove them manually.

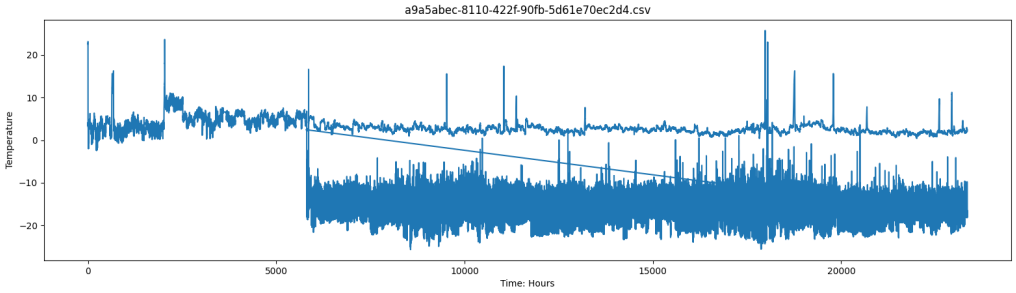


Figure 3.5: Overlapping measurements from one refrigerator (top) and one freezer (bottom) in one signal.

There was no information regarding the refrigerators themselves, e.g., volume, effect, sensor position, etc. We assumed that these factors would affect the detectable patterns. Larger refrigerators require more power to cool down, while they can maintain the temperature for longer in case the door remains open or the refrigerator is turned off. Different insulation properties for refrigerators will affect how much energy and work is required to maintain the proper temperature. A sensor placed near the refrigerator door was more sensitive to temperature change due to opening the door, than a sensor placed deep inside the refrigerator. However, due to the missing information, it was omitted from this thesis.

3.1.5 Interpolation of Data

In order to perform dimensionality reduction, some requirements needed to be met. One was that each data sample must have had the same format and length. Second, all data measurements must be uniformly distributed over time within the sample itself. Due to variation in time between measurements, the latter condition could not be met, and would therefore pose as an extra complication in the chain of dimensionality reduction. By using interpolation was it possible to generate new estimated measurements at a desired point in time to bypass this problem. Piecewise interpolation would impair the relational time interval between two consecutive measurements by shifting established values in the time domain. The polynomial approach would likely produce values far outside the expected temperature range given the potential polynomial of high degree needed, and could include destructive noise. Hence each signal was interpolated at intervals of 15 minutes using linear interpolation, providing a set of uniformly distributed measurements. Given the interval size, 40 consecutive interpolated measurements would then contain ten hours of data, which was the chosen sample size for this thesis. Interpolation resulted in more uniform time sequence, rendering timestamps for each measurement negligible. The samples are simplified to contain only 40 temperature values, and a separate list of metadata for each sample was stored. This enables lookup of time and sensor for respective samples which can be useful for any later practical evaluation.

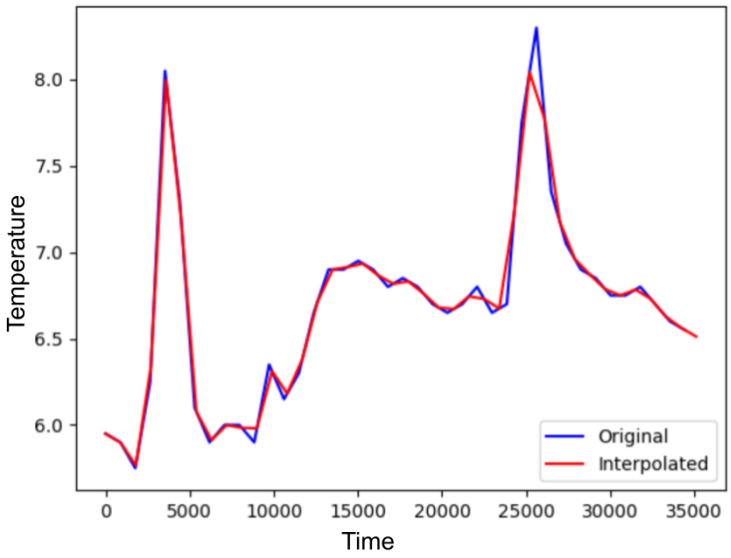


Figure 3.6: The graphs shows the interpolated data overlapped on the original data.

Through interpolation, all samples are formatted while capturing temperature behavior. Some details were lost as a product of interpolation, as seen in Figure 3.6. It remains uncertain if these details were important in locating patterns in the data, or if the general behavior is sufficient. We were aware that interpolation would influence the results, but simplification was necessary due to the complexity related to multiple dimensionality reduction and clustering algorithms.

3.1.6 Overlapping Samples

Samples overlap such that one value of one sample, will also be in another sample, see Figure 3.7. Overlapping allows each measurement to be used multiple times, providing more data for training.

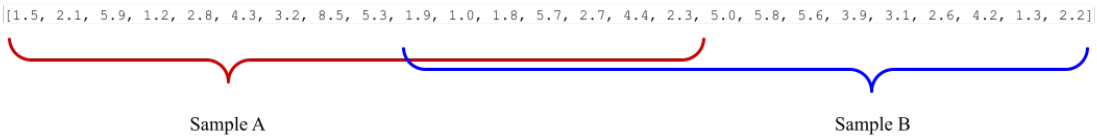


Figure 3.7: Example of overlapping samples.

Neglecting overlapping data could result in a loss of patterns in the data that would exceed ten hours. Another rationale for having overlapping data, was that otherwise, the selection of sample size could promote artificial patterns, which was undesirable. A pattern smaller than ten hours, such as defrosting, could appear multiple times in one sample at different points. Through overlapping, we ensure that these defrosting patterns will appear at all points in time.

By using overlapping data, it was possible to analyze the same scenario at multiple points in time. We assumed that this would provide a better perspective and aid the clustering algorithms in finding real-life, relevant patterns.

Although overlapping was utilized to discover large patterns, it was possible that the exact same pattern could end up in different clusters when analyzed from different points in time.

E.g., in one sample, we observed the entire pattern, while for another sample, we could only observe the beginning.

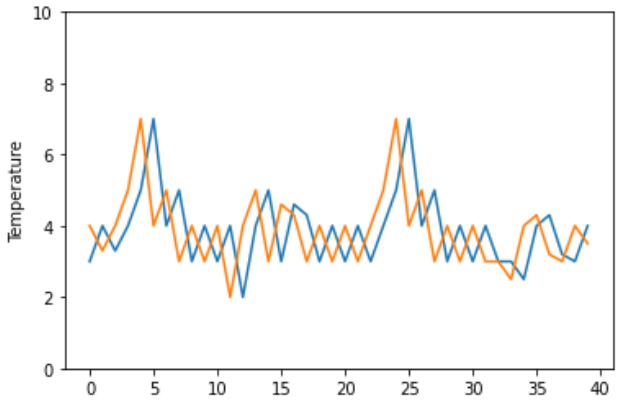


Figure 3.8: Graphical example of overlapping samples with minimal time shift.

By overlapping the samples, we increased the dataset, and removed undesired enforced patterns due to improper sample selection. Overlapping samples had uncertain effects on the results, but we expected that it helped in pattern detection, and seeing patterns from multiple points in time.

3.1.7 Data Curation

The next step in data analysis was curating the data and removing uninteresting samples. If a refrigerator remains shut off over a longer period of time, the temperature will eventually stabilize around room temperature. This was considered irrelevant to this thesis. However, it can be interesting to find scenarios in which a refrigerator has been recently turned off, to investigate how temperature may change. Figure 3.9 shows a sample taken from a permanently turned off refrigerator, and one sample from a refrigerator recently turned off.

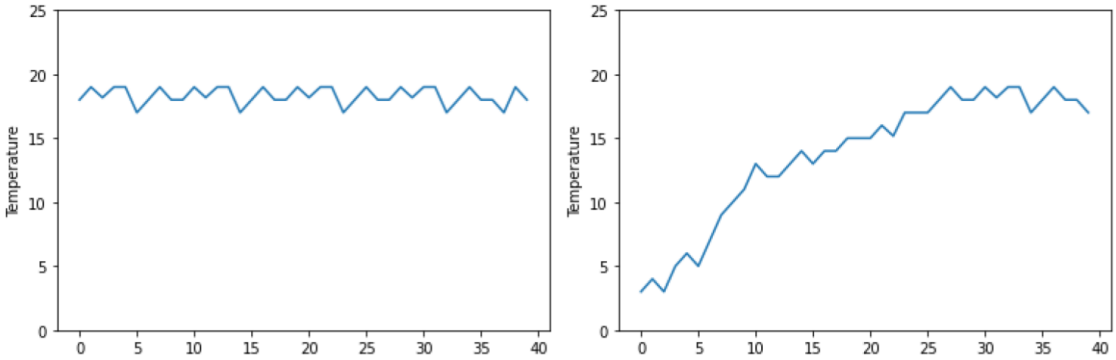


Figure 3.9: The sample on the left was removed because it exceeded the mean limit, while the sample on the right was kept.

To identify samples regarded as uninteresting, we removed all samples where the *sample mean* was three times the *signal mean*. A total of 1 216 394 samples were created, where 49 648 samples were removed.

The remaining data were normalized once all undesirable samples were removed. Normalization was conducted individually for all signals using the MinMaxScaler, transforming the data to be between zero and one. When normalizing for each individual signal, we ensure

that the normalized values are relative to the range of the respective refrigerator, such that the different temperature range only affect the measurements of said refrigerator. Normalization was utilized to ensure that all samples were within the same range. After interpolation, curation, and normalization, the data was ready for dimensionality reduction.

3.2 Algorithms for Dimensionality Reduction

The initial step of this thesis was to implement dimensionality reduction methods (DRM). The three DRMs presented below took the curated data and reduced it to a lower-dimensional representation, called dimensionality reduced data (DRD).

To ensure the only changing factor for this step was the chosen DRM, the input and output dimensions were similar for all methods. The input contained 40 dimensions, and the reduced output contained three dimensions. The methods might perform differently given different input and output dimensions, we did some experiments, but have chosen not to include the results in this thesis.

There were several reasons why reducing to three dimensions was desirable. Smaller data size led to a substantially reduced computation time. Three dimensions enable 3D plotting for data visualization that makes it easier for humans to analyze. It was also a point of interest for research purposes. The goal was to learn if such a significant dimensionality reduction would provide sufficient information to conduct clustering. To evaluate if such a comprehensive reduction would suffice, we performed a manual evaluation of the results, and compared it to the state-of-the-art clustering algorithm for time-series data.

Obviously, such a significant reduction would decrease the accuracy and details of the data. Therefore, we additionally tested the clustering algorithms using different DRMs before concluding. This was done ranging from three to ten dimensions. When adopting a dimensional space larger than three, visual representation tend to become complicated. We used *linear discriminant analysis*, a method for separating classes and reducing dimensions [23], to represent reduced data in two dimensions visually.

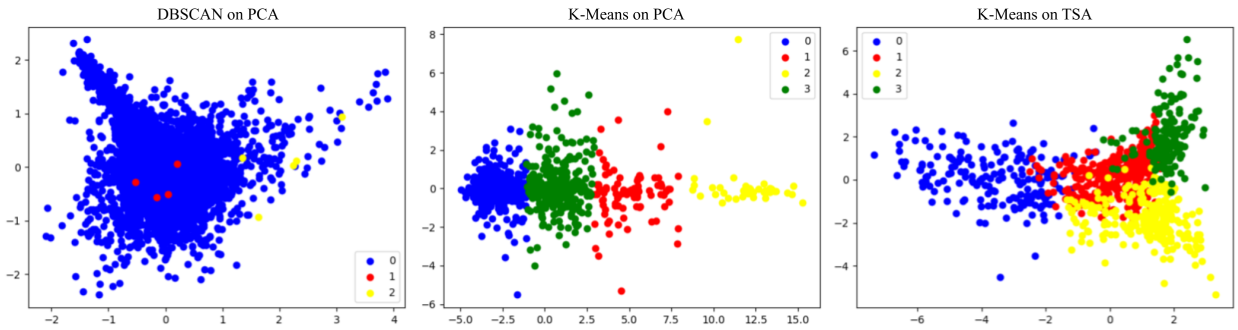


Figure 3.10: Different clustering algorithms on 10-dimensional DRD.

Surprisingly, larger dimensional spaces had little effect on defining distinct and separated clusters. Hence we concluded to use three dimensions for this thesis.

3.2.1 Principal Component Analysis

The first DRM applied, was the PCA. The DRD from the PCA would account for the largest possible variance within the data, which was appropriate for the upcoming clustering. As all dimensions are units of temperature, every dimension will contribute to the principal component equally.

A principal component analysis was implemented to reduce the data input with 40 dimensions to the desired three dimensions. This model was implemented using the PCA module from *Scikit Learn* [22]. This package is an *off the shelf* solution, requiring only the number of principal components to be specified before fitting.

To preserve resources, the model was fitted onto a representative 10 000 data samples. The entire dataset was then reduced by extracting the three foremost principal components. These principal components will account for the majority of variation in the data.

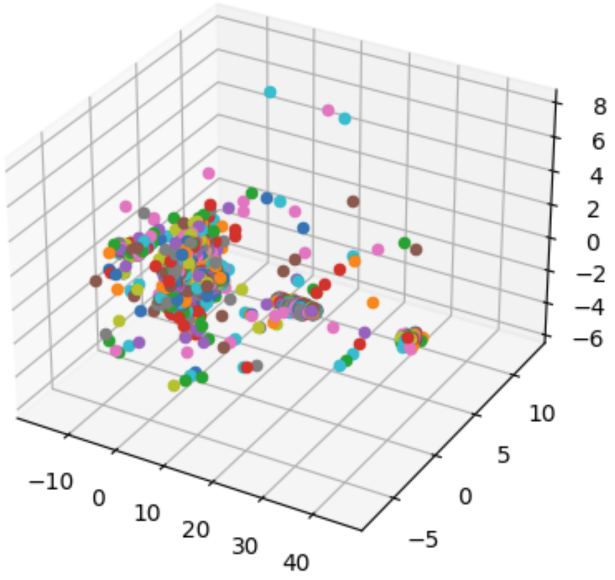


Figure 3.11: Visual representation of the data after PCA reduction.

3.2.2 Time-Series Autoencoder

The second DRM used for reduction, was the Time-Series Autoencoder (TSA). The TSA combines the strengths of AE and LSTM. AE was chosen due to its established application for reducing dimensionality. It is also a neural network, which we have experience with as a part of our field of study. LSTM is considered a highly suitable method for extracting important information from time-series, which lead to the exploration of potential possibilities for a combination of AE and LSTM.

The structure of the model was more flexible than the PCA, so it could be customized for time-series using LSTM layers. We used the flexibility of neural networks to implement the custom method TSA for dimensionality reduction.

The TSA was implemented using Keras, an API for the deep learning library Tensorflow [41].

Two sub-models, the encoder and decoder, were independently implemented and used to construct the TSA model. As mentioned in Section 2.4.2, the connection between these sub-models and the latent-space, provided a compressed representation of the input data, and was used for the dimensionality reduction.

As mentioned in Section 2.4.2 AEs are commonly symmetrical, however as this thesis revolves around time-series, we wanted to adjust the encoder using LSTM technology when creating the TSA. LSTM is well suited for extracting the overall patterns of time-series, which was admissible for the encoder. The encoder model starts with an LSTM layer, comprised of 40 LSTM blocks with an output length of 20. Next is another LSTM layer, comprised of 20 LSTM blocks, with an output length of three. This is fully connected to a hidden layer of three outputs. This is the *latent-space*, and it is the compressed representation of the original input, and what the decoder will use to reconstruct the data.

The decoder model starts with three input values. The model is comprised of four fully connected layers, with an output layer of 40 output values. See Figure 3.12 for a visual representation of the models.

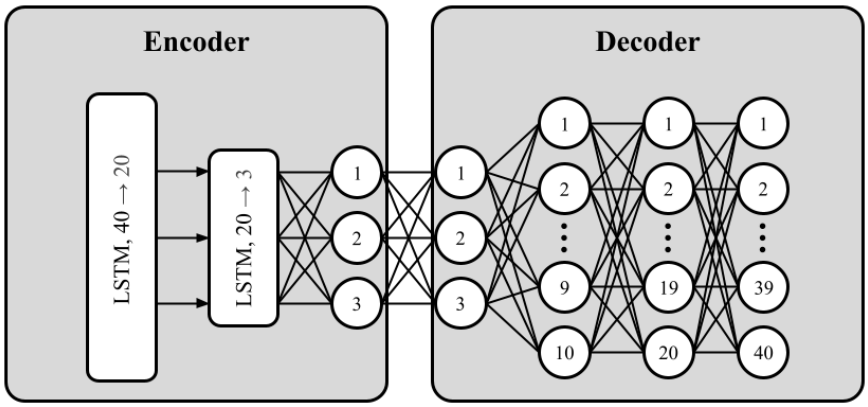


Figure 3.12: The structure of the TSA model.

The model was trained for 10 000 epochs, using backpropagation to minimize the loss between the input to the encoder, and the output from the decoder.

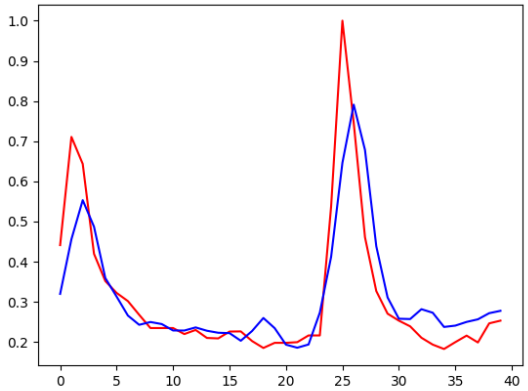


Figure 3.13: The decoder successfully regenerates the time-series from only three dimensions. Red is the input, and blue is the output.

By utilizing the LSTM's ability to extract important information from time-series, and the compression strength of AE, the TSA was able to successfully capture the behavior of the

data using only three *latent-variables*! Figure 3.13 shows how the TSA managed to encode and decode the data and capture the overall behavior successfully.

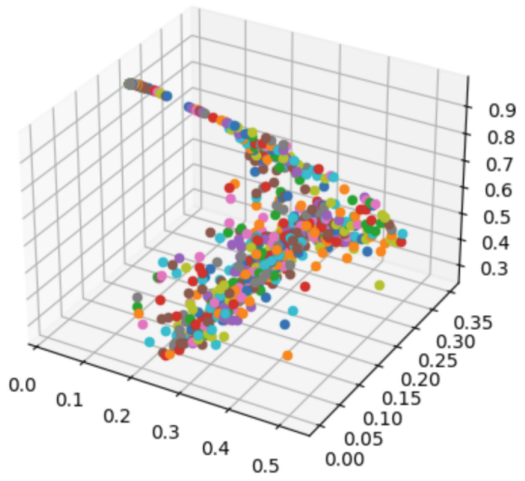


Figure 3.14: Visual example of the data after TSA reduction in 3D.

3.2.3 Self-Organizing Maps

SOM was the last of the three DRMs implemented for this thesis. A rationale for testing and using SOM was first and foremost because it was able to reduce from a high-dimensional to a low-dimensional representation of the input space, while retaining important information. Topology preservation can serve as an advantage, since a reduction in dimensionality traditionally results in loss of important and crucial information. A second rationale for using SOM is the fact that it did not exist in any common ML or AI packages, nor was it widely used, which made it an interesting method to explore. The drawback is that due to this, there were little to no previous implementations in which a comparison could be carried out.

SOM was implemented using the SOM package under *sklearn-som*. Although documentation was limited, the implementation was uncomplicated. Due to lack of comparison material, the implementation, and especially the *fit* function was subject to experimentation. This function digest training data, a boolean value that tells whether to shuffle it or not, and a number of epochs. The default values were one epoch and shuffle set to true. Shuffling enabled randomization of order of training data when fitting [38]. This proved to be the better option.

Data of dimension 40 was input, and a reduction resulted in a three dimensional representation, which was desired.

An issue regarding all DRMs validating the quality of the feature extraction. The TSA DRM could be validated by decoding the data from the latent space and computing the loss, but apart from that, there are no methods for validating the other two DRMs. Hence, there was no way to verify the quality of the extracted features from the DRMs.

3.3 Algorithms for Clustering

After reducing dimensionality, application of clustering algorithms was due. Each method was implemented and optimized individually, but the reduction was performed uniformly such that each algorithm only took an input of dimension three. The rationale for implementing exactly the three following algorithms were that they each have properties that can be desirable for clustering, and that they have different approaches for clustering.

3.3.1 K-Means

The first clustering algorithm that was implemented was the *K-Means*. We implemented K-Means using the K-Means class from the Scikit Learn library.

For K-Means, the number of clusters K must be defined before commencing, which is one of the largest drawbacks of this algorithm. The optimal K is linked to the data used. Hence, K will differ for each of the DRDs. To find the optimal K, the K-Means algorithm was executed iterative for increasing K values from two to 30 on the different DRDs. The *Silhouette Score* was computed for each iteration, and saved. The Silhouette Score was graphed, and the *Elbow Method* was used to find the optimal K. We did not find an obvious *elbow* for all of the graphs, but the results were helpful in choosing K. For the PCA DRD and SOM DRD data the chosen K was 13, while for TSA DRD it was 20. Figure 3.15 shows the graphed Silhouette Score for the individual DRMs.

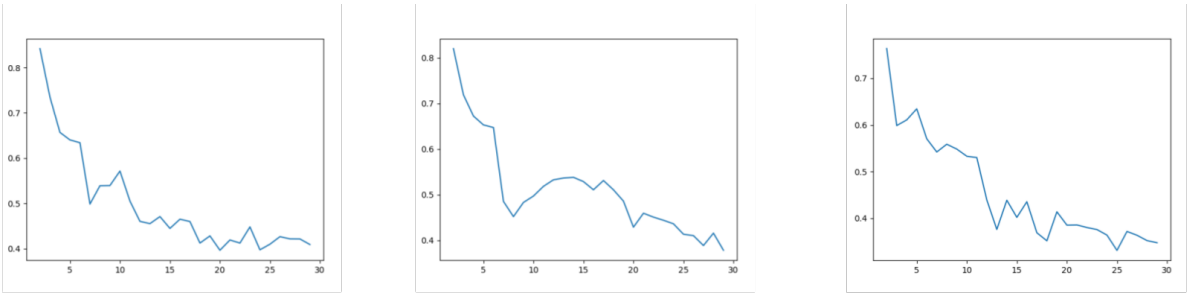


Figure 3.15: The graphs shows the results of the elbow method when performed on all the DRDs for K-Means.

3.3.2 DBSCAN

The second clustering algorithm for this thesis was the DBSCAN algorithm. Data from the DRMs had arbitrary shapes, and hence DBSCAN's ability to conduct clustering on arbitrary shaped clusters could be advantageous. As we are missing ground truths for the data, using an algorithm such as DBSCAN, which does not require a specified amount of clusters, was suitable.

This algorithm was implemented using the DBSCAN class from Scikit Learn. The algorithm was implemented and adjusted to be run on three different data sets, from the three DRMs. The DBSCAN algorithm is controlled by two parameters, ϵ and *min_ samples*. ϵ defines the radius around a data point to look for other data points. *min_ samples* defines how many

data points within the radius are required for the data point to be considered a *core point*. DBSCAN is sensitive for these parameters, and different parameters yield different results. This means the optimal parameter was different for each DRM. Hence finding the optimal parameters for this scenario is important.

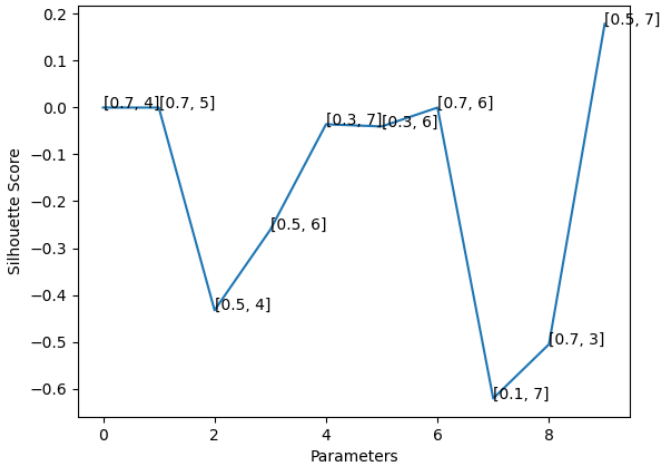


Figure 3.16: Silhouette Score for different parameters using the data from PCA.

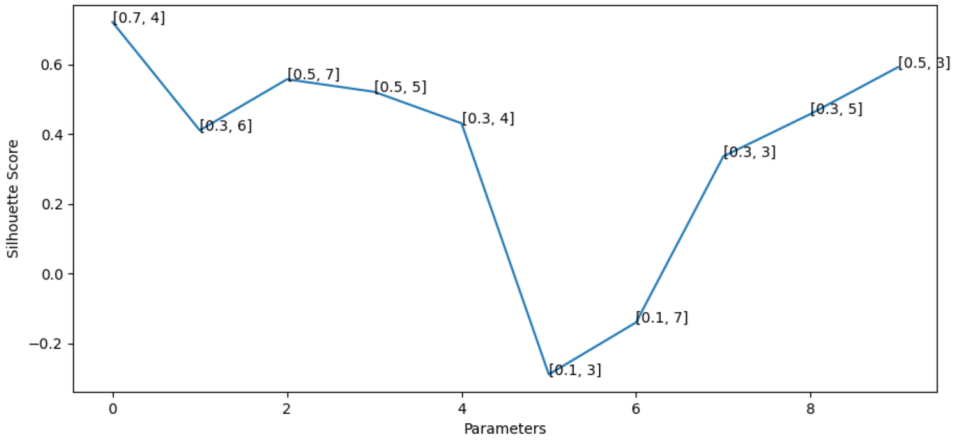


Figure 3.17: Silhouette Score for different parameters using the data from SOM.

There are an infinite possible combinations of parameters to explore for DBSCAN, and it was impossible to test the performance for all. Two methods were used to find the optimal ϵ and $min\_samples$ for PCA, TSA, and SOM DRDs. The first method implemented to find the optimal parameters for DBSCAN on the TSA DRD was a brute-force method. Although crude in its nature. Testing every possible combination from $\epsilon = 0.001$ to $\epsilon = 1$ with a 0.001 increment, as well as from $min\_samples = 1$ to $min\_samples = 10$ with a increment of 1. Hence, a total of 10 000 parameter combinations were tested. The second method implemented to find the optimal parameters for DBSCAN on PCA and SOM DRDs was a probing method. The parameters are randomly paired from two lists, one containing possible ϵ values, and one containing $min\_samples$ values.

The algorithms were run on a smaller, randomly selected, subset of the total dataset. The performance of the algorithms was then evaluated using the Silhouette Score. Once the estimated optimal parameters were found, the algorithm was run on the entire dataset.

The estimated optimal parameters for DBSCAN using the PCA DRD is $\epsilon = 0.5$ and

$min\_samples = 7$ as seen in Figure 3.16. And the estimated optimal parameters for DBSCAN using the SOM DRD is $\epsilon = 0.7$ and $min\_samples = 4$ as seen in Figure 3.17. No estimated optimal parameters were found for the TSA DRD.

Two methods were used because the second method, the probing method, did not yield any results for the TSA DRD. The first method, the brute-force method was not applied to the PCA and SOM DRD due to limitations in the implementation and resources. The brute-force method implementation was built on the assumption that most, if not all, parameter combinations would fail, which means it would not have been time efficent to apply the method to the PCA and SOM DRDs, as they would create clusters with a larger than near zero number of parameter combinations.

3.3.3 Agglomerative Hierarchical Clustering

Lastly, the Agglomerative Hierarchical Clustering (AHC) algorithm was implemented. AHC was implemented using the *AgglomerativeClustering* class from Scikit Learn.

The AHC algorithm could be adjusted using two mutually exclusive parameters, *threshold* and K. The *threshold* parameter is the maximum distance between two sub clusters allowed for them to merge into a new sub cluster. While the K is the desired amount of clusters. If either of these parameters is specified, the algorithm will halt when the criterion is met. Alternatively, it will generate the entire binary tree.

A smaller *threshold* will lead to more clusters that are less diverse. This will lead to high performance, but could in turn prevent the algorithm from finding larger over-arching clusters. Defining a high K will lead to the same issue.

To find an optimal number of K, the elbow method was applied ranging from one to 30. The results were evaluated using the elbow method. Elbow method and threshold parameter are mutually exclusive, only one can be applied at a time. For this thesis, only K was tested.

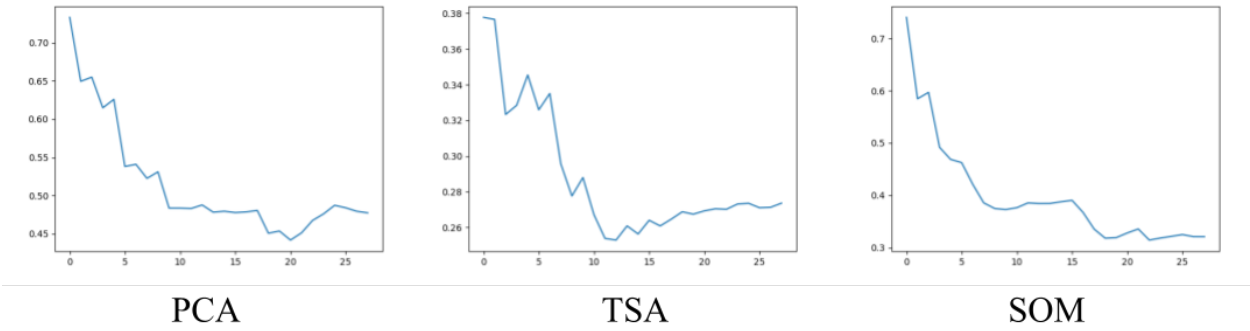


Figure 3.18: The graphs shows the results of the elbow method when performed on all the DRDs for AHC.

3.4 TSK-Means

The three clustering algorithms above were compared to the de facto solution for clustering of time-series, namely TSK-Means, using dynamic time warping (DTW). This algorithm

clusters the samples without dimensionality reduction, hence is more computational complex than the other algorithms discussed above. TSK-Means has proven to be a good candidate for clustering time-series data, and it was therefore interesting to test this algorithm for this thesis. Furthermore, it is an algorithm clustering directly on the time-series data without feature extraction.

Using a Python package for time-series analysis called *tslearn* the algorithm was implemented. This algorithm is highly customizable, with four different parameters. These being number of clusters (K), number of iterations for barycenter computation (max_iter_barycenter), number of trials (n_init) and distance metric. Similarly to K-Means, TSK-Means' performance was reliant on an appropriate K value. For smaller K values we discovered that the algorithm performed poorly, and no patterns was found. Instead it formed horizontal barycenters. We think this might be due to the *curse of dimensionality*, enforcing the algorithm to minimize the over all loss by averaging too many samples for one barycenter.

After further investigation of the clusters, we decided to conduct two rounds of clustering. Initially the entire dataset was clustered into ten clusters. No distinct patterns was formed here. Commencing we conducted clustering on all the samples within each individual cluster, to form additional ten clusters. Hence in the end, we had 100 clusters from the TSK-Means algorithm.

3.5 Evaluation Criteria for the Clustering Results

Before evaluation of the results, some ground rules were established. The ground rules covered known patters, performance metrics, and visual analysis. When the results were evaluated, were all three of these factors utilized. Combined, they provided an insight into the performance of the different algorithms and methods implemented and tested in this thesis.

3.5.1 Known Truths

As an added feature, we installed a sensor in a control refrigerator to evoke known manufactured patterns. These patterns will further be used to evaluate the results by comparing them to the clusters formed by the algorithms. Unfortunately, some scenarios are complicated to manufacture, such as malfunctioning thermostats, etc. We name the artificially constructed patterns *manufactured known patterns* (MPK).

Scenarios we manufactured are:

MKP-0 Normal Operation

When the refrigerator is operating normally, there are seasonal fluctuations in the temperature as seen in Figure 3.19.

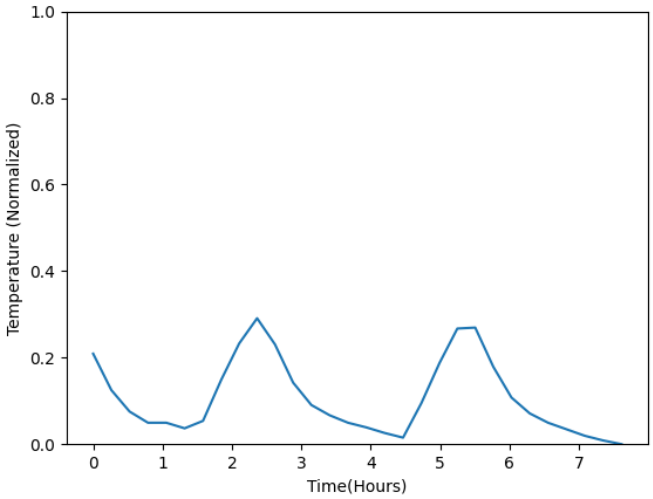


Figure 3.19: Normal behavior of the temperature.

MKP-1 Open Door

Suppose the door of the control refrigerator is left open. In that case, the temperature will increase rapidly and form a distinct curve, which can be seen in Figure 3.20, when the refrigerator goes from normal operation to the door being open.

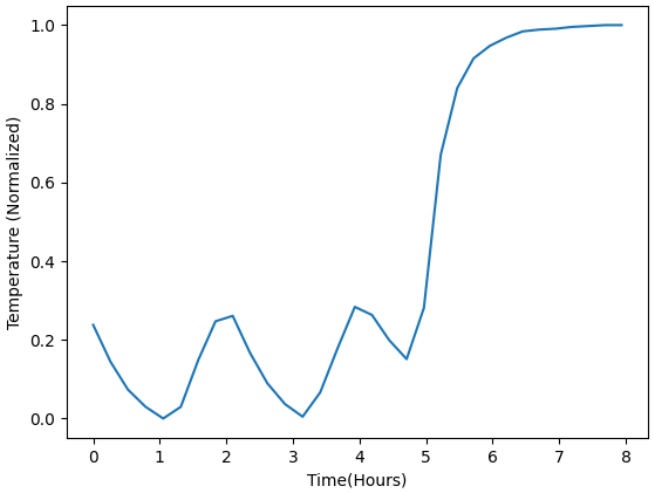


Figure 3.20: The temperature behavior for when the door is left open.

MKP-2 Closed Door

Deliberately leaving the door open resulted in an expected increase in temperature. Subsequently, the door was shut. Naturally, this resulted in a decrease in temperature. Granted, it being a slow decrease, and the refrigerator takes over 8 hours to normalize.

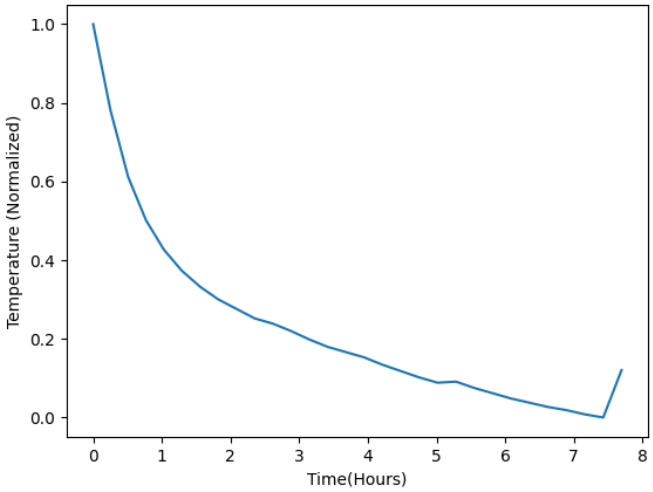


Figure 3.21: The temperature behavior for when the door is closed after being left open.

MKP-3 Refrigerator Turned Off

If the refrigerator is turned off, and remains offline, the temperature will increase, however, not as rapidly as an open door. It is slowly increasing and stabilizing at room temperature after about 9 hours, as seen in Figure 3.22.

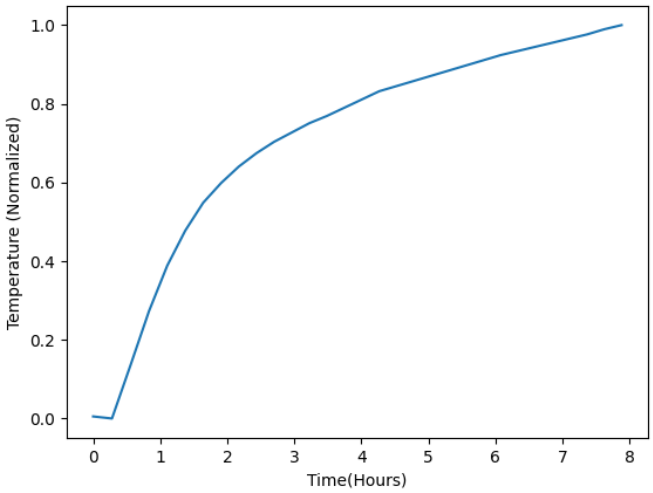


Figure 3.22: The temperature behavior for when the refrigerator is turned off.

MKP-4 Refrigerator Turned Back On

As expected, the temperature started dropping once the refrigerator was turned back on. Surprisingly, the curve is not similar to the curve from Section 3.5.1. This curve is more linear, and takes about 8 hours to reach normal operation temperature.

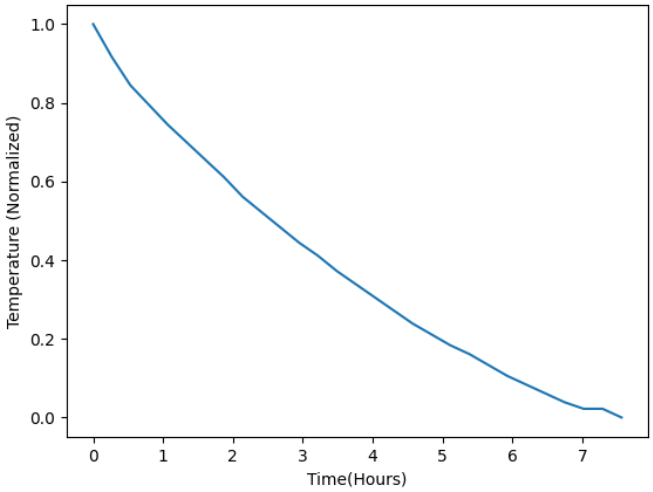


Figure 3.23: The temperature behavior for when the refrigerator is turned on after being turned off.

MKP-5 Goods Delivery

When delivering goods, they usually have different temperatures than the internal refrigerator temperature. This forces the internal temperature to change before the refrigerator has been able to adapt and stabilize the temperature again. This pattern was manufactured by placing four room temperate water bottles inside the control entity.

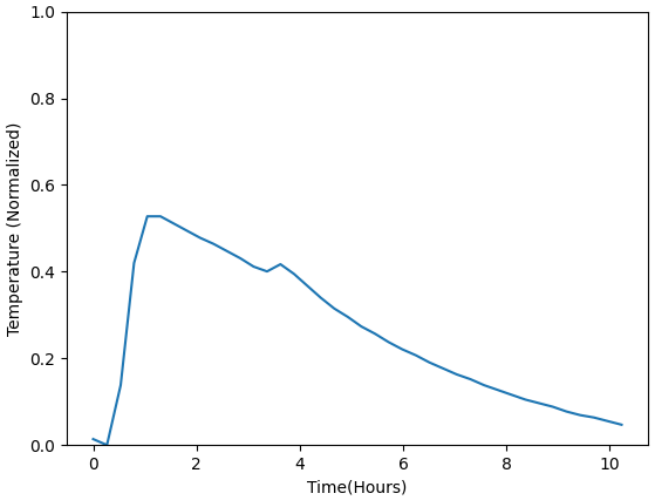


Figure 3.24: The temperature behavior of the refrigerator for when goods are delivered.

3.5.2 Performance Metrics

As stated in Section 2.6, clustering algorithms are unsupervised learning methods, which cannot be evaluated for, e.g., loss or accuracy, unlike supervised machine learning. To evaluate clustering performance, three performance score metrics were implemented, namely Silhouette Coefficient, Calinski-Harabasz Index and Davies-Bouldin Index. The rationale for using these performance metrics was because they support algorithms in which ground truth labels are unknown. This means that no labels to check against after training are provided. There exist several performance score metrics, however, these are only applicable to algorithms where ground truth labels are known.

These score metrics provide a numeric evaluation for the clustering algorithms. This evaluation will be used to compare the different clustering algorithms, as well as the different DRMs, as seen in Figure 3.25.

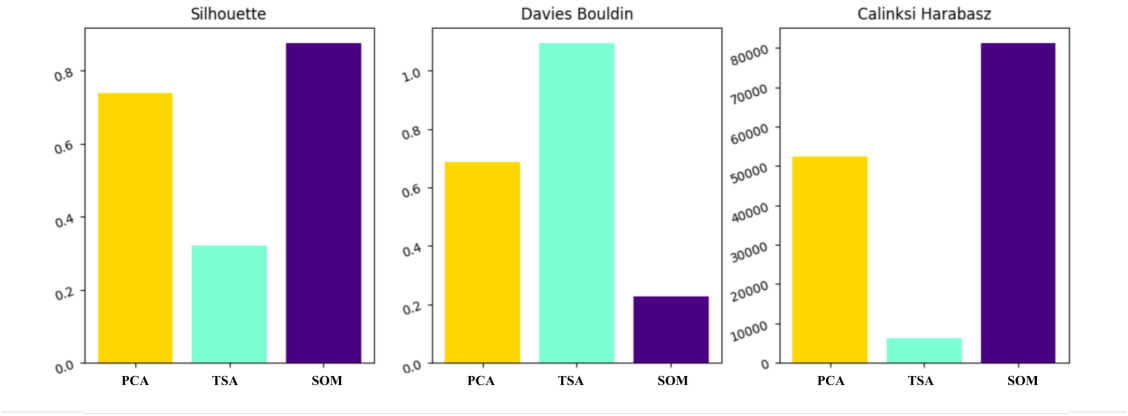


Figure 3.25: Example of Performance Score Results.

All three performance metrics are implemented via sklearn. A single module is created to calculate and plot for all metrics, which enables immediate score comparison for each clustering algorithm.

3.5.3 Visual Inspection and Domain Expert

Although performance metrics indicate what respective algorithm was able to achieve, it is desired to visually inspect and analyze the results for patterns. Each cluster was manually checked for patterns, then any observable pattern was compared to the MKPs.

Patterns are also subject for evaluation by a domain expert. This domain expert is an employee at Bitmesh AS, who has gained experience regarding temperature fluctuation after working in the field of abnormality detection and deviation management. They were able to provide valuable insight into these patterns, related to cause, consequences, and potential future application.

Altogether, performance metrics, MKPs, and domain expertise lay a foundation on which the results will be evaluated. This information enables evaluation for the performance of the clustering algorithms and the patterns they discover on a metric and practical level.

3.6 Tools and Organization

Workflow and structure was mainly organized by using Scrum, GitHub, Azure, and Discord. Scrum was the agile method chosen, and its main purpose was to ensure proper and seamless workflow. Every issue, or task, written in Scrum can be thought of as brainstorming, in which they represent a suggestion for project requirements. Decomposing a project scope into small issues can be helpful for larger projects, and when work needs to be distributed among multiple people.

GitHub served as a code repository to which implementation was pushed. For projects consisting of several contributors, it is advantageous to have a common repository for peer reviewing code, as well as push and pull updates.

Access to Azure resources was provided by Bitmesh AS. Training a dataset of larger sizes, and running the algorithm after, requires a certain length of time. In order to let it run uninterrupted, processing was outsourced to a virtual machine in Azure. This reduced strain and processing power of personal computers.

Azure Compute instances were mainly used for training TSA, as it required a significant amount of time to become sufficiently proficient.

Discord served as the main channel for communication. Weekly stand-up meetings were held, results were shared, information was evaluated, and general communication found place here.

Chapter 4

Results and Evaluation

In this chapter, we present, evaluate, and compare the results of the proposed algorithms for this thesis. We will introduce this chapter by presenting results from the DRMs and visualize the data in 3D space. Due to having implemented multiple different clustering algorithms, results will initially be presented for each algorithm before the results are compared. In addition, we have included performance scores to provide an indication of how well each algorithm has performed. For *K-Means*, *DBSCAN* and *AHC*, the clusters will be presented in 3D scatter charts, to display how clusters were formed. Samples within clusters will then be plotted. This is essential for the discovery and identification of intra-cluster patterns. For convenience, only graphs of relevant and discussed clusters will be included in this chapter. The remaining graphs can be seen in Appendix B. We will, in addition to presenting results from the main algorithms, present the TSK-Means algorithm. This will be the baseline for the other three clustering algorithms.

As explained, the discovered patterns were discussed with a domain expert from Bitmesh AS, which provided invaluable feedback and aided in the interpretation of results. Performance scores, visual inspection and domain expertise will set the foundation on which the results are evaluated, and establish to what degree the results are applicable in the field.

4.1 Dimensionality Reduction

Three DRMs produced their respective DRD. The purpose of the DRMs was to provide a lower dimensional representation of the data. As mentioned in Section 2.4, overcoming the *curse of dimensionality* can be accomplished by reducing dimensionality, hence making the clustering process less computational complex. To accomplish good DRD, data should be distributed across the three dimensions, and the position of a sample in 3D space should represent its behavior.

DRD can be viewed as a smaller representation of original data. Little spread in DRD indicate lack of diversity in the original data, and that even outliers and abnormal patterns are relatively similar to normally observed patterns and operations. An unfortunate consequence of reduction was therefore little separation and that data was grouped closely together for all DRMs.

As observed in Figure 4.1, most of the datapoints are located in a triangle shaped cluster, with some outliers. The highest single axial number of the PCA DRM was 25.97 and the

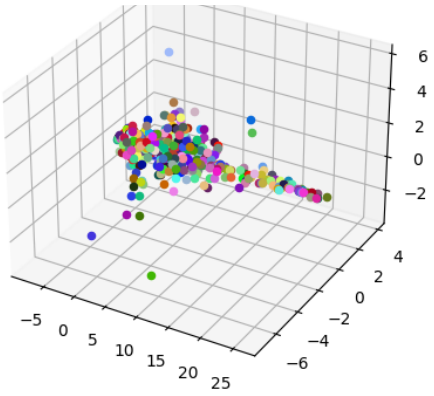


Figure 4.1: Representation of the results for the PCA DRM.

smallest number was -11.20.

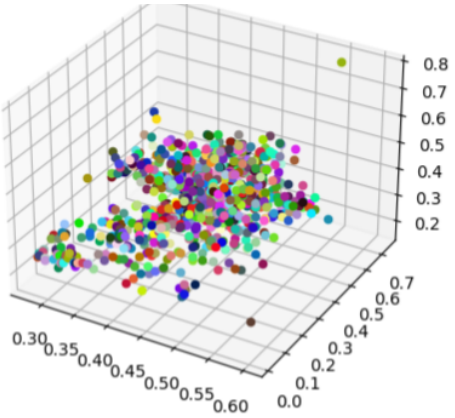


Figure 4.2: Representation of the DRD from TSA DRM.

In contrast to the other two DRMs, the TSA exists on a 0 to 1 axis scale, as seen in Figure 4.2. The datapoints do not take a particaluar shape, but truly looks like a point cloud. The shape of the datapoints made it difficult to cluster on, even with its clearly defined outliers. The highest single axial number of the TSA DRM was 0.7774 and the smallest number was 0.0009.

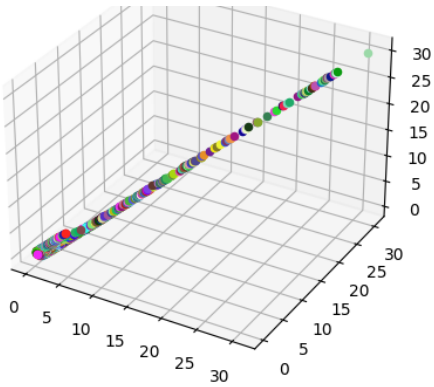


Figure 4.3: Representation of the results for the SOM DRM.

The SOM DRM has the shape of a linear line, with a slight droplet shape towards the (0,0,0)

origin, as seen in Figure 4.3. The shape produces no visually obvious clusters, and has no clear outlier compare to the other two DRMs.

The highest single axial number of the SOM DRM was 30.99 and the smallest number was 0.06.

Ideally, the DRM would distribute its data over the entire 3D-space, with clearly separated clusters of data points. It is uncertain if any DRM would succeed at this, given the insufficient dataset, or if more tinkering of the proposed methods would result in any improvements.

We speculate that due to high normal to abnormal sample ratio, the DRM struggled to extract valuable features for describing different behaviors. *Normal samples* are here used for samples from when the refrigerator was operating normally, while *abnormal samples* for when the refrigerator operates outside expected behavior. Due to this unbalanced distribution, we expect the DRMs to be biased towards describing normal samples, hence ignoring features which might have been beneficial for describing abnormal samples and aiding the feature extraction process. Balancing the ratio of normal and abnormal samples, perhaps remove normal samples entirely, could be helpful. The problem is however, that there are no labels in the data, and selection of samples would have to be performed manually which is a tedious and time consuming task. This is assumed we could classify a sample as normal or not with high accuracy.

4.2 K-Means

This section will address the clusters formed by the *K-Means* algorithm. This algorithm was conducted on all three DRD. We will evaluate the clusters based on the similarity of data points within a cluster, and how each cluster applies to the real world.

Figure 4.4 represents performance score for K-Means, for each DRD. Silhouette Score was high for PCA and SOM, while a little lower for TSA, overall, performance scores are satisfactory. For Davies-Bouldin Index, the score was between 0.4 and 0-5 for PCA and SOM, while not great for TSA. For Calinski-Harabasz Index, the score for PCA was high, while for SOM it was lower and for TSA, quite low. This was expected as the DRD from the TSA method was denser, less scattered, and more spherical shaped than the linear appearance of PCA and SOM. Despite this, differences in performance were small for Silhouette Coefficient and Davies-Bouldin Index, except for the Calinski Harabasz score.

For the PCA DRD we observe that all clusters were grouped together, except for a few outliers. This is expected as the data points are closely positioned. Despite the lack of distinct clusters, it is still interesting to look for patterns.

We observed that most clusters look similar, with no distinct pattern observable. Cluster 3 and cluster 9 however, had emerging patterns as shown in Figure 4.6. We observed that cluster 3 displayed a trend of slowly rising temperatures. There was noise here, which made it hard to differentiate each individual sample, but the general trend in the samples for this cluster was increasing after 5 hours. This had similarities to both the patterns from an MKP-1 and MKP-3. The observed pattern was interesting, as detecting and classifying rising temperature was a core aspect in this thesis. However, the domain expert could not elaborate the likely cause of this pattern, and with only one such pattern detected, it could not provide sufficient understanding to classify different patterns in rising temperature. We also observed

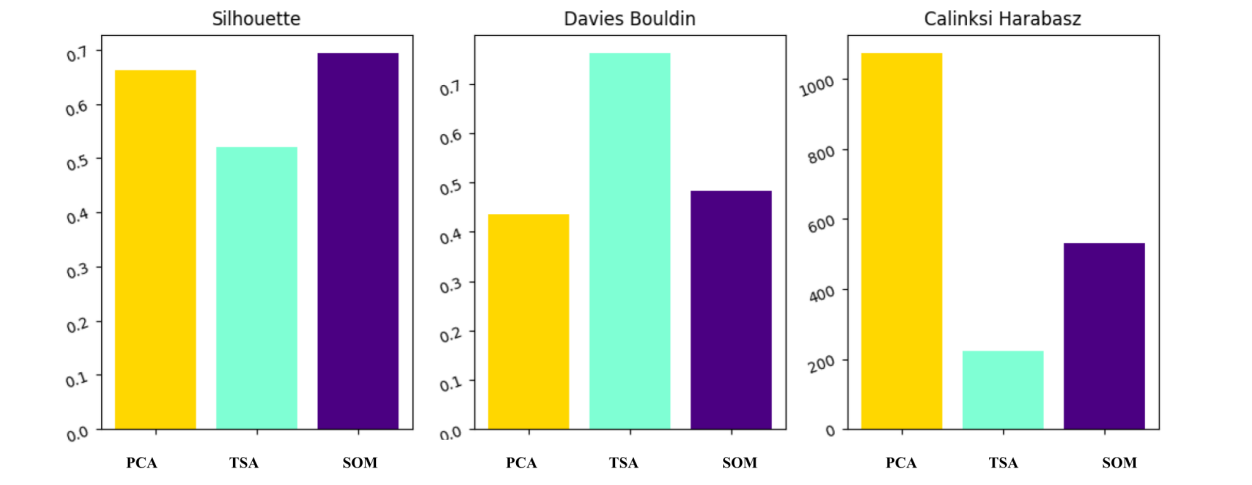


Figure 4.4: Performance scores for the results of the K-Means algorithm.

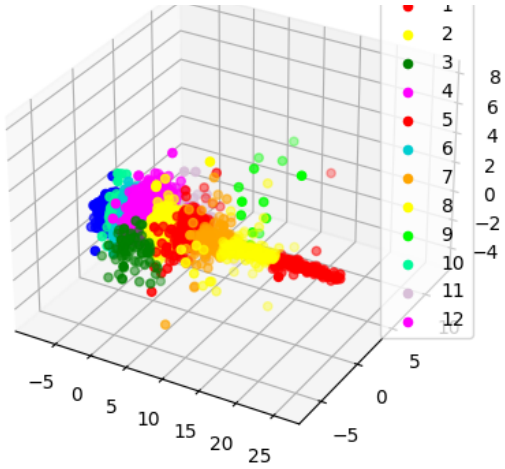


Figure 4.5: Clusters formed on the PCA DRD by K-Means.

that cluster 9 displayed a trend of descending temperatures. Dropping temperatures are also points of interest, which made the pattern discovered by this cluster particularly interesting. Cluster 9 contained both drastically dropping and slowly descending temperatures, which lead to the cluster not containing any distinct pattern. Hence it is important to emphasize that one such pattern alone will not provide enough information for further classification.

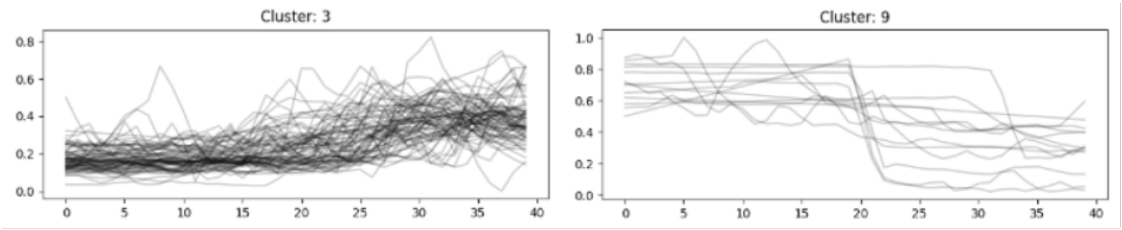


Figure 4.6: Visualization of the samples within cluster three and nine from K-Means on PCA DRD.

The K-Means algorithm struggled to find distinct clusters for PCA DRD, hence few interesting and valuable patterns were discovered. From this we can conclude that K-Means on

PCA DRD did not perform as desired.

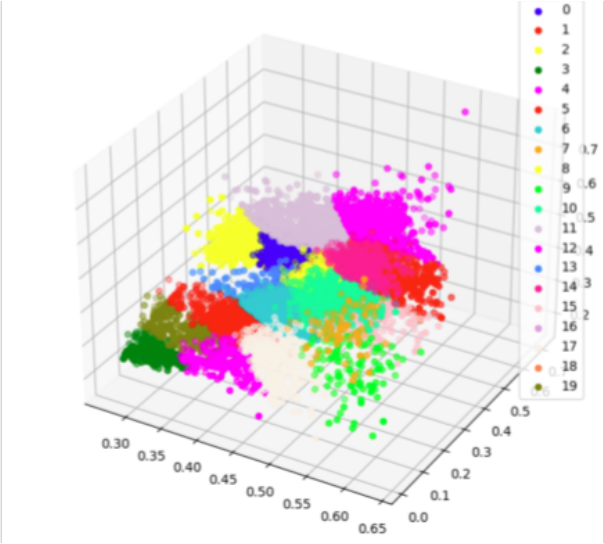


Figure 4.7: Clusters formed on the TSA DRD by K-Means.

From the performance score, we expected that the clusters formed by K-Means on the TSA DRD would be in close resemblance. Figure 4.7 proves this. All clusters were grouped together, with no obvious separation between them.

We observed this resemblance when plotting the graphs for each cluster as well. There was a significant amount of noise in each cluster. Nevertheless, cluster 7 and cluster 9 showed trends of increasing temperatures, similar to both MKP-1 and MKP-3. From Figure 4.7 we could see that 7 and 9 were neighboring clusters, which indicated some correlation between position of a sample in 3D space from the DRD, and the behavior of said sample. The slowly rising temperature is commonly found in open or turned off refrigerators according to the domain expert. The domain expert also noted that it is more likely that the pattern from cluster 7 is caused by an open door, while the pattern from cluster 9 is caused by turning of the refrigerator. This corresponds to the similarities of cluster 7 and cluster 9, and MKP-1 and MKP-3.

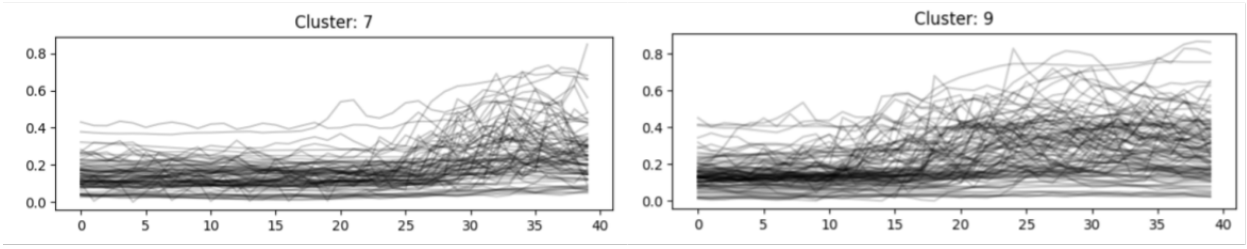


Figure 4.8: Visualization of the samples within cluster 7 and 9 from K-Means on TSA DRD.

Despite hints of patterns discovered in cluster 7 and cluster 9, the results from K-Means on TSA DRD were inadequate. This might be because TSA struggled with describing different patterns in latent space.

Lastly the K-Means algorithm was conducted on SOM DRD. For an unknown reason, the datapoints were distributed along one three dimensional line. This means the clusters will be slices on along this line, this is clearly seen in Figure 4.9. Naturally, the clusters then had little overlap as they were linearly positioned, hence the high performance score.

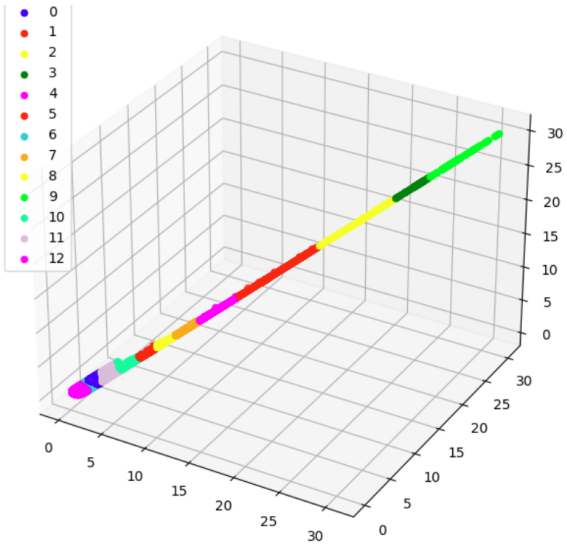


Figure 4.9: Clusters formed on the SOM DRD by K-Means.

We expected the placement of samples along this line to characterize the sample, hence samples with similar behavior would be close by. Unfortunately this was not the case. As seen in Figure 4.10 there seemed to be no intra cluster correlation. We did not observe any patterns from the results from K-Means on SOM.

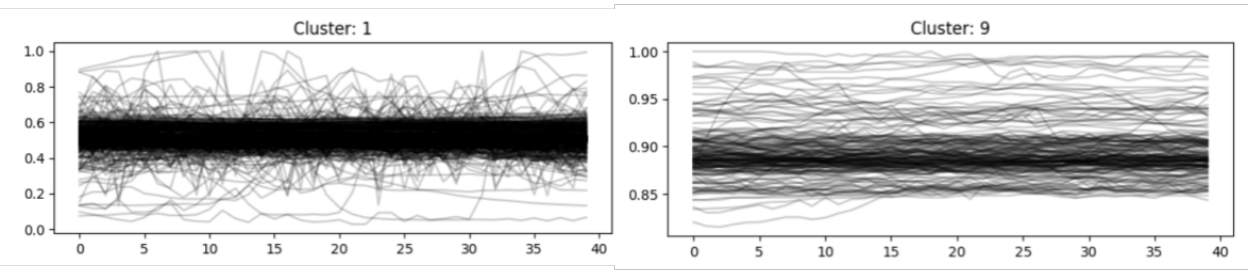


Figure 4.10: Visualization of the samples within cluster 1 and 9 from K-Means on SOM DRD.

Surprisingly, performance metrics and reality of the clusters formed by this algorithm, hardly correlated. The Silhouette Score suggested that for PCA and SOM, the clusters were well separated and distinguishable, indicating a well fitted clustering. Through visual inspection, we could determine that reality was not reflected. There was little intra-correlation in clusters with few exceptions. Calinski-Harabasz Index indicate that PCA outperforms the other DRDs, and although it can be argued that K-Means on PCA performed better than for the other DRDs, there was little difference when visually inspecting the clusters.

For K-Means, the results for all DRDs were not as expected. As already established, the performance scores did not properly reflect the reality of the clusters in this case. Therefore, application of elbow method to find an optimal K might have been flawed. Perhaps there is a specific hidden K in which this algorithm can discover additional distinct patterns more accurately.

To summarize the results from K-Means, we experienced that some patterns were discovered in both PCA DRD and TSA DRD, while none were discovered for SOM DRD. K-Means discovered patterns comparable to MKP-1, MKP-2 and MKP-3. Although some patterns were detected, it was not sufficient to use for further classification. We expected that the

DRD from PCA and TSA would provide better separation, and allow the K-Means algorithm to discover even more patterns, including varieties of similar behavior, such as fluctuating temperatures.

4.3 Density-Based Spatial Clustering of Applications with Noise

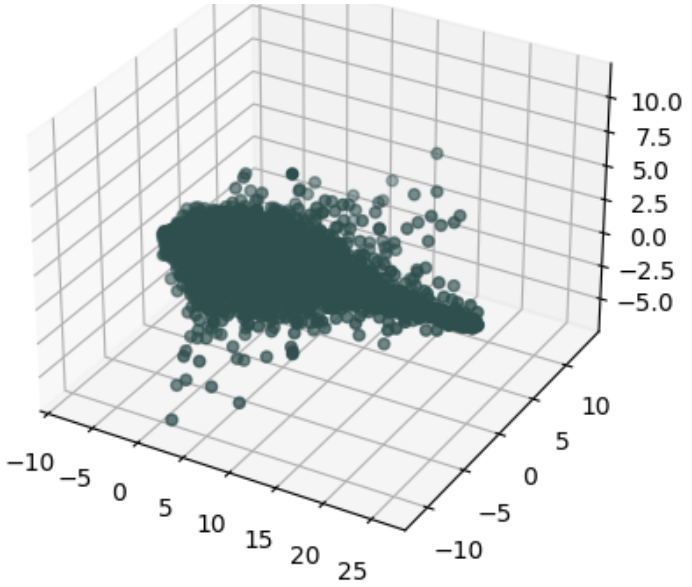


Figure 4.11: DBSCAN clustering on the PCA DRD.

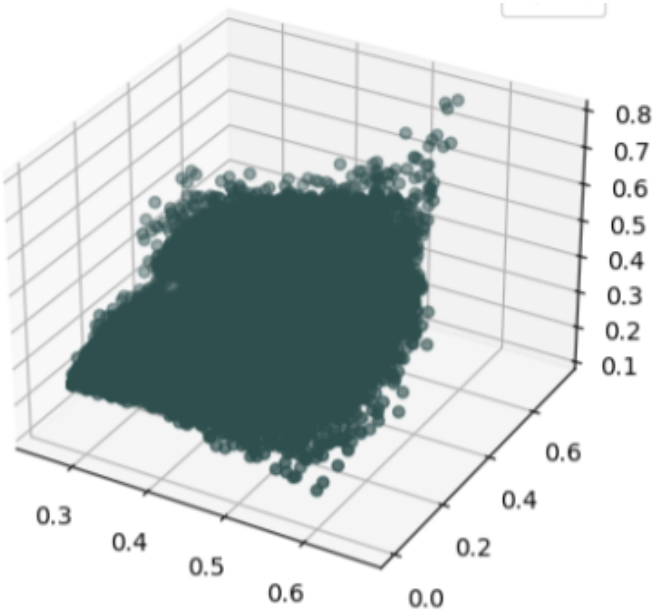


Figure 4.12: DBSCAN clustering on the TSA DRD.

The following section will address the results from the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm on all three DRDs. DBSCAN failed to form clusters as it could not find any separations in the DRD. This algorithm utilizes the different

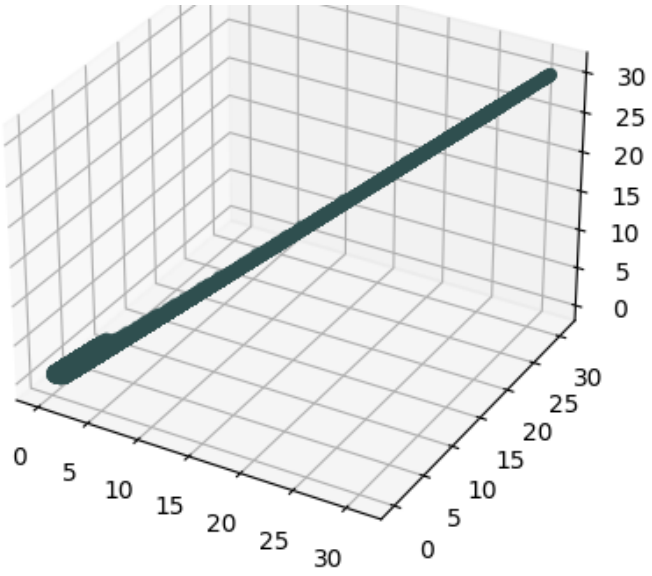


Figure 4.13: DBSCAN clustering on the SOM DRD.

density of data points to form clusters, but in this case the data contains little separation, and DBSCAN tends to view all samples as one large cluster.

As mentioned in Section 3.3.2, 10 000 combinations of parameters was tested for DBSCAN for TSA DRD. This extensive parameter testing indicated that there was no optimal parameters for this method, in fact, for TSA DRD, the algorithm yielded no viable results for any possible combination of parameters. This reflected the observation regarding no natural groupings in the data, and was the reason DBSCAN struggled to find clusters in the data, as seen in Figure 4.11, Figure 4.12, and Figure 4.13.

Since there are no sub-clusters is it not possible to further investigate the results of the DBSCAN algorithm.

4.4 Agglomerative Hierarchical Clustering

Agglomerative Hierarchical Clustering (AHC) was the last algorithm conducted on all three DRD. Figure Figure 4.14 represents performance score for AHC for each DRD. Silhouette Score for AHC on PCA DRD was almost 0.4, while higher for SOM DRD and lower for TSA DRD. Silhouette Coefficient returned relatively satisfactory results for AHC and was as expected. For Davies-Bouldin Index, a similar pattern applied, where SOM scored the best, PCA second, and TSA last. For Calinski-Harabasz Index, AHC on SOM DRD scored the highest. The performance scores indicated that AHC on SOM DRD would perform the best.

For the PCA DRD, we observed that all clusters were grouped together, with a few outliers, as seen in Figure 4.15. Examples of outliers can be seen in figure Figure 4.16. Aside from that, most clusters look similar, and there is no particular distinction of interest in patterns between clusters. Cluster 4 shows resemblance of a more comprehensive fluctuation. There were no trends to identify for PCA. Having interpreted these results, we observed that PCA DRD did not perform as desired, therefore, we concluded that results for AHC would not

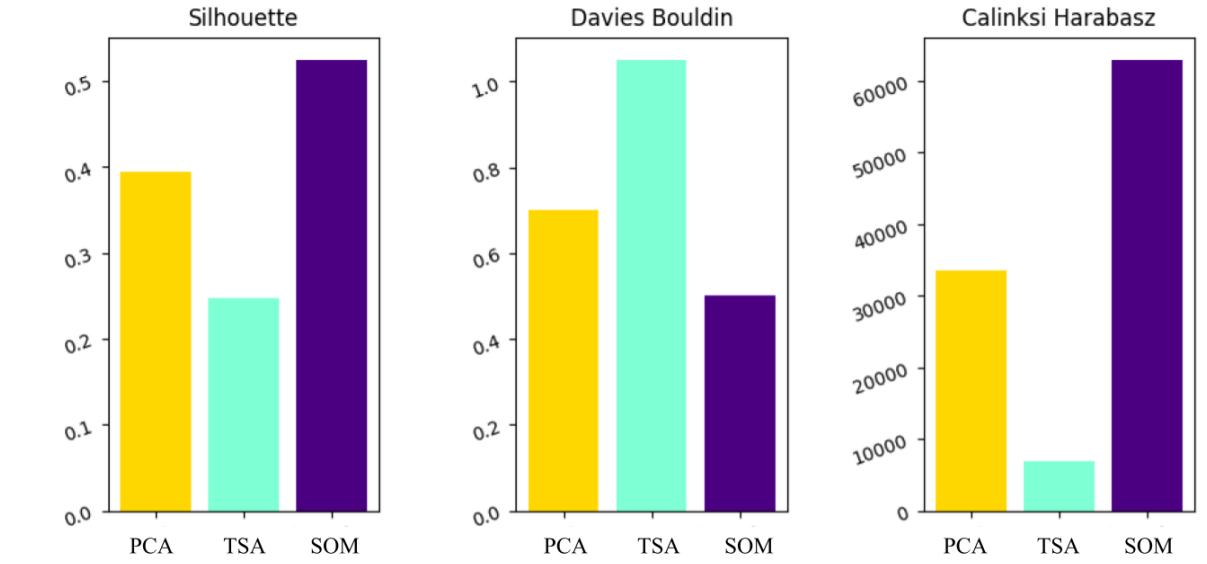


Figure 4.14: Performance scores for the results of the AHC algorithm.

provide sufficient understanding to perform any classification of patterns.

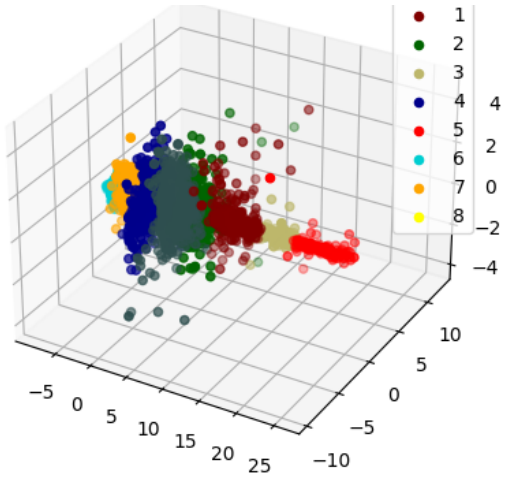


Figure 4.15: Clusters formed on the PCA DRD by AHC.

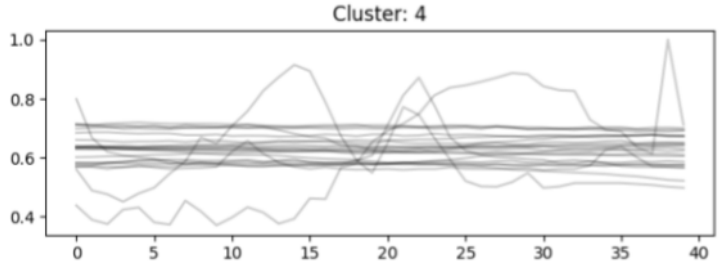


Figure 4.16: Visualization of samples within cluster 4 from AHC on PCA DRD.

For AHC on the TSA DRD, the case is similar to PCA DRD. Clusters were grouped together, there were no particular distinction between them as seen in Figure 4.17, and there were no trends to identify, except for a vague trend in cluster 9, as seen in Figure 4.18. This alone

will not suffice for any further classification. There is also a significant amount of noise in each cluster.

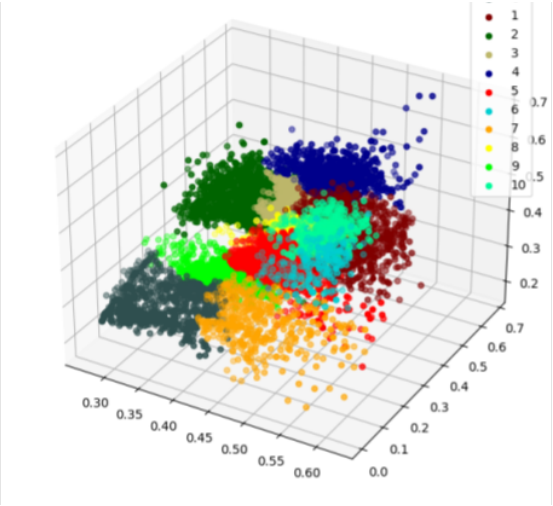


Figure 4.17: Clusters formed on the TSA DRD by AHC.

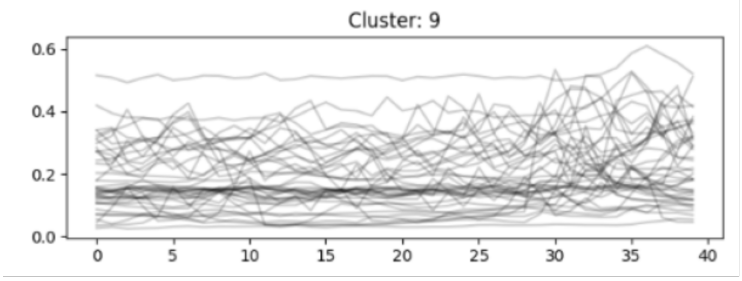


Figure 4.18: Visualization of samples within cluster 9 from AHC on TSA DRD.

SOM was the last DRM conducted for AHC, and is almost identical to SOM for both DBSCAN and K-Means, in terms of characteristics. This can be seen in Figure 4.19. Datapoints are distributed along this line, and clusters are therefore slices of this line. The only remark is that some clusters may have a significant amount of noise compared to the others. See Figure 4.20. Similar to K-Means, the performance scores and the reality of the clusters from AHC did not correspond.

To summarize results from AHC, we experienced meager pattern discovery. For PCA DRD and TSA DRD, resemblance of patterns could be identified, while none for SOM DRD. These patterns were inadequate for any further classification. We expected, as we did with K-means, that DRD from PCA and TSA would show a clear separation and allow the algorithm to discover more patterns for rising and falling temperatures. This was not the case.

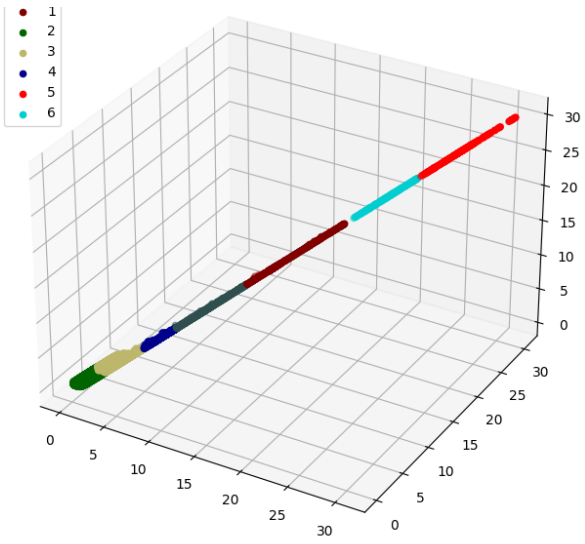


Figure 4.19: Clusters formed on the SOM DRD by AHC.

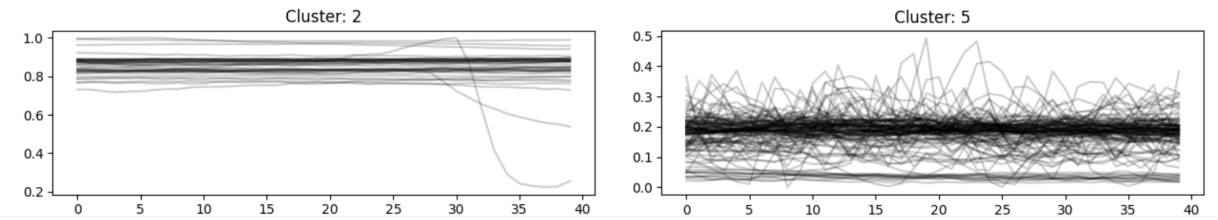


Figure 4.20: Visualization of samples within cluster 2 and 5 from AHC on SOM DRD.

4.5 Time-Series K-Means

Contrary to previous algorithms, TSK-Means did not use DRD to form clusters, instead, it took in raw time-series samples. Hence, the clusters cannot be presented in a scatter chart. The reader might remember that for this thesis, two iterations of clustering was done using the TSK-Means algorithm. Initially, ten clusters were formed on all the data, here we observed no distinct patterns in the clusters, but the clusters had variances as seen in Figure 4.21. Then each of the ten initial clusters were used to form ten new clusters. Patterns were observable in the second iteration. We will refer to the initial clusters, as *parent clusters* and the second iteration of clusters as *children clusters*. Appendix B shows all the clusters formed in this thesis, including children clusters from TSK-Means. The rationale behind two iterations of clustering was that by initially separating the data into larger parent clusters, the data would be divided into smaller batches, making the secondary clustering simpler. Thus we expected the algorithm to find more distinct clusters.

In this section, a red line drawn in the graphs represent the cluster center (barycenter), while black lines indicates samples within that cluster. We will evaluate each parent cluster and discuss what patterns, if any, that are observed.

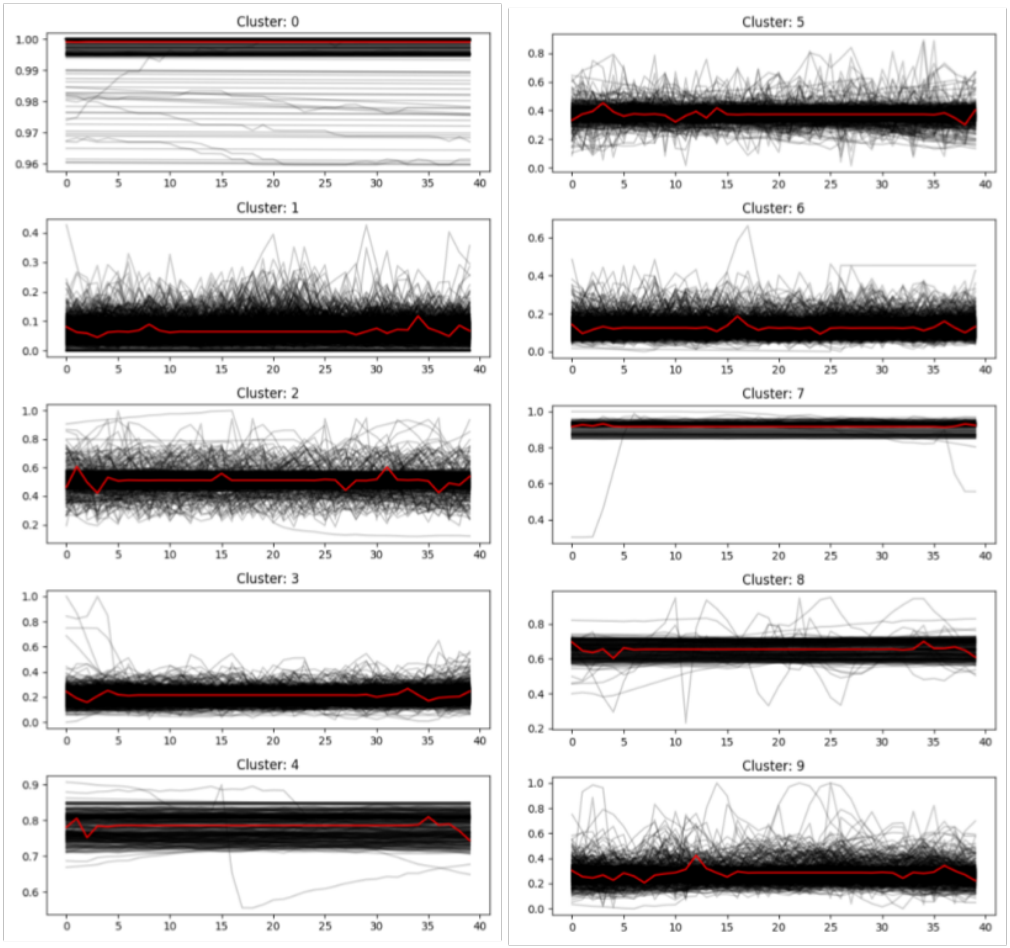


Figure 4.21: Visualization of parent clusters after the first iteration.

For parent cluster 0 there were no obvious patterns detected in any of the children clusters. The children clusters contained few samples, except child cluster 0, which seemingly contained mostly completely flat samples as shown in Figure 4.22.

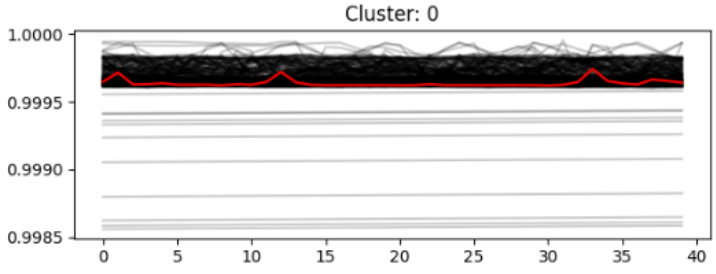


Figure 4.22: Visualization of samples in child cluster zero of parent cluster zero.

In parent cluster 1 we observed that child cluster seven displayed an interesting pattern. It was unclear what this pattern was representing, and it did not conform to any of the MKPs. It was however, compelling that TSK-Means found it, and that multiple samples behaved like this. The domain expert pointed out a possible defrosting pattern, and specified that defrosting intervals are commonly six hours. Although this did not account for the trend of higher temperatures in the end of the samples.

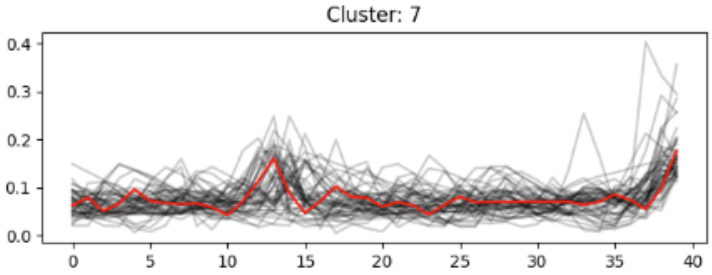


Figure 4.23: Visualization of samples in child cluster seven of parent cluster one.

Parent cluster two captured most seasonal patterns. This was not observable by looking at the parent cluster alone, but rather by looking into each child cluster. See Figure 4.24. We found it interesting that parent cluster two captured multiple, independent seasonal patterns. These clusters had similarities with MKP-0, which is normal operation. We could see that child cluster eight held the temperature for longer than child cluster five. The domain expert found this case particularly interesting, and pointed out that child cluster eight was a result of a better insulated refrigerator. This could be used to evaluate the quality of refrigerators, and in turn be used to choose environmental friendly refrigerators, thus reducing running costs. It was expected that TSK-Means would identify multiple clusters for normal operation, as this was the majority of the samples.

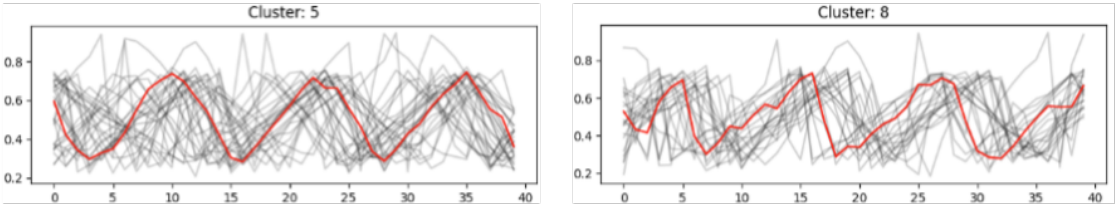


Figure 4.24: Visualization of samples in child clusters five and eight of parent cluster two.

For parent cluster three, only child cluster five contained an interesting pattern, as seen in Figure 4.25. This pattern displayed dropping temperature, comparable to MKP-2, which is marked as a point of interest. Despite the similarities to MKP-2, the temperature is descending faster than the MKP, which argued that it might not be related. According to the domain expert this pattern can be observed after a refrigerator is turned back on after being off for a longer period. There were, however, only four samples within this cluster, which made it less valuable.

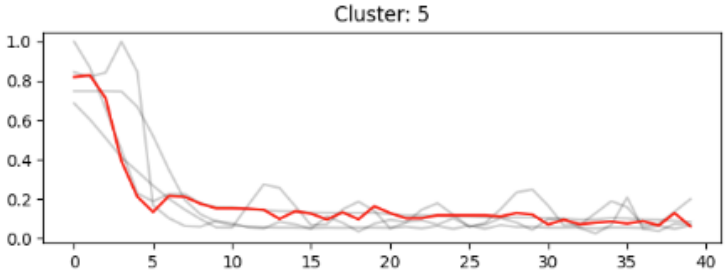


Figure 4.25: Visualization of samples in child cluster five of parent cluster three.

In parent clusters four and five, we could detect no obvious patterns. This meant that the samples assigned to each cluster were seemingly random, with little similarity.

We observed multiple child clusters with seasonal patterns in parent cluster six. These patterns have longer intervals than the patterns discovered in parent cluster two, as seen in Figure 4.26, and correlates to the six hour defrosting interval mentioned by the domain expert. Such patterns are common for refrigerators, thus not a point of interest for this thesis.

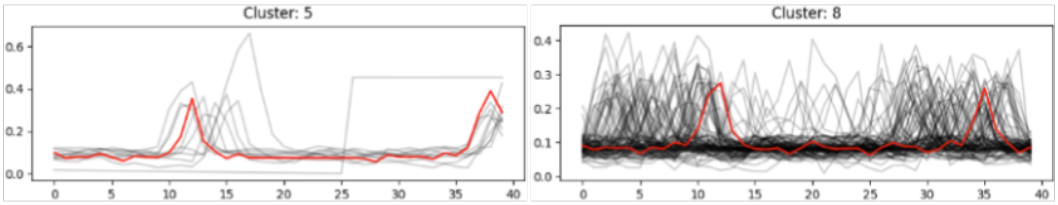


Figure 4.26: Visualization of samples in parent cluster six.

Both parent cluster seven and eight contained few samples. As a result, some child clusters were left containing only one sample. These child clusters were therefore insufficient for detecting patterns and results were not of interest.

The results from parent cluster nine however, were promising. Child cluster zero showed a pattern of slowly increasing temperatures, which were not similar to any of the MKPs. The domain expert argued it could be related to thermostats being adjusted or sensors being displaced. After approximately five hours, the temperature range seemingly shifts, indicating a change of sorts. This was marked as a point of interest, because the pattern might be subject for future work. Child cluster nine showed a pattern of a rapidly increasing, then slowly decreasing temperature, resembling the MKP-5, indicating delivery of goods. According to the domain expert this could be the case. The patterns can be seen in Figure 4.27.

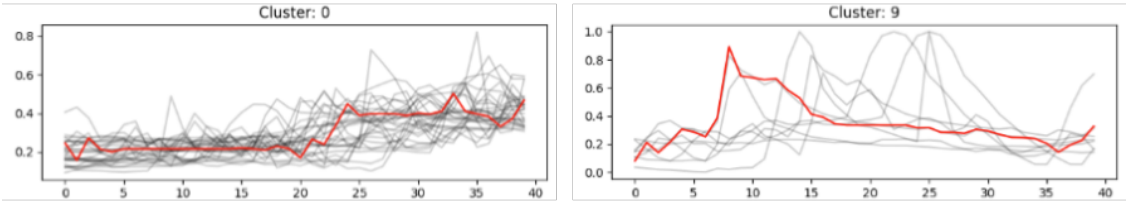


Figure 4.27: Visualization of samples in parent cluster nine.

Overall, the initial clustering managed to separate samples in such a way that a secondary clustering was able to detect a total of eight patterns. Even though eight patterns were discovered, the majority of child clusters were meaningless. We observed that the majority of samples did not conform to a valuable pattern, and only a few distinct patterns were discovered by TSK-Means. It was, however, interesting to examine how the initial clustering seemingly did not capture any patterns, while the secondary clustering was, even across child clusters.

4.6 Comparison

Unfortunately, there was no abundance of clustering algorithms designed purely for time-series data. Theoretically, most clustering algorithms could be applied directly on time-series data, treating every point in time as a dimension. This is, however, neglecting of the natural

dynamics of time-series, such as time shifting, speed, and length. TSK-Means using DTW was one of the algorithms suitable for time-series clustering, and the results reflected that. TSK-Means discovered multiple patterns, albeit, most clusters contained largely noise, and no distinct patterns. This indicated that TSK-Means algorithm was suitable to look for patterns in temperature time-series, but not optimal in this case. It was likely due to a combination of an sub-optimal K and limited diversity in the data.

Apart form TSK-Means, each of the three algorithms were only able to capture a few different patterns. K-Means discovered four patterns, two for PCA DRD and two for TSA DRD. Neither DBSCAN or AHC found any particular patterns. There are some *known unknowns* we expected to identify patterns from. This included; food delivery, faulty refrigerators, high customer traffic, refrigerator maintenance, door ajar, and more. These are scenarios that we know will occur, and scenarios that likely would leave data with information from which distinguishable pattern could be extracted. Despite the lack of distinct patterns detected by the K-Means algorithm, it did find four patterns. While still being less that the eight patterns found by TSK-Means, it is less computationally complex, hence more efficient, which is valuable in itself.

To conclude, we expected each of the algorithms to form clusters and discover patterns aligned with the known unknowns. In other words, we expected more distinct patterns to be observed in the results. Unambiguously, the results indicate that conducting clustering on DRD was not optimal, though still useful, for the provided data for this thesis. We believe, with high certainty, the reason for this was due to the combination of poor diversity in the initial data, and the loss of important details due to DRMs. Regardless, clustering on the DRDs indicated that it was possible to detect patterns despite such a significant data reduction.

Chapter 5

Discussion and Summary

In this chapter, we evaluate and discuss the work, results and future work of this thesis. Along with a time-series specific clustering algorithm, three other clustering algorithms were implemented, which was used to cluster data from three DRMs. We argue that one algorithm in particular stands out as the better option for clustering temporal data values, namely TSK-Means, despite being the least efficient.

Initially, we estimated that the size of the provided dataset would suffice for all processing. This included data handling in terms of pre-processing, feature extraction, and dimensionality reduction. Although thorough data handling was performed, we argued that the data was rather insufficient for this scenario, as evidenced by the following results. Little variation in terms of anomalies and recognizable patterns made it hard to produce profitable results.

With a higher quality dataset, there would likely be better pattern discovery, and better overall results. Larger quantity of data, more variation in the data in general, better resolution, and auxiliary data, would improve the quality of the dataset. We will discuss these implications further in this chapter.

We have implemented and tested a variety of dimensionality reduction and clustering methods. For dimensionality reduction we have used the mathematical PCA method, the unsupervised neural network TSA method, and the competitive learning SOM method. This has helped us explore the strengths and weaknesses of multiple different approaches of feature extraction. Hence, an argument can be made that testing other DRMs would yield similar outputs, following the patterns of PCA, TSA, and SOM. Although, this claim can not be verified, without implementing and testing for all types of feature exaction methods. We tested multiple different dimensional sizes for the DRM, and concluded that even for ten dimensions, there was little diversity in the data. In hindsight we could have invested more time into the DRM for higher dimensional output which could have yielded better results for the DRD. That would however, promote more computational complex clustering later on, and increase run time significantly. Higher dimensional DRD could provide more details for the clustering algorithms, but would obscure the visual representation.

Despite inadequate results from DRMs, we observed that the TSA method stood out. This method combined the strengths of LSTM neural networks and the AE neural network in an attempt to successfully extract enough information from a sample to decode it with minimal loss in an unsupervised manner. Although the DRD was not optimal for this thesis, the method was able to successfully encode and decode the time-series, hence proved to be a powerful feature extraction method for time-series, which might be applied in another

use-case with more success.

Furthermore for clustering we have used convex, density based and connectivity based clustering for the K-Means, DBSCAN and AHC respectively. This gave us a total of nine clustering results, not counting the TSK-Means algorithm. Based on these three clustering methods, we can expect that clustering on the DRDs most likely would not perform any better for any other clustering algorithm.

Unambiguously, the results indicate that conducting clustering on DRD was not optimal for finding distinct patterns given the provided data and configurations for this thesis. We speculate that a combination of poor diversity and loss of details in the data due to preprocessing and DRMs might be the cause. As expected TSK-Means utilizing DTW was the better pattern detection option because it is specialized and designed for time-series data. Additionally this method utilized data before DRMs, hence containing more details, which could have improved pattern detection. As mentioned, the TSK-Means algorithm has a higher complexity compared to the other clustering algorithms. Comparing the algorithms based on the amount of discovered patterns alone, is not reasonable. K-Means on both PCA and TSA discovered four patterns, while being significantly less complex than the TSK-Means. These results are promising, and indicate that it was possible to detect patterns in the data given such a comprehensive dimensionality reduction. For accuracy, TSK-Means will be the better option, but for efficiency K-Means using DRMs is promising and interesting for future experimentation.

We did not factor in refrigerator insulation and size. In retrospect, we realized that different refrigerator insulation and volumes would affect how the temperature change over time, and that the effect of one incident would have on the overall temperature could vary substantially from one refrigerator to another. Information regarding the insulation and volume of the refrigerators was not available and thus not possible to accommodate for. With this in mind we expect that the patterns for one scenario might look different for two refrigerators with different insulation and volumes. This factor might have contributed to the lack of distinct patterns discovered by the clustering algorithms.

Throughout this thesis multiple clusters have been formed by the proposed clustering algorithms, most clusters did not conform to any obvious patterns. However, we found some valuable patterns. These patterns are helpful regarding classification and deviation management, but also provided valuable insight into the operation of refrigerators.

We believe our results suffers as the quality of the data we gathered for the thesis were less than optimal for the use-case. As we mostly wanted to cluster outliers or irregular behavior, it would suffice to estimate that if the amount of data is increased, it will produce a proportional amount of abnormal data compared to normal behavior of the refrigerator. We believe that an increase in abnormal patterns would have made them more easily identifiable by the clustering algorithms. Irregular time intervals between measurements of the data proved a challenge as the algorithms require an even distribution of data point. This could have been amended if the data had a higher resolution, as it would have allowed combining multiple of the data points into larger time gaps, compared to the loss from interpolation. Interpolating the data means sacrificing accuracy and missing out on peaks and troughs if said peaks or troughs are outside the range of the resulting interpolation. We believe that in addition to more raw temperature data, auxiliary data could provide better results. Auxiliary metadata such as the total volume of the refrigerator, sensor location, type, usage, outside temperature, etc.

Altogether, TSK-Means proved to be the better solution when it comes to detecting the

most amounts of patterns in the data, though this option is less efficient than the other proposed methods. K-Means and DRMs indicated the possibility to discover patterns despite a significant reduction in complexity and dimensionality. We believe that given more time for experimentation, more diversity in the dataset, auxiliary data, and experience, DRMs could be an effective way of optimizing clustering of time-series data.

Chapter 6

Conclusions

Throughout this thesis, we have researched, implemented, and evaluated dimensionality reduction methods and clustering for temperature time-series data. Three DRMs and three clustering algorithms were explored, as well as the TSK-Means algorithm. The thesis' product development goals were to determine whether simplification of the clustering process of the otherwise complex temporal data could aid in pattern detection, and additionally find patterns helpful for deviation management. From a business perspective, it was important to detect and distinguish abnormal refrigerator temperature patterns for economic purposes due to food loss, and for deviation management and prevention. However, one limitation was the shortage of clustering algorithms suitable for clustering time-series data. TSK-Means was the algorithm used in this thesis for time-series clustering, but it is time-intensive due to its computational complexity. Therefore, it was interesting to examine methods of aiding the clustering process, by simplifying the data through feature extraction and use of other clustering algorithms.

Overall the results indicate that TSK-Means is the most suitable algorithm for clustering temperature time-series data. We experienced that the proposed methods are less computationally complex and time-intensive, and provided some interesting results. However, the methods could not compete with TSK-Means in detecting patterns in the data. The patterns detected provided a valuable insight into the behavior of the data, which Bitmesh AS can apply to help with mainly deviation management. We expect that a more extensive and diverse dataset with higher resolution would provide improved feature extraction and pattern recognition. The results are promising, indicating that there are recognizable patterns and that the TSK-Means can detect them. The results also indicate that, while not optimal in this case, dimensionality reduction can help with the complexity of clustering time-series on the cost of accuracy. Although we did not find many patterns, we are satisfied with the detected patterns, and the results regarding the applicability of the proposed methods.

Bibliography

[1] O. A. Abbas. "Comparison Between Data Clustering Algorithms." In: (2008).

[2] H. Abdi and L. J. Williams. *Principal Component Analysis*. John Wiley & Sons, Inc, 2010.

[3] A. Amidon. *How to Apply K-means Clustering to Time Series Data*. URL: https://towardsdatascience.com/how-to-apply-k-means-clustering-to-time-series-data-28d04a8f7da3.

[4] bitmeshA AS. *Bitmesh*. URL: https://bitmesh.no/.

[5] P. Bholowalia and A. Kumar. "EBK-Means: A Clustering Technique based on Elbow Method and K-Means in WSN." In: *International Journal of Computer Applications* (2014).

[6] J. Brownlee. *10 Clustering Algorithms With Python*. URL: https://machinelearningmastery.com/clustering-algorithms-with-python/.

[7] C. Chatfield. *The Analysis of Time Series: Theory And Practice*. Chapman and Hall, 1975.

[8] P. J. Davis. *Interpolation and approximation*. Courier Corporation, 1975.

[9] M. Ester et al. "A Density-Based Algorithm for Discovering Clusters." In: *AAAI* (1996).

[10] Z. Ghahramani. "Unsupervised learning." In: *Summer school on machine learning*. Springer. 2003, pp. 72–112.

[11] G. Hamerly and C. Elkan. "Learning the K in K-means." In: *NIPS* (2003). DOI: 92093-0114.

[12] W. Hannes and S. K. Ashenden. *The Era of Artificial Intelligence, Machine Learning, and Data Science in the Pharmaceutical Industry*. 2021. ISBN: 978-0-12-820045-2.

[13] X. Huang et al. "Time series k-means: A new k-means type smooth subspace clustering for time series data." In: *Elsevier* (2016).

[14] IBM. *Recurrent Neural Networks*. URL: https://www.ibm.com/cloud/learn/recurrent-neural-networks.

[15] IBM. *Unsupervised Learning*. URL: https://www.ibm.com/cloud/learn/unsupervised-learning.

[16] IntechOpen. "Data Mining - Methods, Applications and Systems." In: IntechOpen, 2021. Chap. 2.3.

[17] W. Kenton. *Interpolation*. URL: https://www.investopedia.com/terms/i/interpolation.asp.

[18] E. Keogh and J. Lin. "Clustering of time-series subsequences is meaningless: implications for previous and future research." In: *Knowledge and Information System* (2004). DOI: 10.1007/s10115-004-0172-7.

[19] K. Khan et al. "DBSCAN: Past, present and future." In: *The fifth international conference on the applications of digital information and web technologies (ICADIWT 2014)*. IEEE. 2014, pp. 232–238.

[20] T. Kohonen. "The self-organizing map." In: *Neurocomputing* 21 (1998), pp. 1–6. URL: https://www.sciencedirect.com/science/article/pii/S0925231298000307.

[21] Scikit Learn. *KMeans*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html.

[22] Scikit Learn. *PCA*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html.

[23] Scikit Learn. *sklearn.lda.LDA*. URL: https://scikit-learn.org/0.16/modules/generated/sklearn.lda.LDA.html.

[24] Scikit Learn. *sklearn.preprocessing.MinMaxScaler*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html.

[25] Mattilsynet. *Internkontroll*. URL: https://www.mattilsynet.no/mat_og_vann/Ny_matbedrift/startpakke/internkontroll.35152.

[26] S. Misra, H. Li, and J. He. *Machine Learning for Subsurface Characterization*. 2020. ISBN: 978-0-12-817736-5.

[27] A. Ng. "Sparse Autoencoder." In: *CS294A Lecture notes* 72.2011 (2011), pp. 1–19. URL: https://web.stanford.edu/class/cs294a/sparseAutoencoder_2011new.pdf.

[28] L. H. Nguyen and S. Holmes. "Ten quick tips for effective dimensionality reduction." In: *PLOS Computational Biology* (2019). DOI: 10.1371/journal.pcbi.1006907.

[29] F. Nielsen. "Introduction to HPC with MPI for Data Science." In: Springer, Cham, 2016. Chap. 8.

[30] C. Olah. *Understanding LSTM Networks*. URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[31] S. G. K. Patro and K. K. Sahu. "Normalization: A preprocessing stage." In: *arXiv preprint arXiv:1503.06462* (2015).

[32] M. Peixeiro. *The complete guide to time series*. URL: https://towardsdatascience.com/the-complete-guide-to-time-series-analysis-and-forecasting-70d476bfe775.

[33] A. Ralhan. *Self Organizing Maps*. URL: https://medium.com/@abhinavr8/self-organizing-maps-ff5853a118d4.

[34] P. J. Rousseeuw. "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis." In: *Journal of Computational and Applied Mathematics* 20 (1987), pp. 53–65. DOI: https://doi.org/10.1016/0377-0427(87)90125-7.

[35] scikit-learn. *2.3. Clustering*. URL: https://scikit-learn.org/stable/modules/clustering.html#clustering-performance-evaluation.

[36] P. Senin. "Dynamic Time Warping Algorithm Review." In: (2008).

[37] K. P Sinaga and M. S. Yang. "Unsupervised K-Means Clustering Algorithm." In: (2020). DOI: 10.1109/ACCESS.2020.2988796.

[38] R. Smith. *sklearn-som v. 1.1.0 Master Documentation*. URL: https://sklearn-som.readthedocs.io/en/latest/.

[39] Distruptive Technologies. *Temperature Sensor View*. URL: https://www.disruptive-technologies.com/hubfs/Temperature_Sensor_views.png.

[40] Distruptive Technologies. *Wireless Temperature Sensor QR*. Revision: 102058 0. 2020.

[41] Tensorflow. *Keras: The Python Deep Learning API*. URL: https://keras.io/.

[42] E. Tiu. *Understanding Latent Space in Machine Learning*. URL: https://towardsdatascience.com/understanding-latent-space-in-machine-learning-de5a7c687d8d.

[43] X. Wang and Y. Xu. "An improved index for clustering validation based on Silhouette index and Calinski-Harabasz index." In: *IOP Conference Series: Materials Science and Engineering*. Vol. 569. 5. IOP Publishing. 2019, p. 052024.

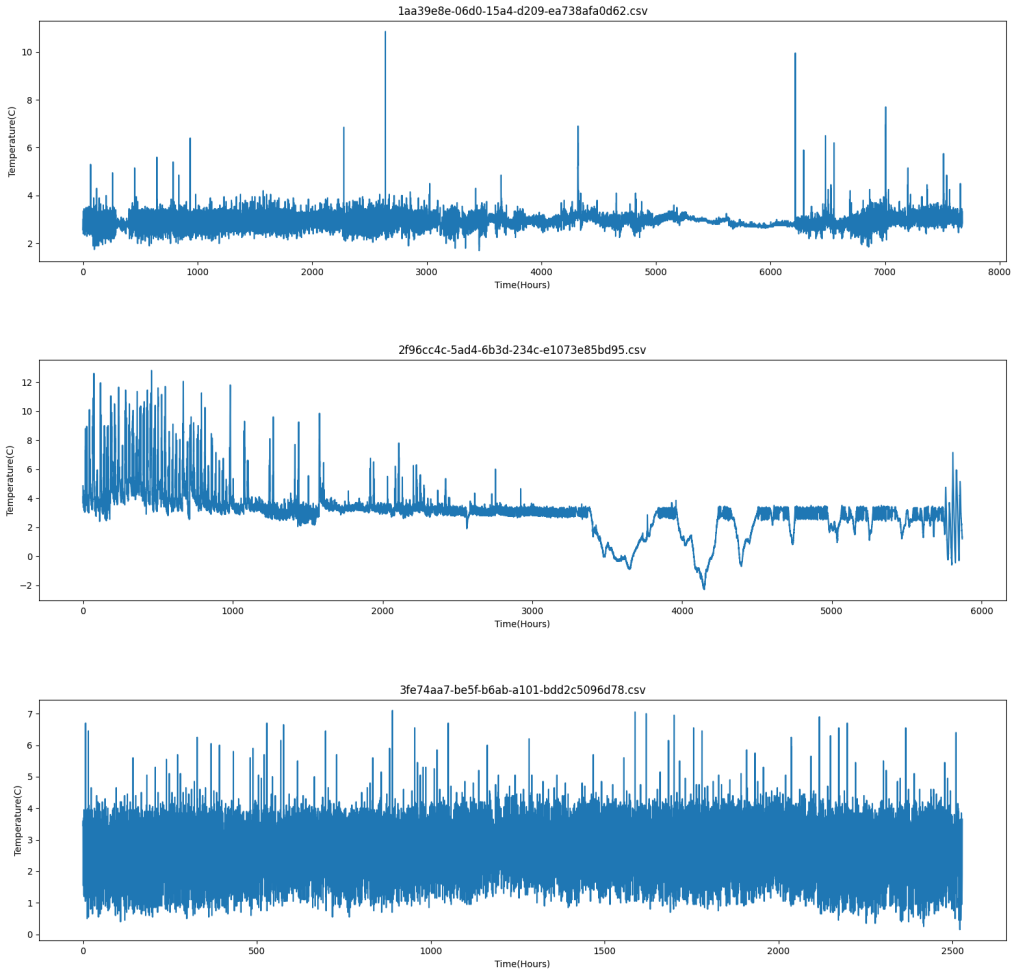
[44] XantaCross. *Euclidean vs DTW*. URL: https://commons.wikimedia.org/wiki/File:Euclidean_vs_DTW.jpg.

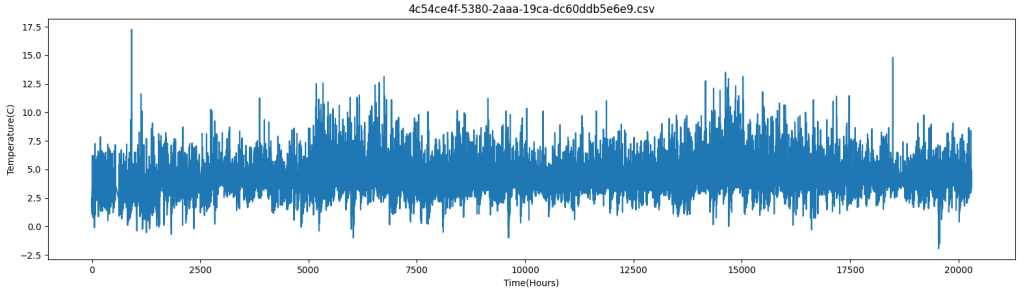
[45] T. Yiu. *The Curse of Dimensionality*. URL: https://towardsdatascience.com/the-curse-of-dimensionality-50dc6e49aa1e.

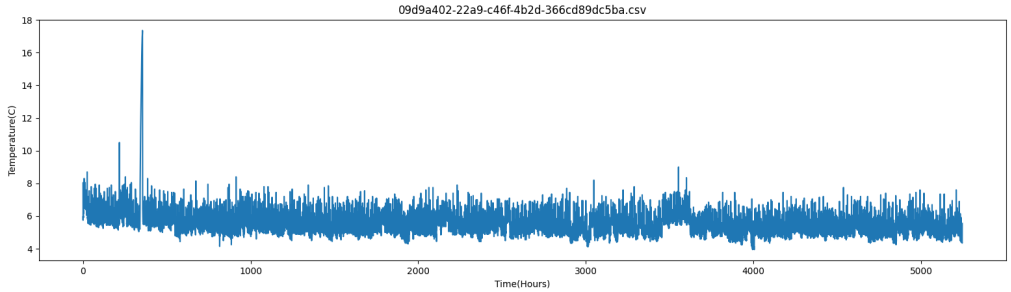
Appendix A

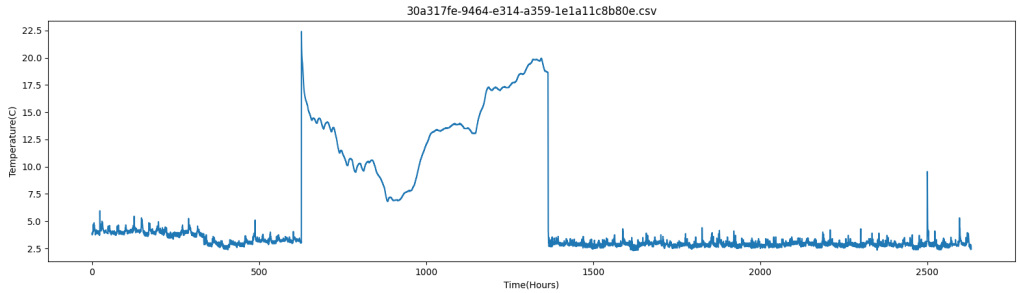
All Signals

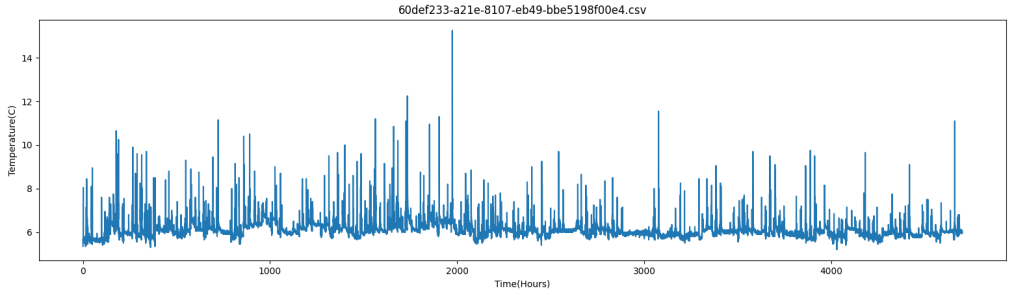
Appendix A presents sensor signals in which data is based upon. Temperature in Celsius and time in hours are shown on the y-axis and x-axis, respectively. Signals are measurements from different refrigerator units, and shows how temperature can fluctuate.

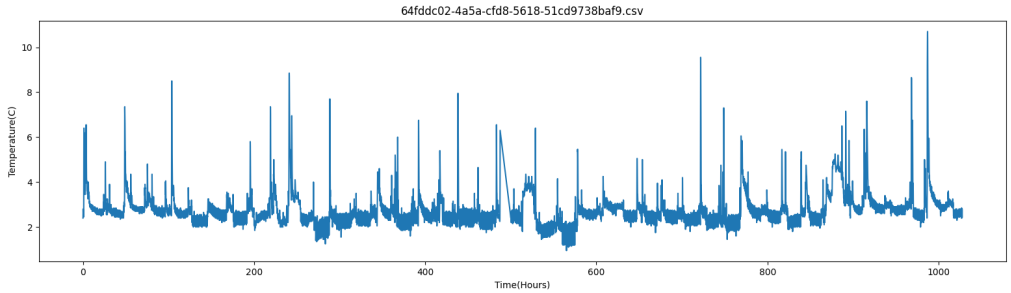


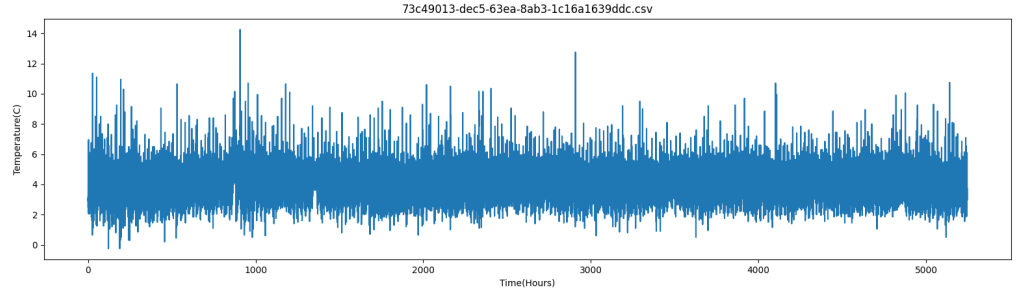

4c54ce4f-5380-2aaa-19ca-dc60ddb5e6e9.csv
Temperature(C)
Time(Hours)

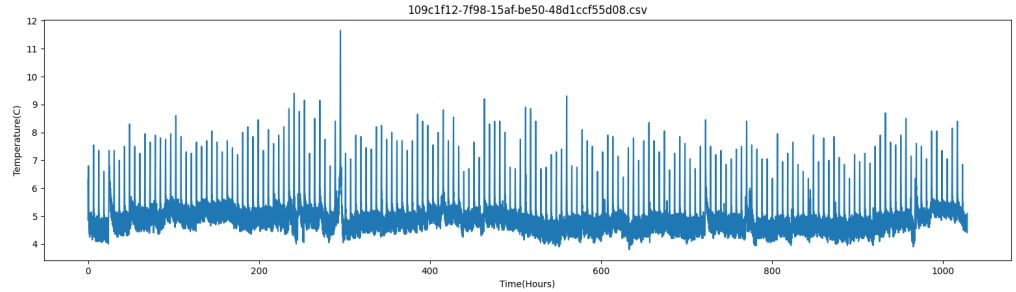

09d9a402-22a9-c46f-4b2d-366cd89dc5ba.csv
Temperature(C)
Time(Hours)

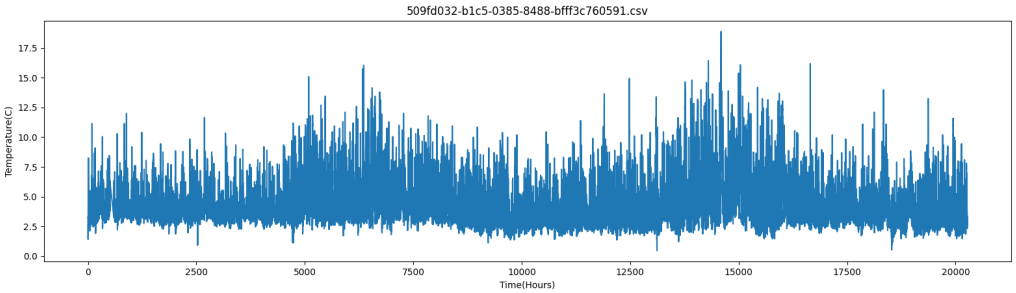

30a317fe-9464-e314-a359-1e1a11c8b80e.csv
Temperature(C)
Time(Hours)

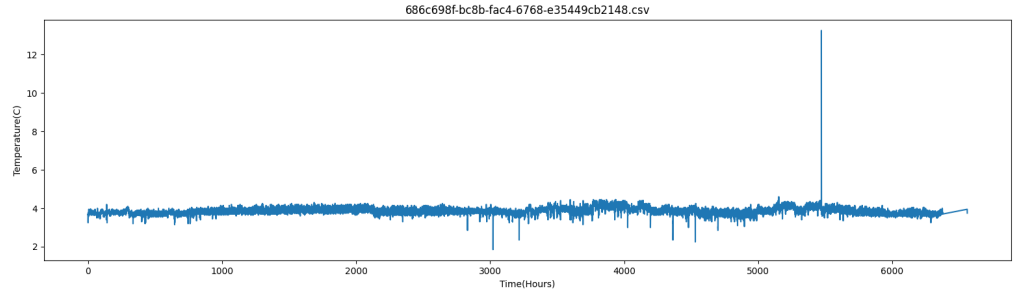

60def233-a21e-8107-eb49-bbe5198f00e4.csv
Temperature(C)
Time(Hours)

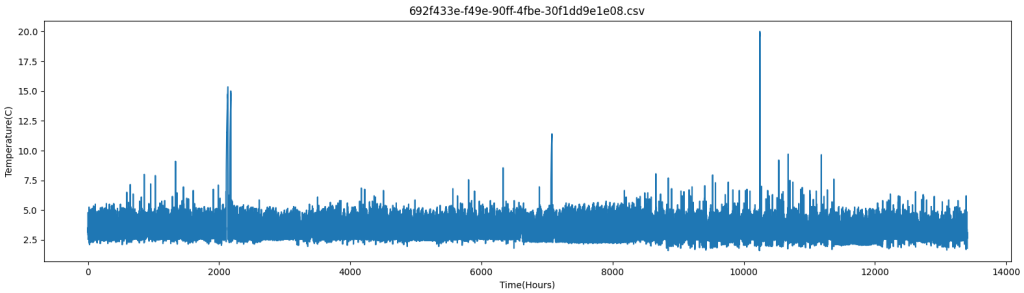

64fddc02-4a5a-cfd8-5618-51cd9738baf9.csv
Temperature(C)
Time(Hours)

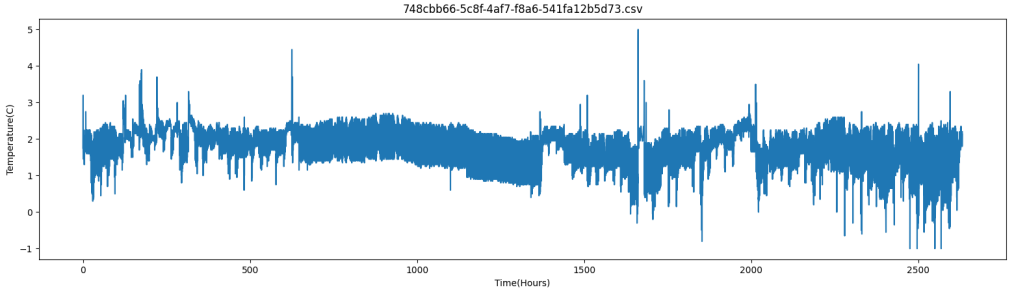

73c49013-dec5-63ea-8ab3-1c16a1639ddc.csv
Temperature(C)
Time(Hours)

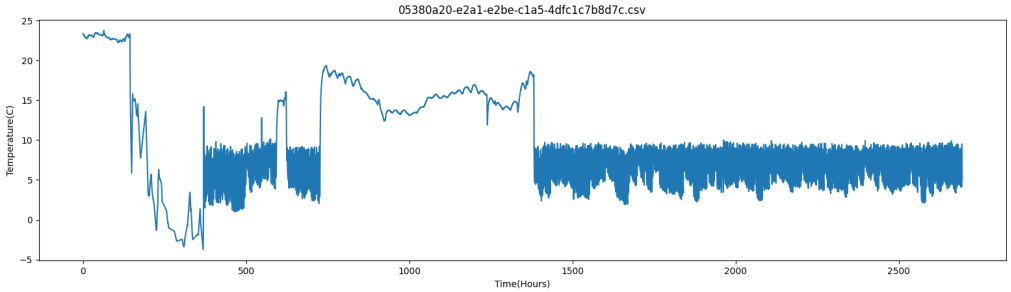

109c1f12-7f98-15af-be50-48d1ccf55d08.csv
Temperature(C)
Time(Hours)

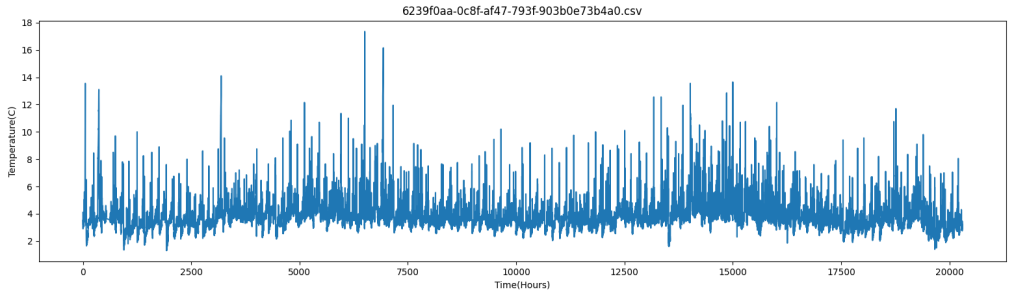

509fd032-b1c5-0385-8488-bfff3c760591.csv
Temperature(C)
Time(Hours)

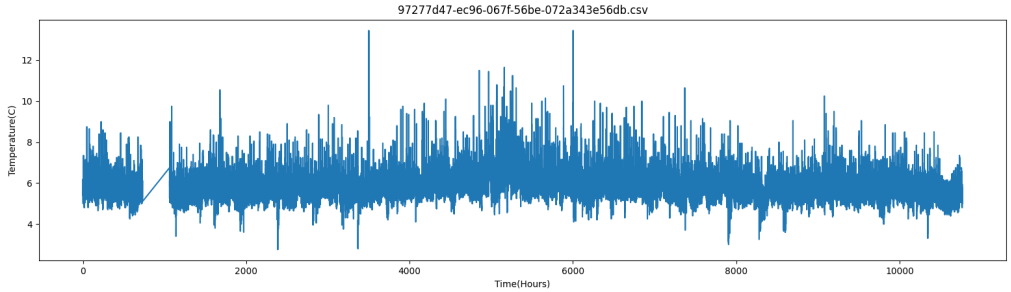

686c698f-bc8b-fac4-6768-e35449cb2148.csv
Temperature(C)
Time(Hours)

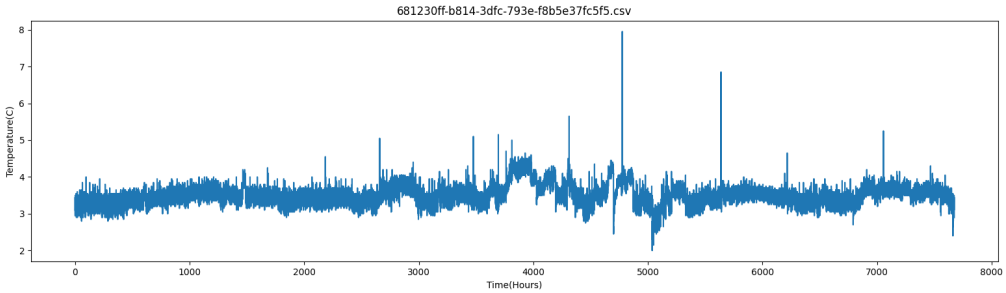

692f433e-f49e-90ff-4fbe-30f1dd9e1e08.csv
Temperature(C)
Time(Hours)

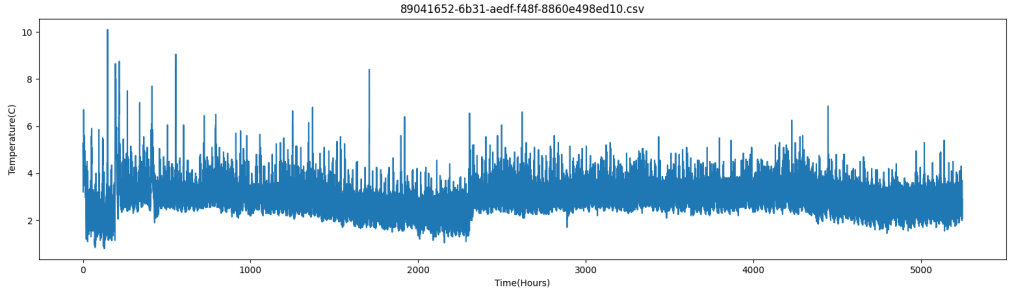

748cbb66-5c8f-4af7-f8a6-541fa12b5d73.csv
Temperature(C)
Time(Hours)

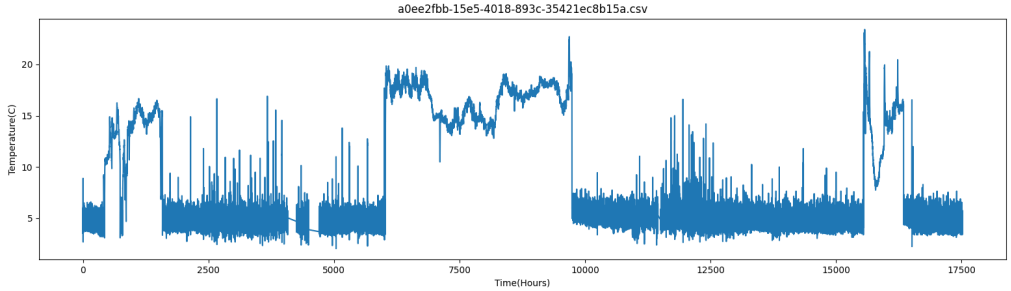

05380a20-e2a1-e2be-c1a5-4dfc1c7b8d7c.csv
Temperature(C)
Time(Hours)

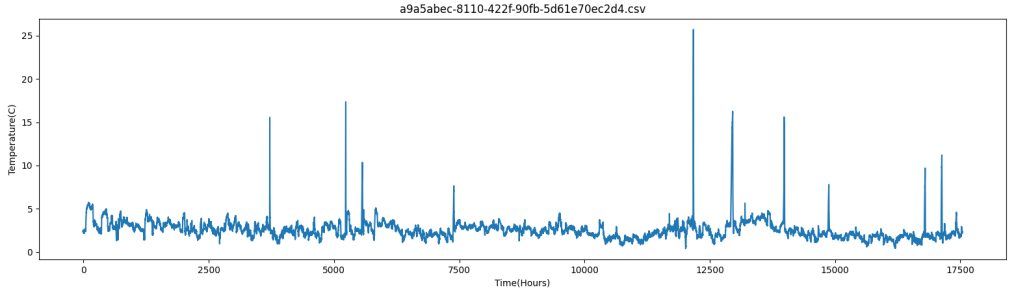

6239f0aa-0c8f-af47-793f-903b0e73b4a0.csv
Temperature(C)
Time(Hours)

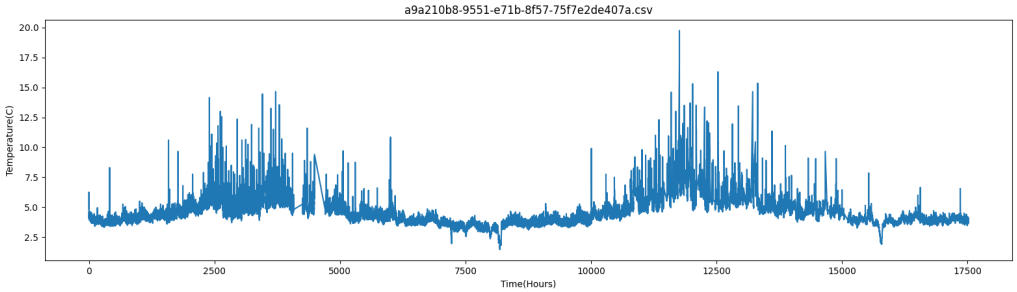

97277d47-ec96-067f-56be-072a343e56db.csv
Temperature(C)
Time(Hours)

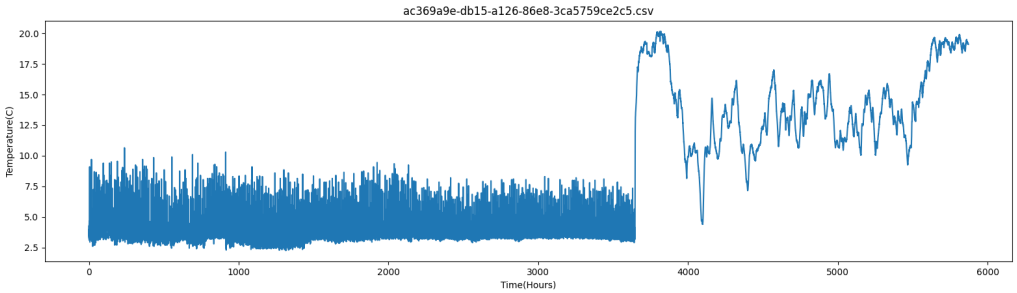

681230ff-b814-3dfc-793e-f8b5e37fc5f5.csv
Temperature(C)
Time(Hours)

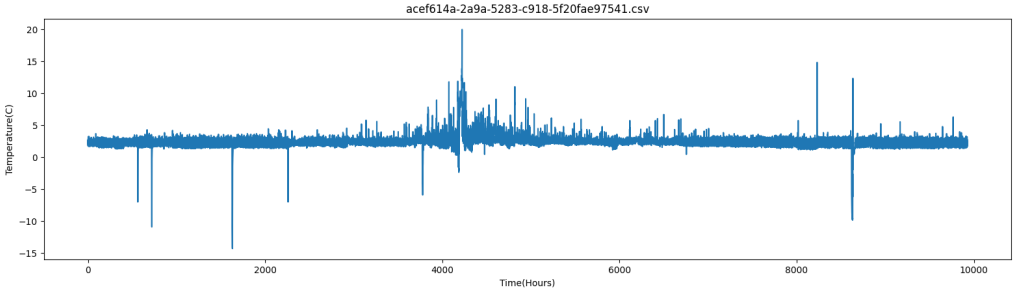

89041652-6b31-aedf-f48f-8860e498ed10.csv
Temperature(C)
Time(Hours)

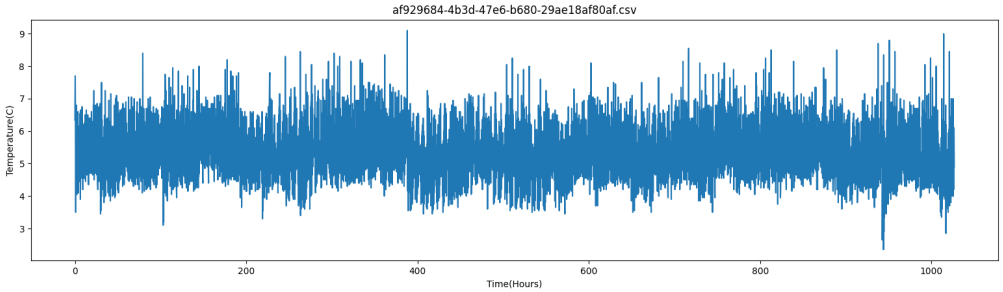

a0ee2fbb-15e5-4018-893c-35421ec8b15a.csv
Temperature(C)
Time(Hours)

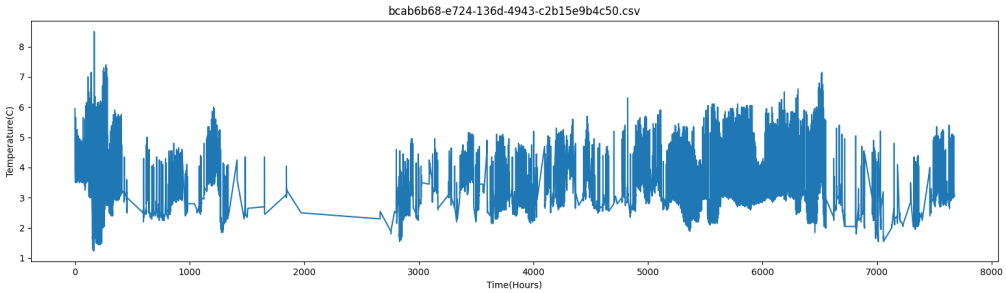

a9a5abec-8110-422f-90fb-5d61e70ec2d4.csv
Temperature(C)
Time(Hours)

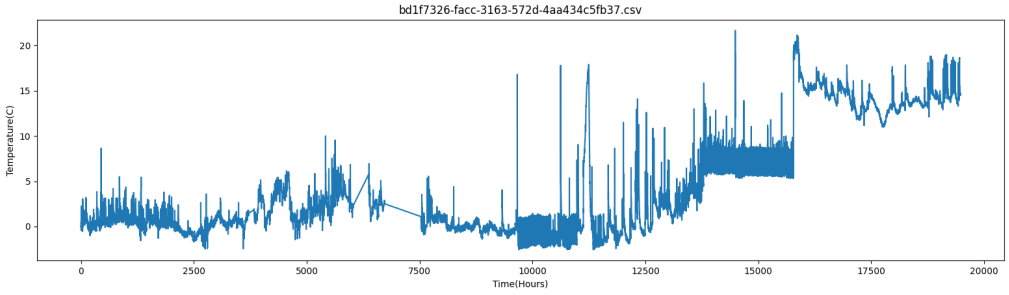

a9a210b8-9551-e71b-8f57-75f7e2de407a.csv
Temperature(C)
Time(Hours)

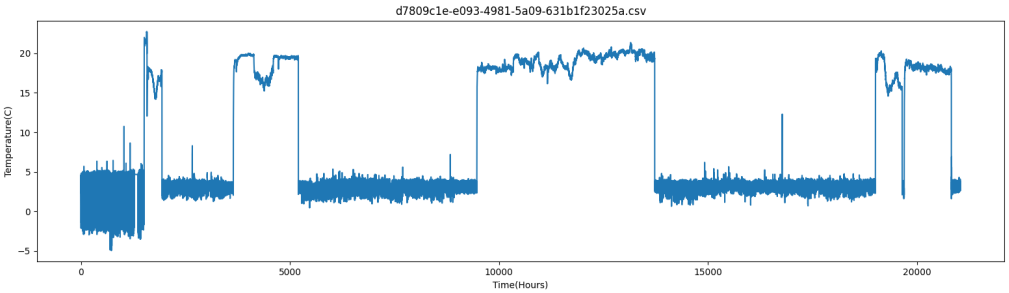

ac369a9e-db15-a126-86e8-3ca5759ce2c5.csv
Temperature(C)
Time(Hours)

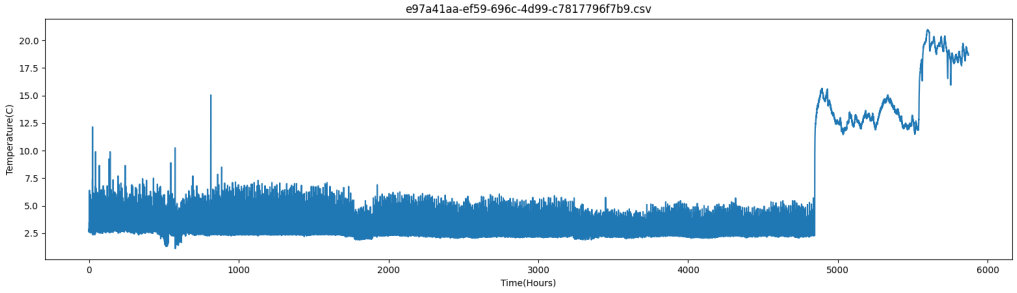

acef614a-2a9a-5283-c918-5f20fae97541.csv
Temperature(C)
Time(Hours)

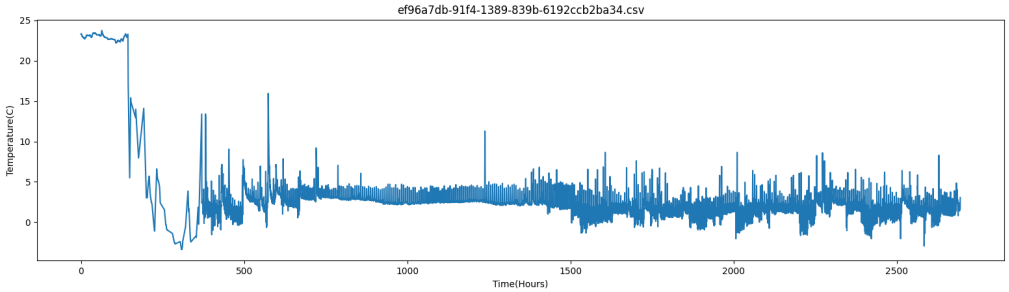

af929684-4b3d-47e6-b680-29ae18af80af.csv
Temperature(C)
Time(Hours)

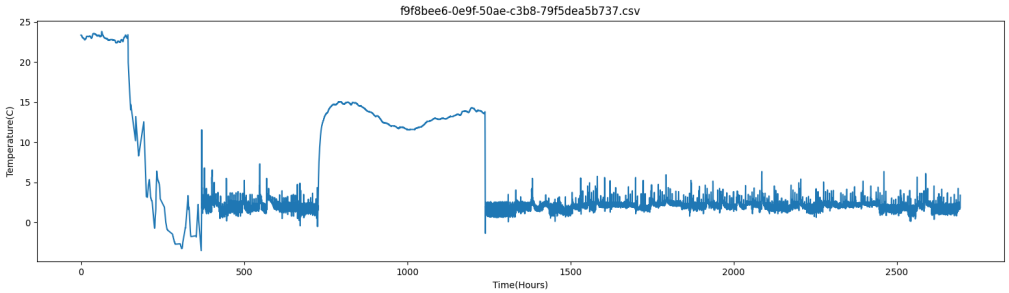

bcab6b68-e724-136d-4943-c2b15e9b4c50.csv
Temperature(C)
Time(Hours)

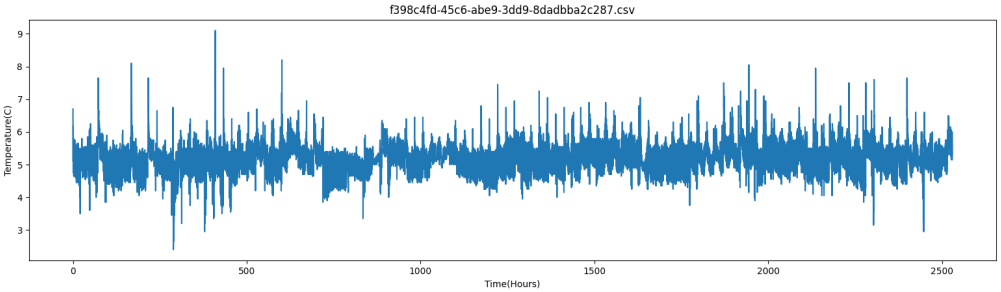

bd1f7326-facc-3163-572d-4aa434c5fb37.csv
Temperature(C)
Time(Hours)

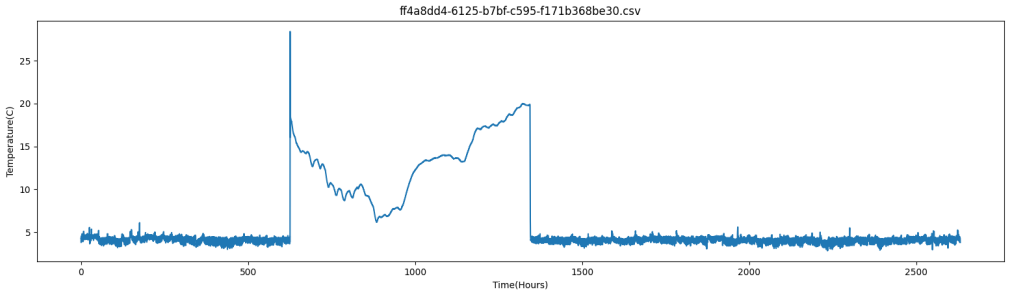

d7809c1e-e093-4981-5a09-631b1f23025a.csv
Temperature(C)
Time(Hours)


e97a41aa-ef59-696c-4d99-c7817796f7b9.csv
Temperature(C)
Time(Hours)


ef96a7db-91f4-1389-839b-6192ccb2ba34.csv
Temperature(C)
Time(Hours)


f9f8bee6-0e9f-50ae-c3b8-79f5dea5b737.csv
Temperature(C)
Time(Hours)


f398c4fd-45c6-abe9-3dd9-8dadbba2c287.csv
Temperature(C)
Time(Hours)


ff4a8dd4-6125-b7bf-c595-f171b368be30.csv
Temperature(C)
Time(Hours)

Appendix B

All Cluster Results

Appendix B presents cluster results formed by the proposed clustering methods.

B.1 Clusters for K-Means

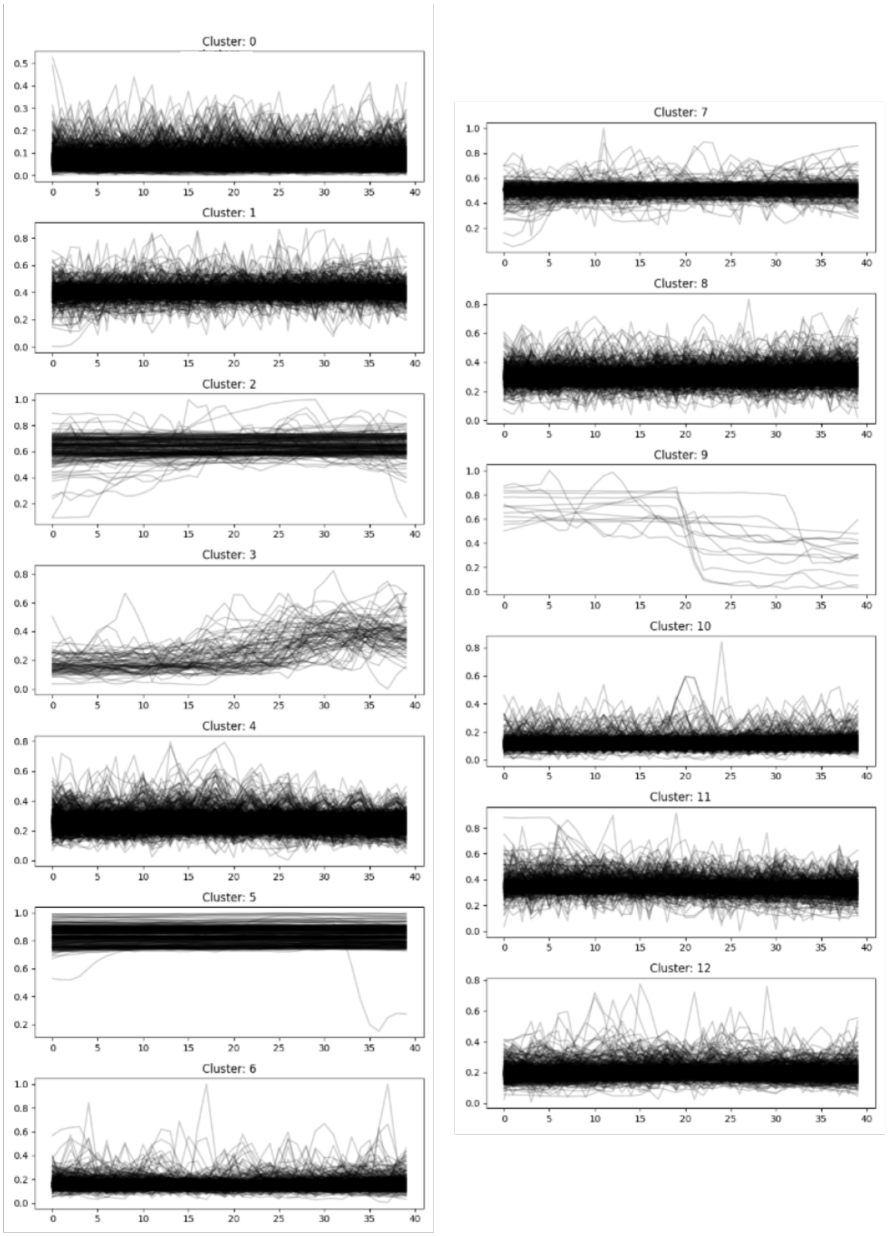


Figure B.1: Clusters for PCA from K-Means.

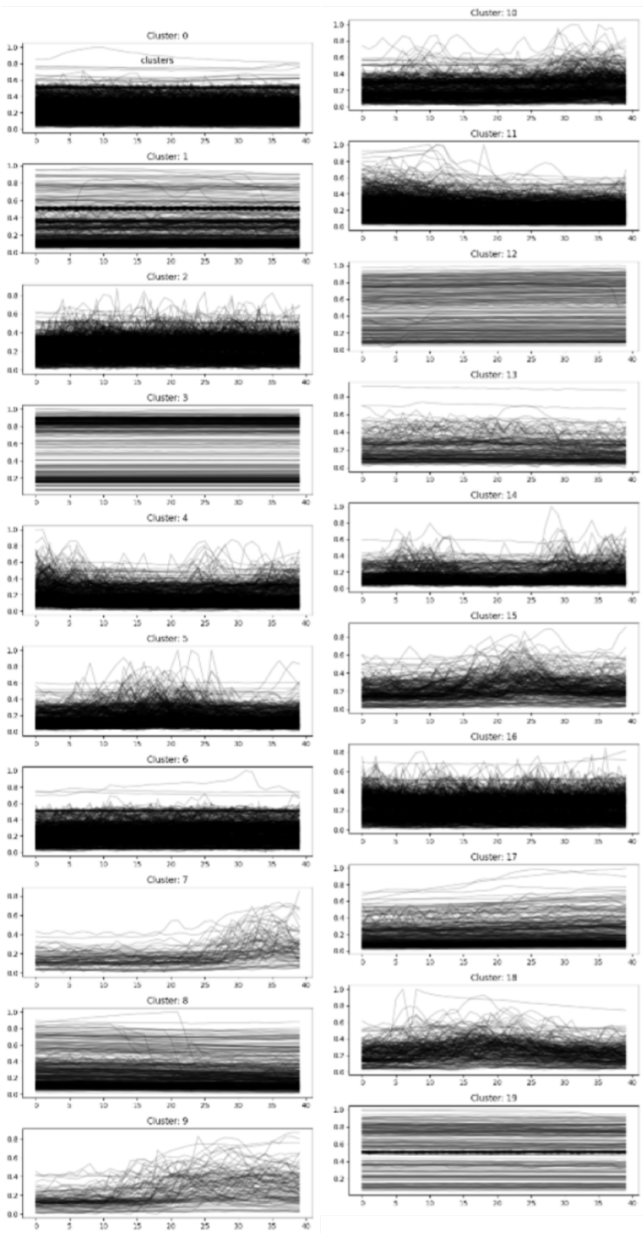


Figure B.2: Clusters for TSA from K-Means.

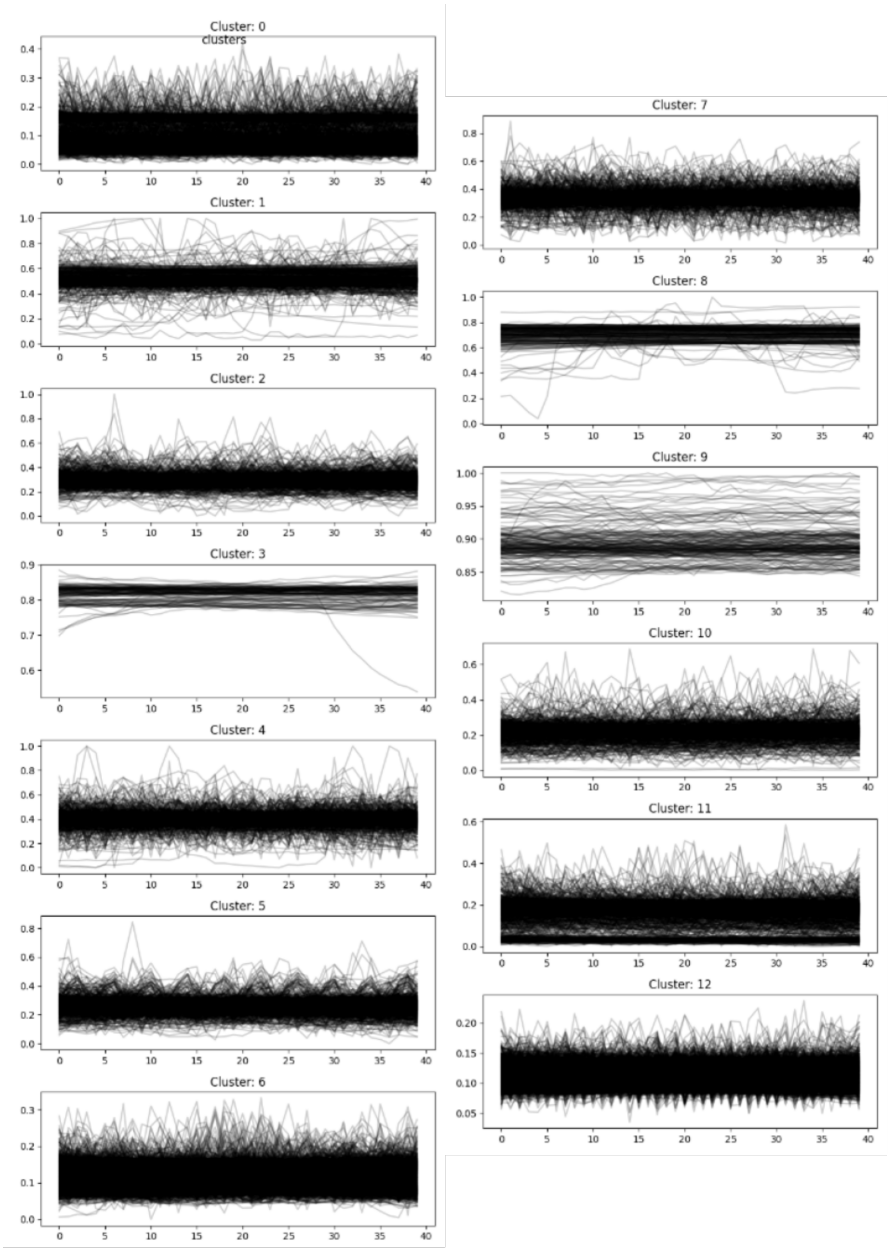


Figure B.3: Clusters for SOM from K-Means.

B.2 Clusters for AHC

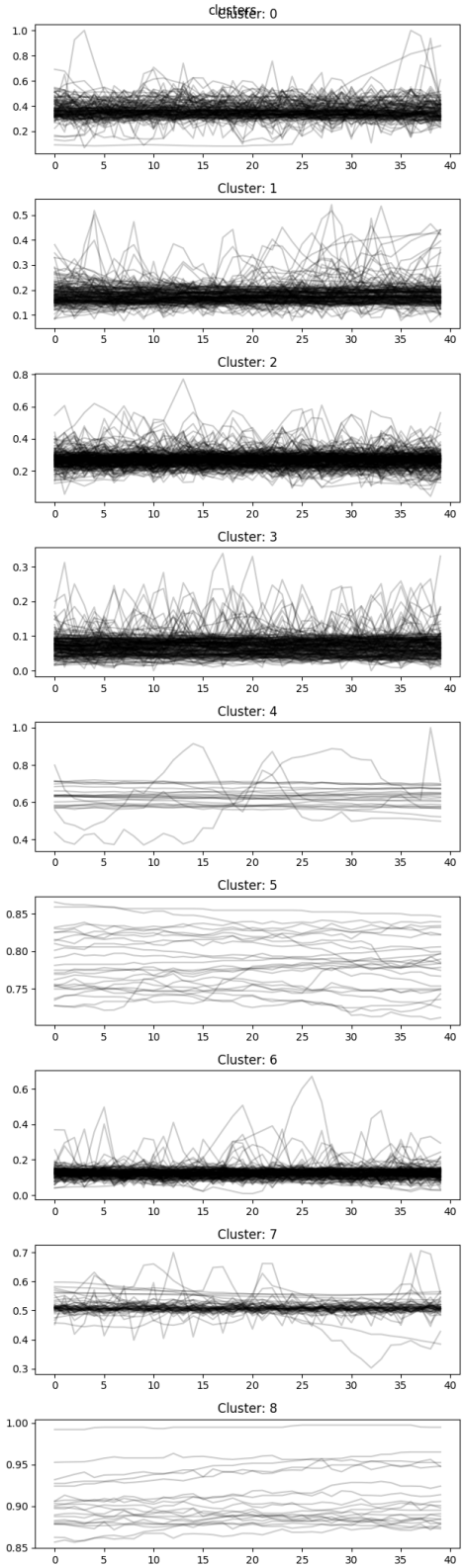


Figure B.4: Clusters for PCA from AHC.

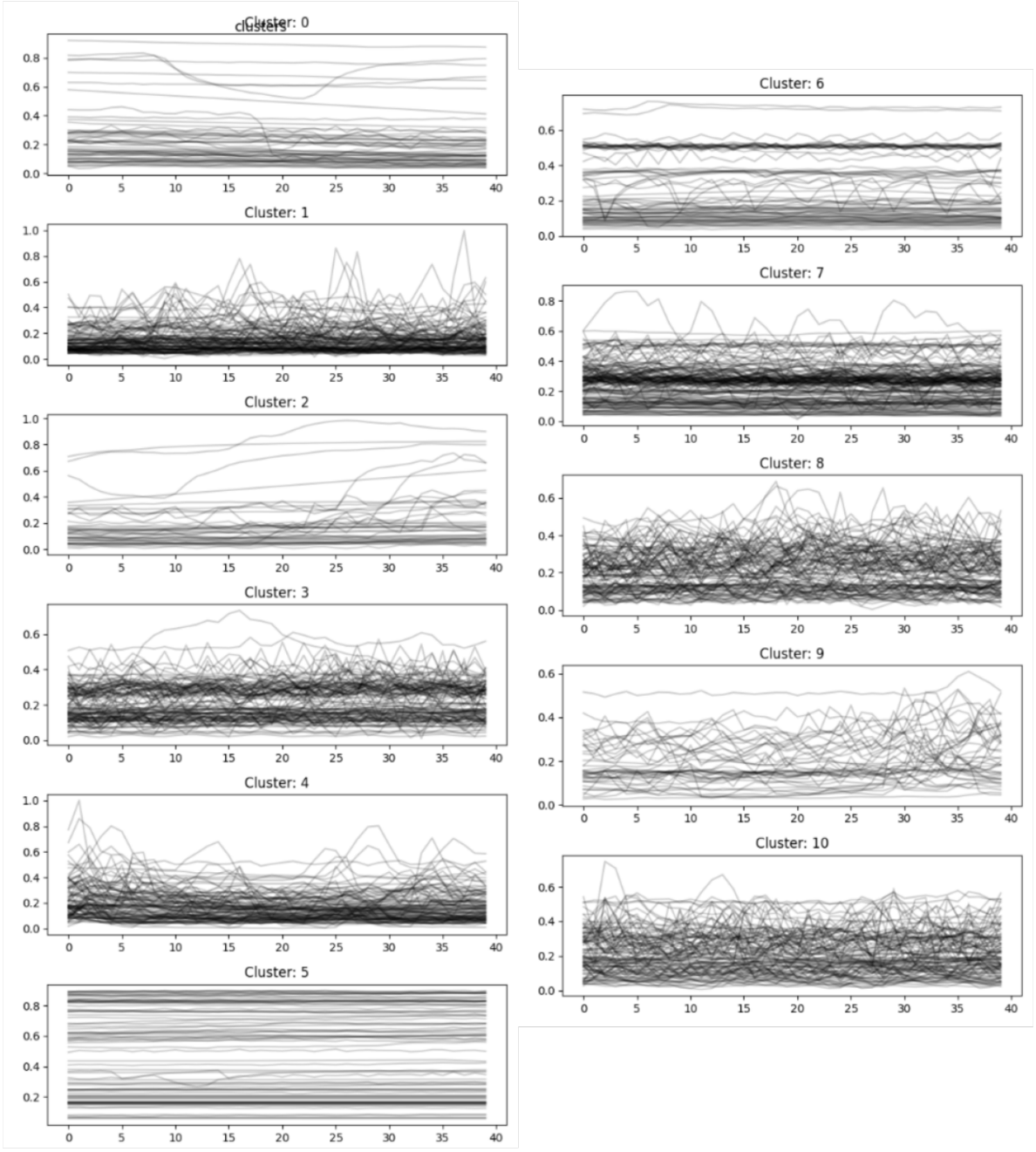


Figure B.5: Clusters for TSA from AHC.

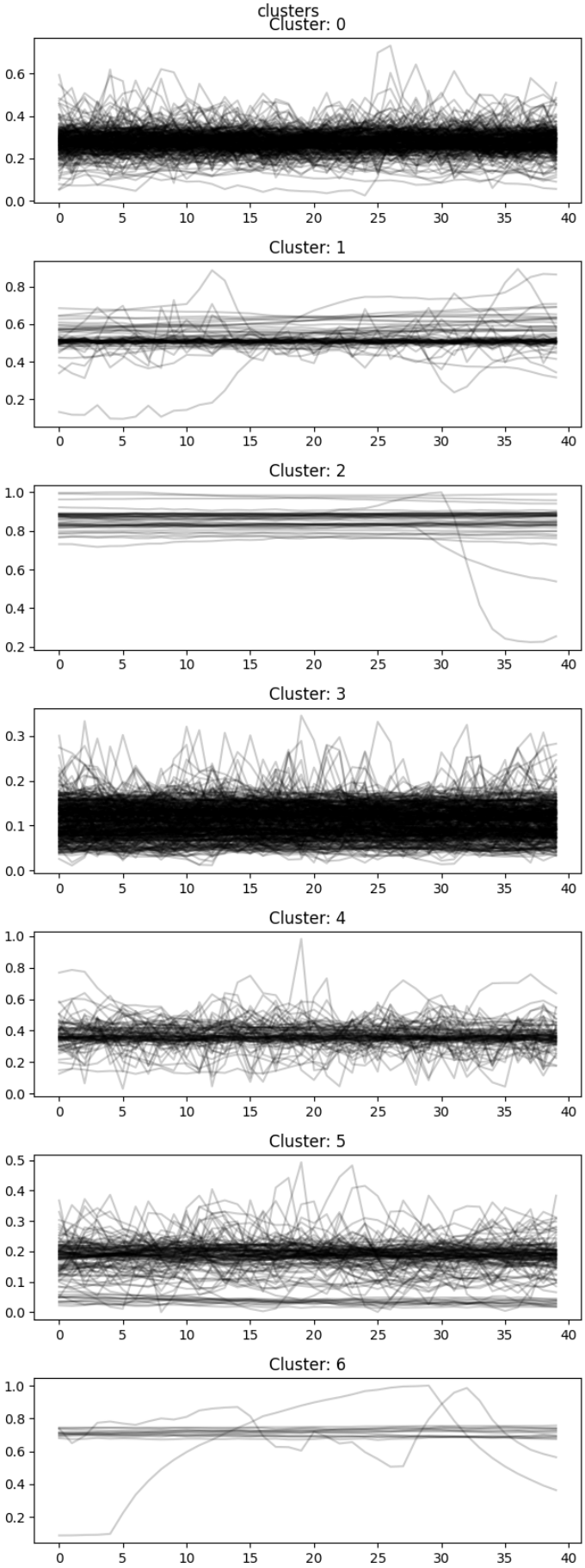


Figure B.6: Clusters for SOM from AHC.

B.3 Clusters for TSK-Means

This section presents the clusters formed by the TSK-Means algorithm. The clustering was conducted in two rounds, where all the data was used to form *ten parent* clusters, then ten more *children clusters* was formed for each parent cluster. Here children clusters will be presented grouped on parent clusters. The black lines are samples and the red line is the barycenter of each cluster.

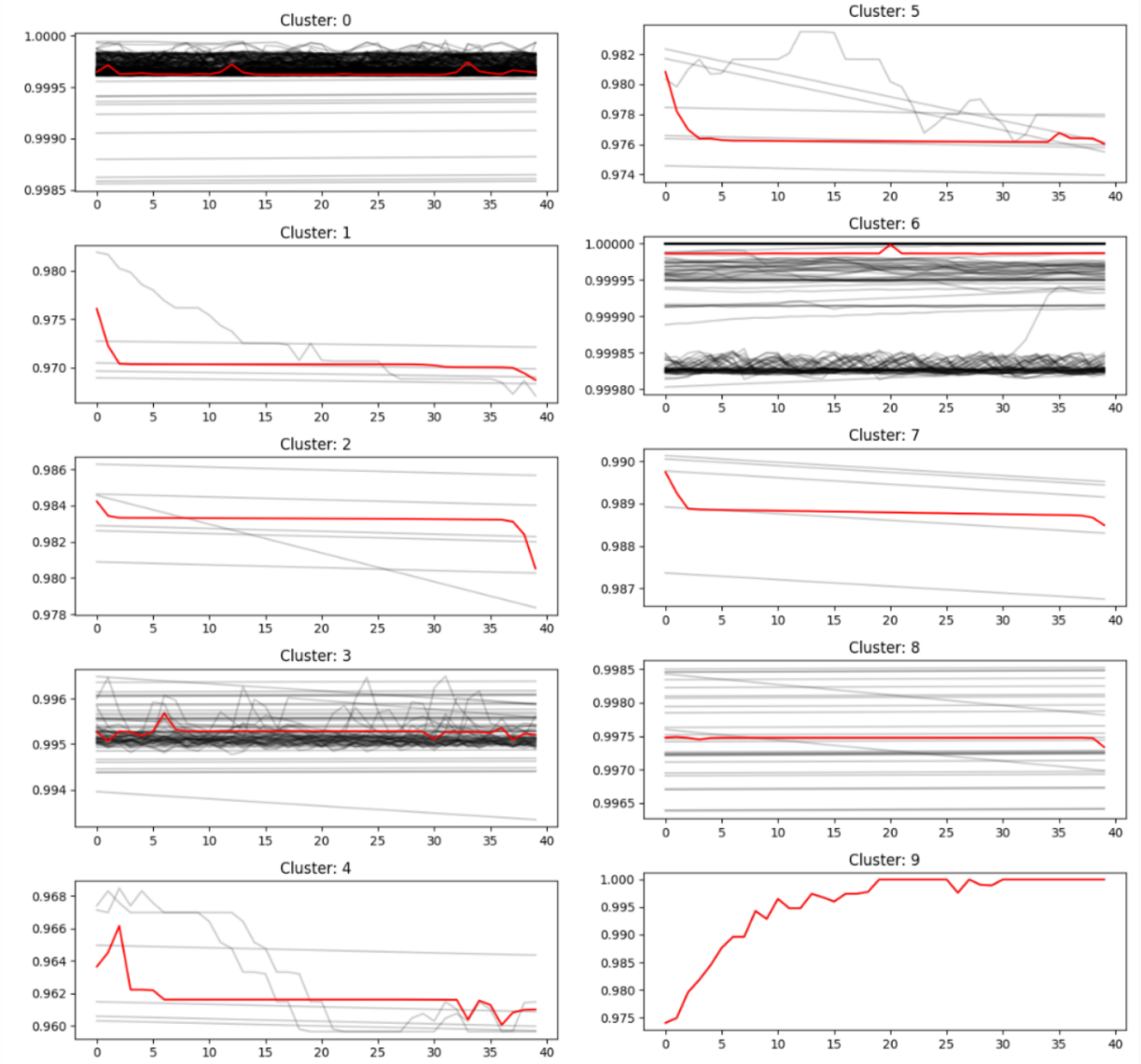


Figure B.7: Child clusters from parent cluster zero from TSK-Means.

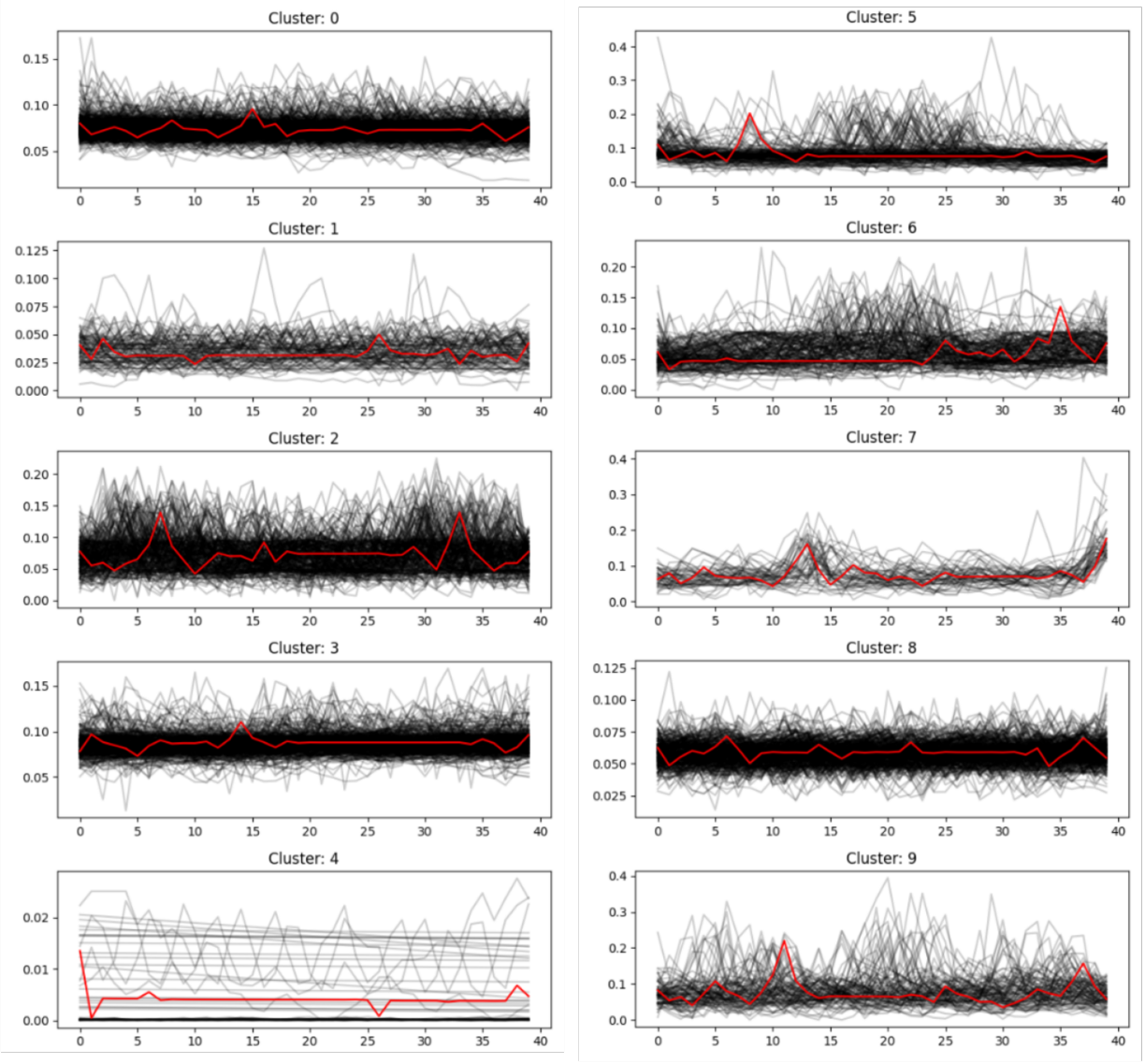


Figure B.8: Child clusters from parent cluster one from TSK-Means.

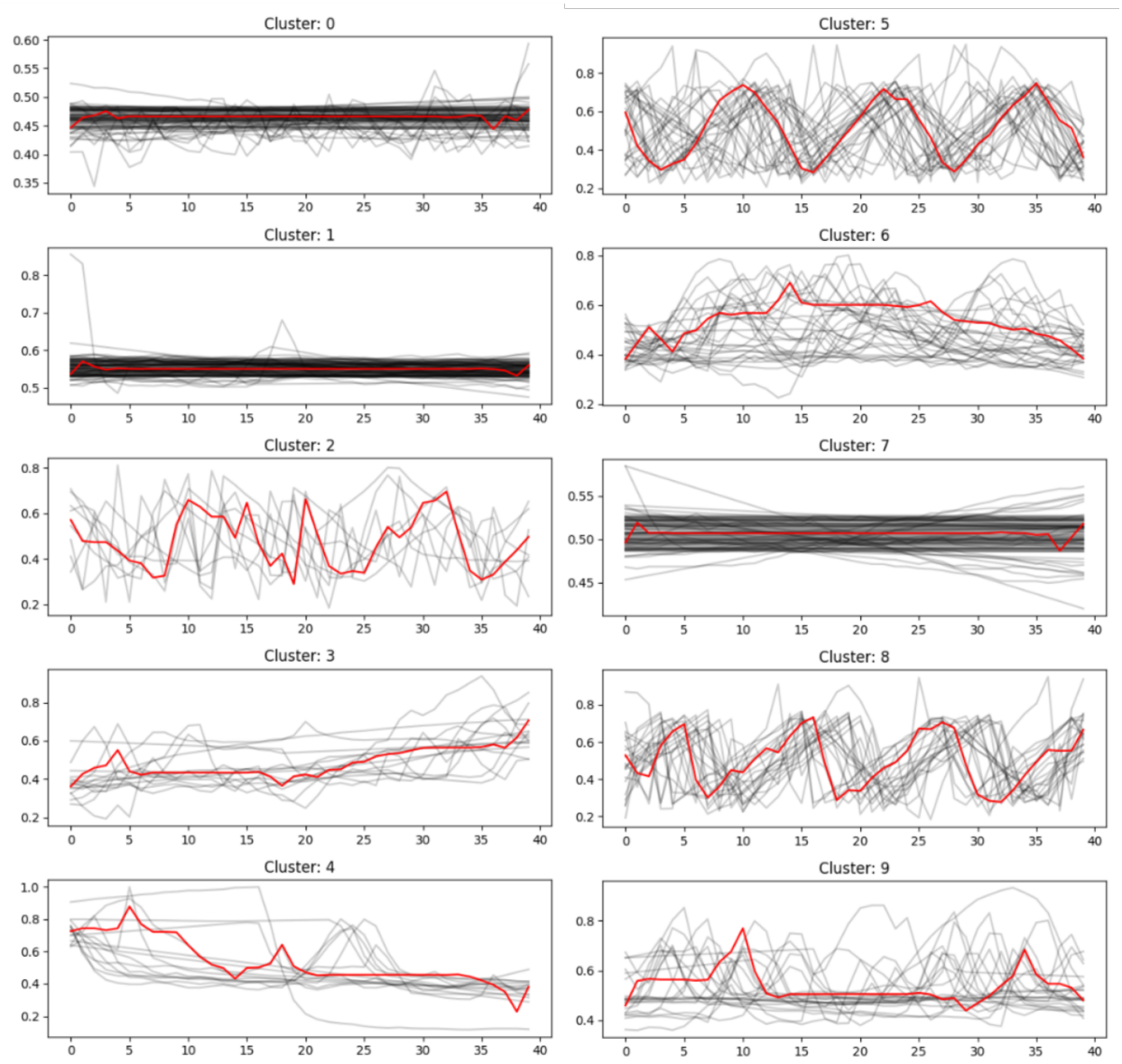


Figure B.9: Child clusters from parent cluster two from TSK-Means.

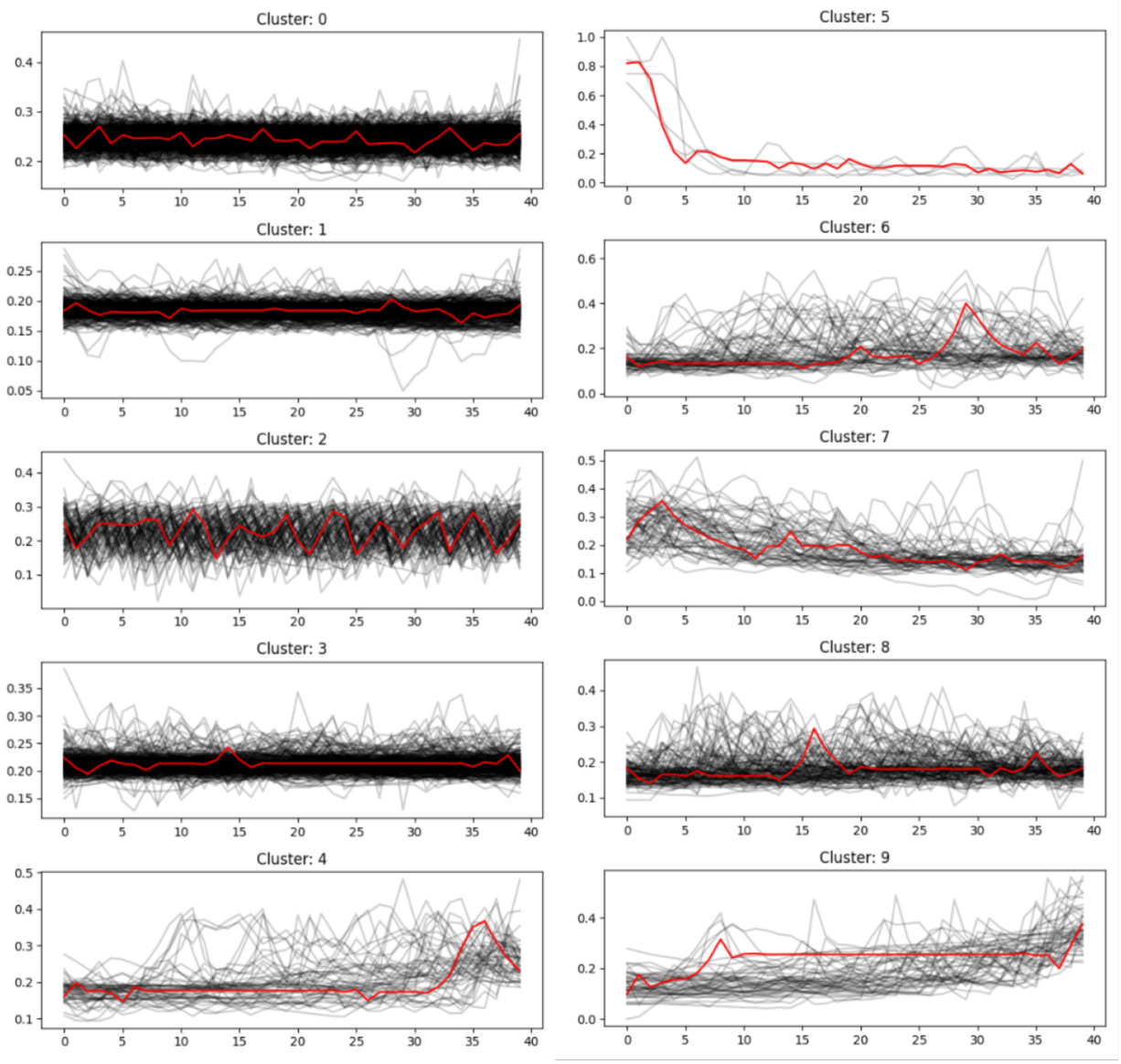


Figure B.10: Child clusters from parent cluster three from TSK-Means.

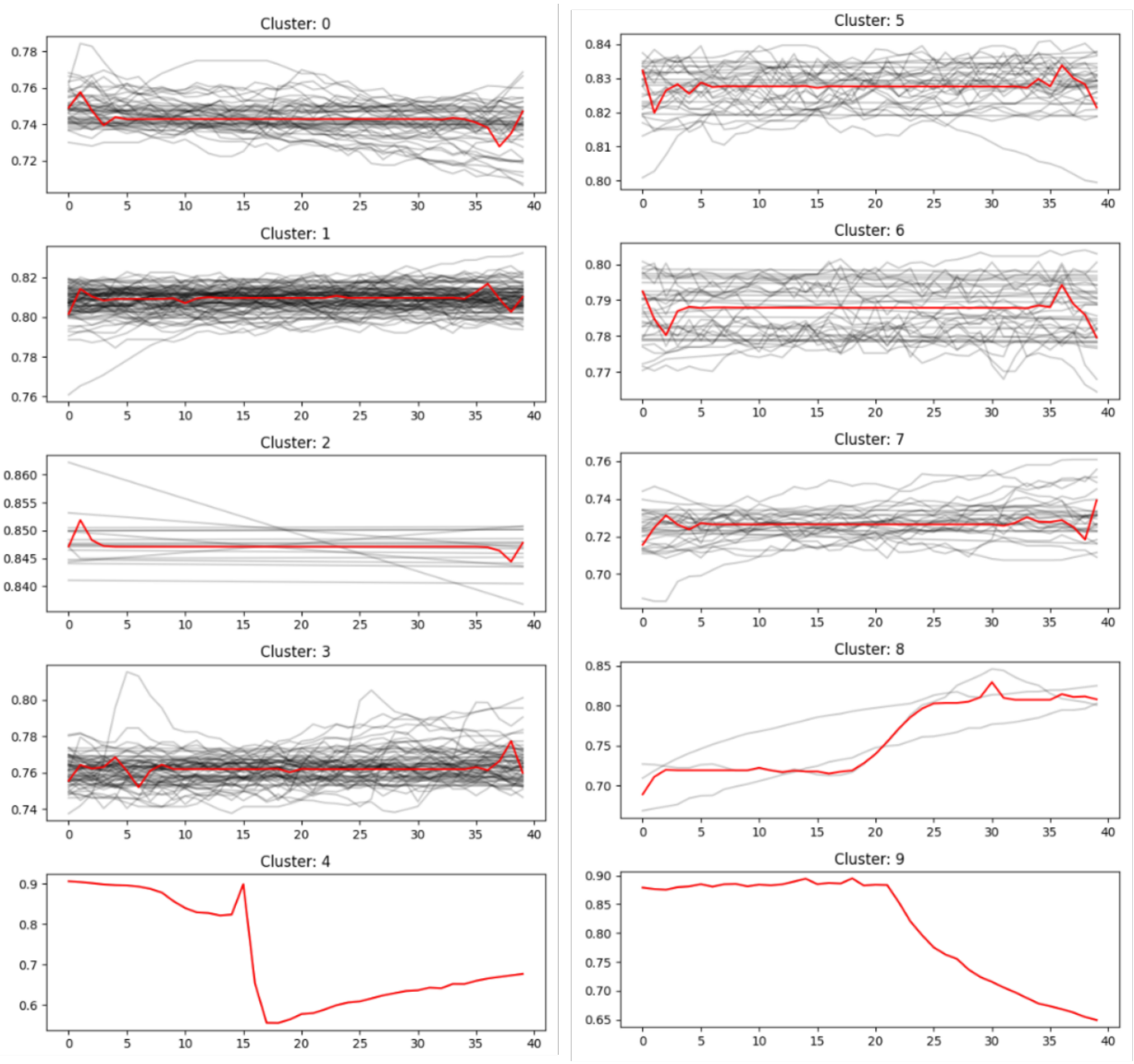


Figure B.11: Child clusters from parent cluster four from TSK-Means.

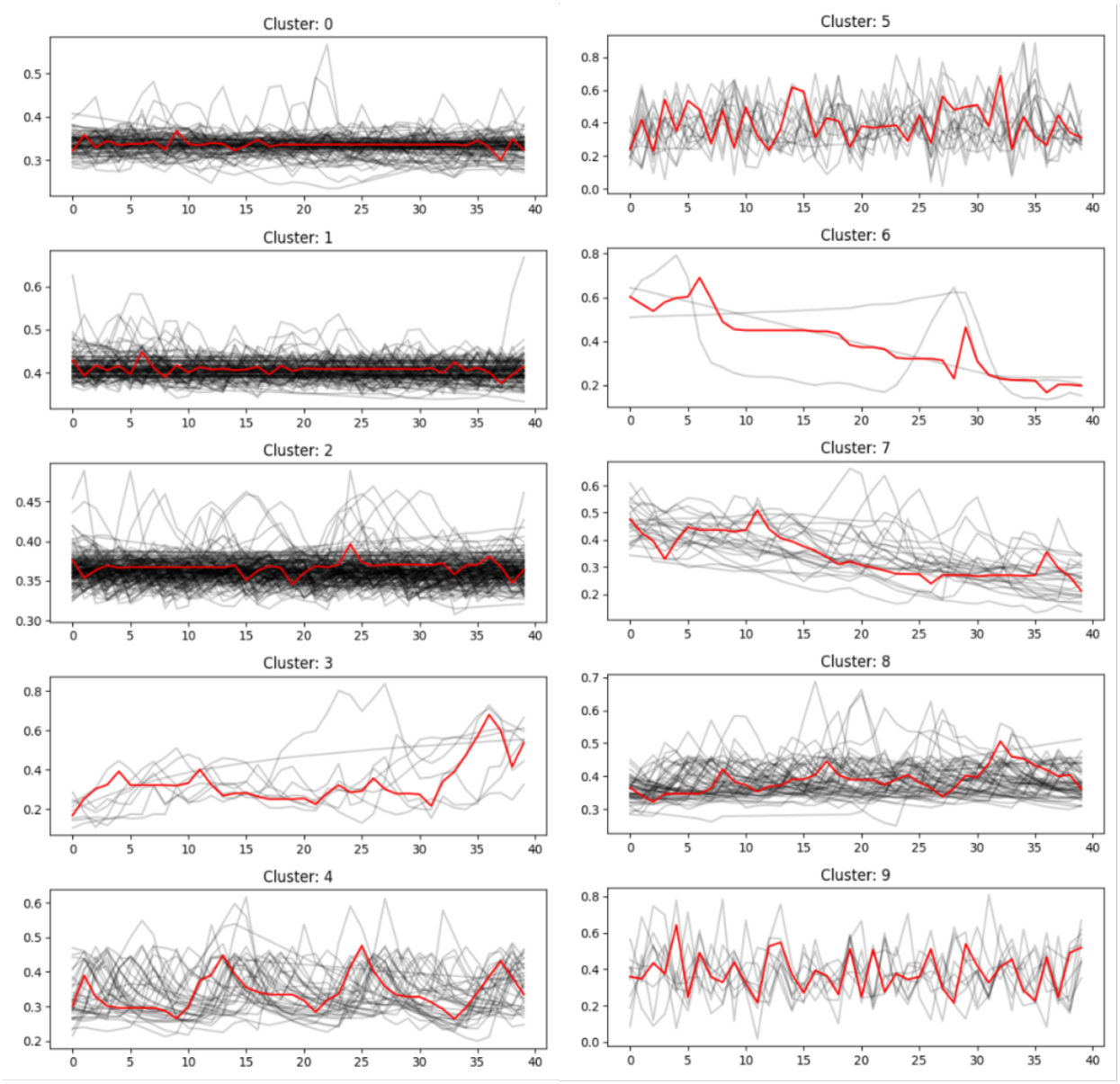


Figure B.12: Child clusters from parent cluster five from TSK-Means.

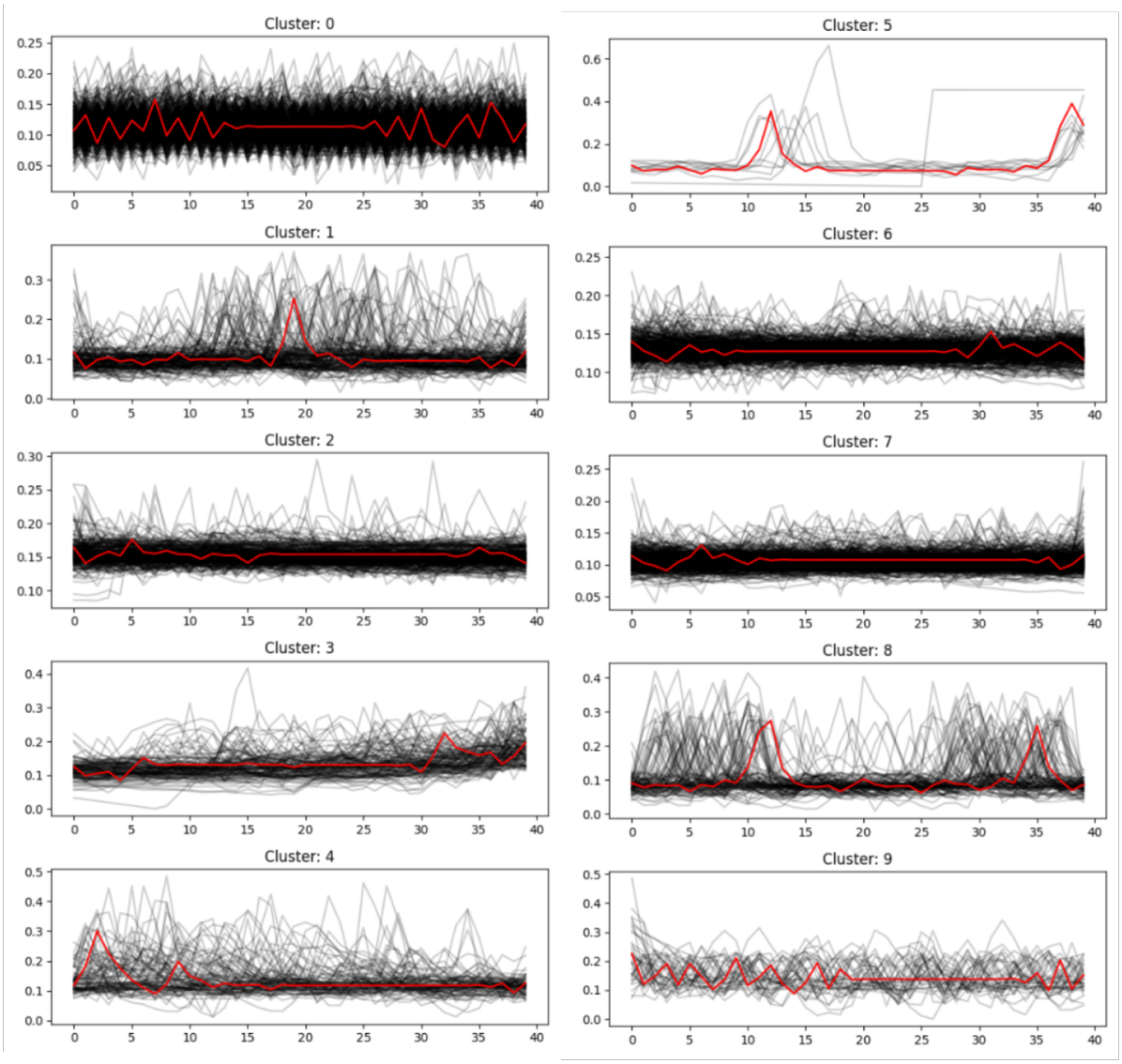


Figure B.13: Child clusters from parent cluster six from TSK-Means.

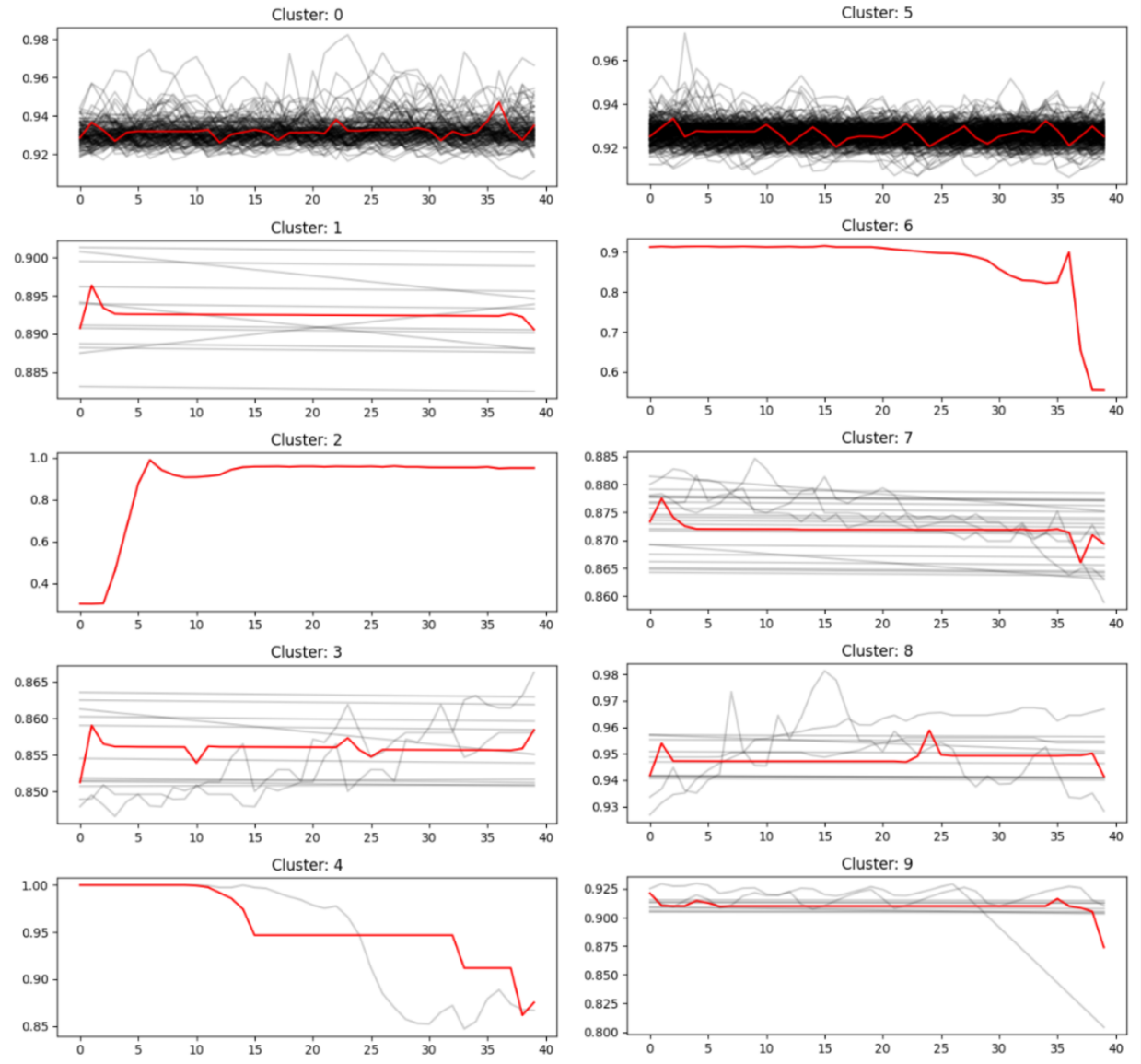


Figure B.14: Child clusters from parent cluster seven from TSK-Means.

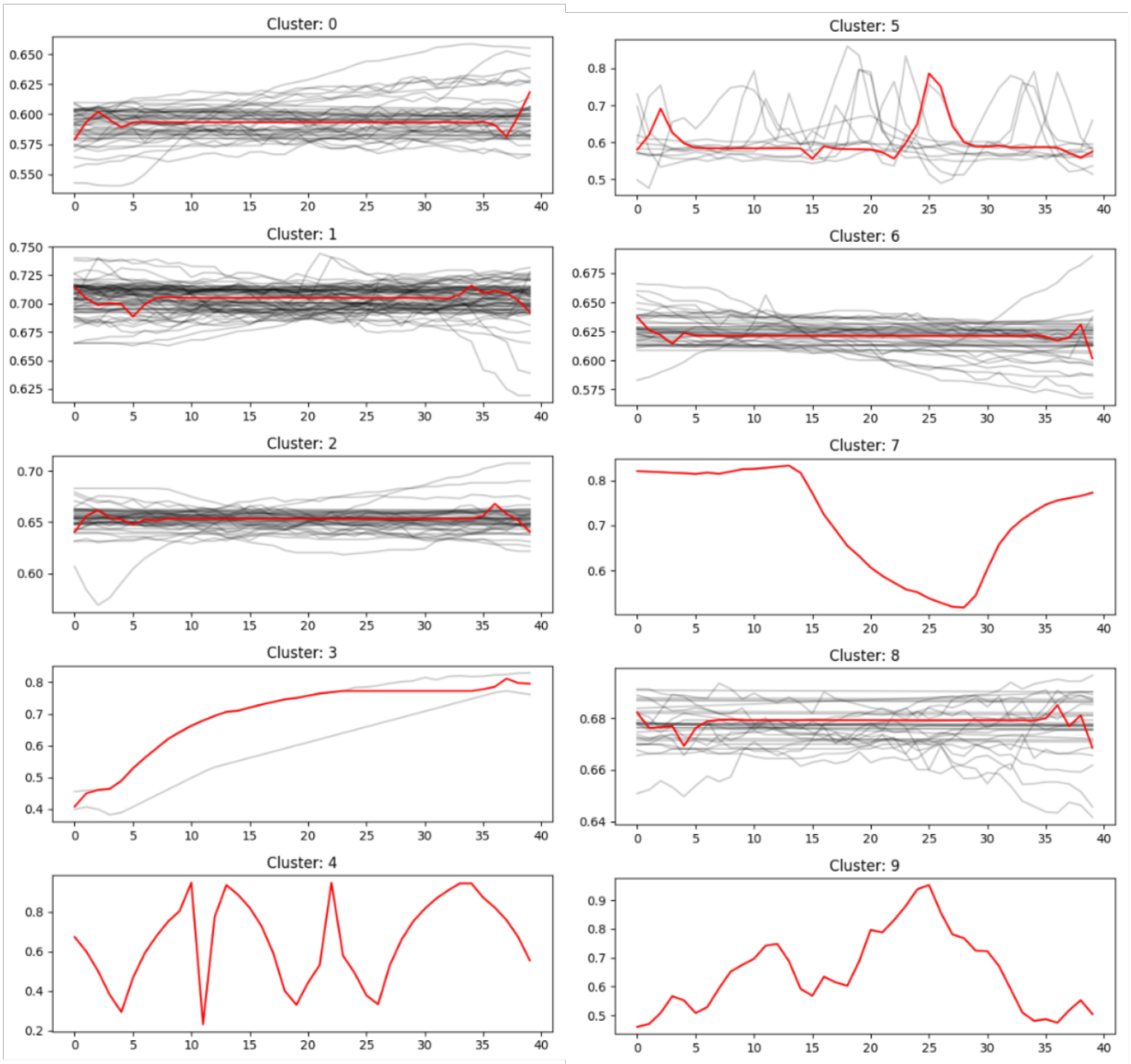


Figure B.15: Child clusters from parent cluster eight from TSK-Means.

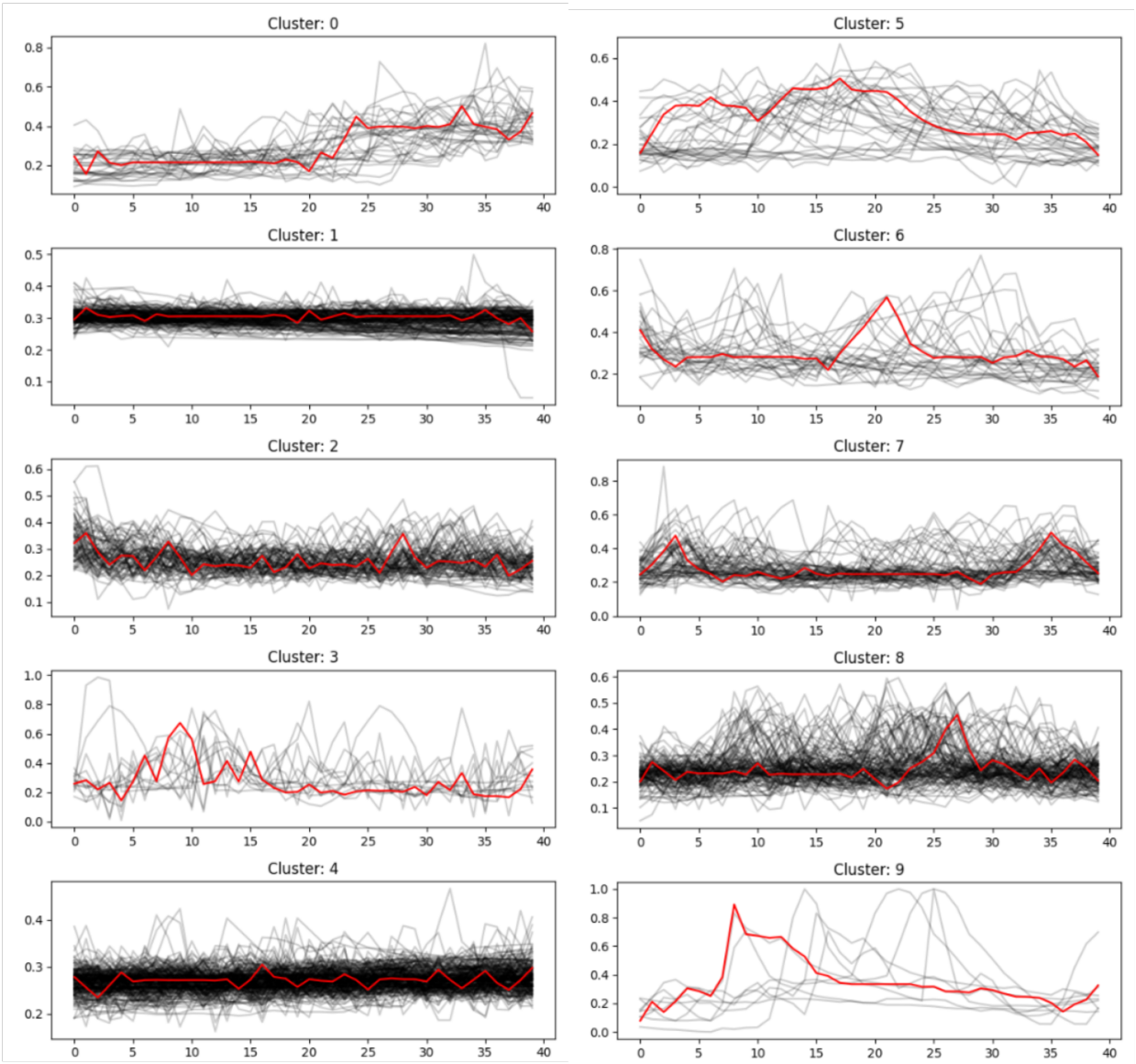


Figure B.16: Child clusters from parent cluster nine from TSK-Means.