technische universität dortmund

IT in Produktion und Logistik

Master Thesis

Data-driven analysis of anomalies in compressors: detection, monitoring, and prediction

A thesis submitted in fulfillment of the requirements for the degree

Master of Science in Manufacturing Technology (MMT)

The department of IT in Production and Logistics

Faculty of Mechanical Engineering

Supervisors:

Univ.-Prof. Dr.-Ing. Markus Rabe

Dr.-Ing. Dipl.-Inform. Anne Antonia Scheidler

Presented by

Majid Sedighi 181978

November 2017

Contents

List of Figures

1. Introduction

The importance of big data analytics is in the age of technology is a widely known and accepted fact in a wide array of businesses. The opportunities in cost reduction and fast decision making enabled by using techniques in big data analytics is desirable to every industry. However, the complexity of analysis in big data, the requirement of expert data scientists, and many other challenges has resulted in process and manufacturing industries not to benefit from the significant potential of big data analytics.

First challenge that industries are facing in the field of big data, is the lack of feasibility in recorded data that is stored in databases. Although managing to keep the data from all the necessary sectors of a process, these databases are not designed for big data analytics. As a result, contextualizing, searching for data, visualization, and other analysis tasks are either too complicated or time consuming for the data storing devices. This is closely related to the other major challenge in process and manufacturing industries: the online monitoring computation requirements. Data is produced a high rate in today's industry, and analyzing this data and extracting information from this data leading to knowledge discovery is achieved at a lower rate. This leads to decision making and relevant actions to be always few steps behind the effects of any occurrence on process performance.

The exponential growth of databases, and more importantly the recognition of the potential it bears has brought data mining to attention. Data mining is defined as the process of discovering patterns in data automatically or semi-automatically. Data mining gains ever increasing popularity through the rapid increase in the amount of all data available, as well as the accessibility of machines able to handle the search tasks (Witten et al. 2011). Current attempts in this technology however, produce results that are not as robust as the industry demands. This inconvenience is rooted in overgeneralization of the methods, the expertise required to keep the solutions running, and lack of flexibility in solutions.

It seems that the next generation in the development of data mining solutions, are plug and play solutions that do not require the information on the physical aspects of the process and are designed with regards to the needs of the manufacturing process specialists.

The main goal of this study is to develop a model-free, data-driven solutions for cross-asset detecting of asset degradation, anomalies, and suboptimal behavior in process industry.

A monitoring strategy using this method is suggested to detect failures from its early indications in the data. The fault detection method is developed with focus on one particular type of assets; serial reciprocating compressors within a chemical production process.

The aim for the research is for solutions to be applicable on a wide range of assets, and not to require modifications for different types of assets. In addition, the solution is aimed to be robust and easy to use, eliminating the need for a data scientist to implement and applicable by an engineer who has knowledge on the process, with little to no data science expertise.

The study starts with an extensive research on data analytics and fault diagnosis in chemical industry, following a review on the most recent developments in the methods for monitoring chemical and process industry assets, with focus on reciprocating compressors. The mathematical necessary for development of the fault detection method is also introduced and described.

The fault detection method is described in detail using a sample set of data, in a step-by-step manner. This method is the results of combining different mathematical methods and concepts in different areas of analytics, such as multivariate non-linear regression methods and artificial neuron networks. The attempt to choose a fitting methodology is made by application and modification of these techniques on use cases where actual process history log is available. In every step, the technique that gives the most advantages for the analysis is chosen and applied. Additionally, a strategy is suggested to complement the fault detection method for online monitoring.

The fault detection method developed in the study is implemented in R programming language as a prototype. This method is applied to real process data from Huntsman as a proof of concept, and the results of this data analysis is studied in detail.

2. Data analytics in chemical industry

Big data and next generation analytics are playing an important role in today's chemical industry. They are present in practically every aspect of the industry including manufacturing enhancement, supply chain management, marketing improvement, price fine-tuning, innovation support, and human resource management (Saha 2017).

There is a tremendous amount of data being produced every day, from which very little is analyzed and turned into useful knowledge. This lead to a universal collective effort to develop methods and tools to profit from this knowledge in a wide variety of applications and industries.

This chapter is dedicated to an in-depth look at big data analytics and statistical methods used in chemical and process industry. An extensive literature review on the state-of-the-art methods in fault diagnosis as well as the mathematical background required for developing the novel fault detection method is presented.

2.1. Big data and data mining

Big Data is defined as *"a situation in which data sets have grown to such enormous sizes that conventional information technologies can no longer effectively handle either the size of the data set or the scale and growth of the data set" (Ohlhorst 2013, p. 1).* Although not a new concept, big data has gained attention only recently since the means to interpret and extract useful information, namely modern computers, have been developed.

The world is in the Era of *Datafication*; a term coined by Mayer-Schönberger, Cukier (2013) Referring to modern technological trend turning many aspects of our life into computerized data. This movement has led the world into a data deluge, meaning that the collective data processing capacity is surpassed by sheer volume of new data being generated. This deluge is caused by a wide array of sources ranging from everyday purposes such as social media and phone sensors, to scientific applications such as geophysical exploration and gene sequencing (Dietrich et al. 2015).

According to Dobre, Xhafa (2014) there is around 2.5 billion gigabytes of data being produced around the world every day, and from this amount, 90% is unstructured. This fact shows a promising opportunity to make use of big data and its enormous potential, and has resulted in

scientists to start incorporating any related dataset, whether unstructured data, archival data, or real-time data into the process (Ohlhorst 2013).

Proper data processing can lead to new knowledge discovery as well as ways to deal with rising challenges and opportunities (Chen et al. 2013). Improved decision-making and business strategies can be achieved through implementation of advanced big data analysis technologies (Yi et al. 2014). Sivarajah et al. (2017) summarize the opportunities of big data as value creation, rich business intelligence for better informed decisions, and enhancement of flexibility in supply chain and resource allocations. However, within these great opportunities lies great challenges. These challenges include data integration complexities, lack of skilled personal and resources, data security and privacy, inadequate infrastructure, and synchronization of large data. With regards to these challenges, Zhang et al. (2015) argue that in order to benefit from the full potential of big data, it is necessary to develop technologies that are extremely efficient, scalable and flexible, with no regards to data type or format. Therefore, common statistical analysis that is widely used today is not nearly sufficient to reveal this potential.

2.2. Fault diagnosis in chemical processes

Abnormal event management (AEM) is the science of detecting diagnosing and correcting the abnormal conditions of faults in a process in a timely manner. Venkatasubramanian et al. (2003b) stated that AEM related problems are the number one cause of petrochemical industry losing 20 billion dollars every year. Fault detection plays an increasingly important role in process industry, due to its key role in ensuring safety and productivity of processes (Jiang et al. 2013). Failures and malfunctions in chemical plants, leads to an increase in operation costs and downtimes. Himmelblau (1978) described chemical plants by their complex processes and equipment, high throughput, and serious consequences for serious accidents. These characteristics leads to a lot of effort and money spent on process fault detection methods.

The term *fault* is generally defined as a departure from an acceptable range of an observed variable or a calculated parameter associated with a process (Andow 1980) . Fault detection lies at the core of AEM, making it a subject of interest for many studies and publications in more recent years. A wide array of fault detection methods have been developed over the past couple of decades, ranging from analytical methods to artificial intelligence and statistical approaches. Venkatasubramanian et al. (2003b) classify these methods based on the type of

prior knowledge and the type of diagnostic search strategy. With this approach, the methods are categorized into three general groups: quantitative model-based methods, qualitative model-based methods, and process history based methods. These categories and their sub-categories can be seen in Figure 2. 1.

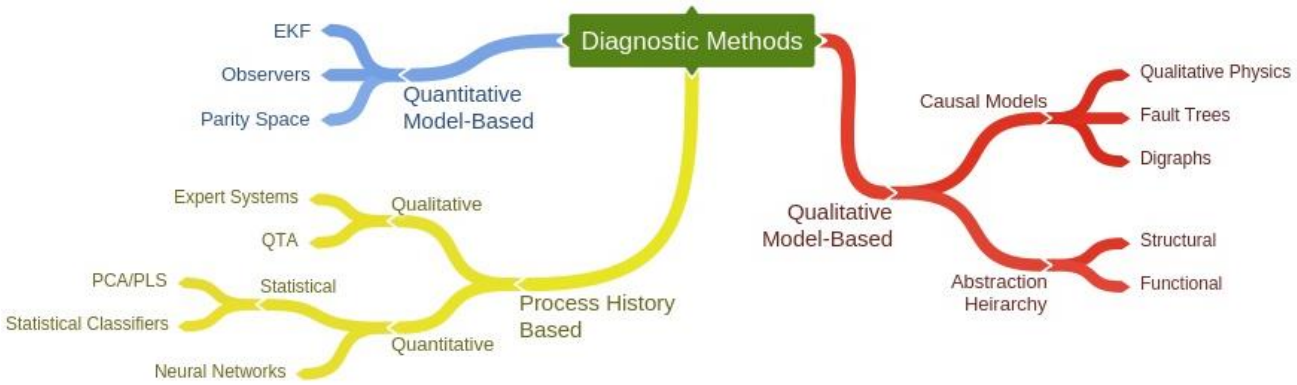

Figure 2. 1 - Classification of Diagnostic methods (Venkatasubramanian et al. 2003b)

A more general approach in classifying process fault detection techniques is introduced by Himmelblau (1978) in two categories: parameter estimation and pattern recognition methods.

2.2.1. Pattern recognition methods for fault detection and diagnosis

Pattern recognition is defined as *"the process of assigning a label or category to a pattern on the basis of certain features in the pattern" (Himmelblau 1978, p. 273)*. Pattern recognition methods are generally used in cases where a complete knowledge on probability distribution of the data is not available. If the number of measurements are not enough to draw meaningful statistical distributions, nonparametric techniques are used instead. The two main techniques of pattern recognition are template matching and feature extraction, of which the latter is usually used in chemical process industry. (Himmelblau 1978).

Pattern recognition methods are categorized by Himmelblau (1978) in three major categories: Fault dictionaries, Cluster Analysis, and Acoustic Noise and Vibration monitoring. Himmelblau argues that statistical analysis applied to pattern recognition is superior deterministic pattern recognition due to its ability to compensate for the small non-faulty drifts and observation noise.

Multivariate statistical techniques have proved to be powerful tools in pattern recognition methods. Their capability in reducing the dimensionality of the data while retaining essential

information, as well as their ability to use correlations to effectively extract true information while handling noise and small inconsistencies in data has made these techniques an essential tool in every diagnostic system that follows a pattern recognizing approach in the recent years (Venkatasubramanian et al. 2003a).

2.2.2. Fault diagnostics and monitoring assets in chemical industry

In this section a comprehensive literature research is presented on the state of the art in fault detection and diagnostics in chemical process assets, namely compressors. The methods that are applicable to chemical and process industry are studied first, following a more focused research on reciprocating compressors specifically.

Multivariate Statistical Process Monitoring (MSPM) methods have recently became the most widely used methods in diagnosis and monitoring of processes and assets in Chemical industry. Among these methods, principal component analysis (PCA) is most frequently used due to its simplicity and ability to work with high-dimensional noisy data (Jiang et al. 2013).

Rotem et al. (2000) combined PCA with model based multivariate statistics to compensate for shortcomings of conventional PCA in the case of nonlinearly correlated parameters and periodic fluctuations in data. This method, also known as Model-Based PCA (MBPCA), the portion of variation that is not predicted in the retained Principle components is analyzed using a model of the process, improving the diagnostic resolution significantly. The study and its validation on an Ethylene compressor confirmed that although MBPCA requires addressing the requirements of both a model-based method and a data-driven one, a good performance is achievable even with a relatively simple model of the process.

Chen and Yan (2012) proposed a novel method for fault diagnosis in chemical industry, incorporating *self-organizing maps (SOM)* and *correlative component analysis (CCA)*. CCA is a pattern classification technique, proposed by Chen et al. (1996), that aims to reduce the dimensions of complex classified data by evaluating the relative importance of each component in the data in a quantitative manner. This method is designed to take classification information supplied with data into consideration while performing dimension reduction tasks, meaning that the result of this transformation in CCA space will attempt to separate the data based on its classification with regards to the correlations between the variables. The results of analysis on sample data shows that this method gives more satisfactory results than PCA and the sequential discriminant analysis in practical classifying problems.

The fault diagnosis method developed by Chen and Yan (2012) takes advantage of CCA abilities in extracting fault classification information, combined with discriminating different states of the process on the output map resulted from SOM. The efficiency of this method in real time monitoring of complex chemical processes is verified in a case study. This method shows improved performance compared to simple SOM and PCA combined with SOM with regards to accuracy of fault diagnosis.

An improvement to the conventional PCA monitoring methods is suggested by Jiang et al. (2013). This modification, called Sensitive Principal Component Analysis (SPCA), aims to provide a guideline to the principal components that are selected for developing control charts for monitoring a process. This guideline is composed through finding principal components that are responsible for dominant variations in an anomaly observation, and mark them as Sensitive Principal Components (SPC).

In an attempt to develop a purely adoptive model maintenance strategy, Lu et al. (2017) proposed a system based on GSOM to model processes with drifts and outliers. This algorithm utilizes local partial least squares (PLS) models in GSOM projections in order to achieve the flexibility required for adjustments while training complex data as well as in online adaptation. This method is specifically designed to be able to explore new data and develop prototype models for data. This model is verified to out show better performance than conventional static, non-linear, and moving window PLS methods. The combination of GSOM flexibility to adapt to overall multi model system combined with the precision of local PLS models makes this method robust in dealing with abrupt process shifts and transitions.

The state of the art in condition monitoring of reciprocating compressors is studied extensively by Schultheis et al. (2007). Their study discussed different measurements and protective functions, and compared different methods of monitoring for reciprocating compressors. As a result, monitoring techniques are categorized in four groups based on the measurement or property that is monitored.

1. **Vibrations**: Measurements on the crankcase as well as crosshead or distance piece of the compressor are the most effective values for monitoring.
2. **Temperature**: The main temperature measurements in a reciprocating compressor are valve temperatures, discharge temperature, packing temperature, and bearings temperatures.
3. **Rod drop and Rod runout**: Rod drop is the indication of the vertical piston movement inside the cylinder, calculated at the pressure packing case. Rod runout refers to deviations of rod from its main axis position.

4. **PV Analysis**: Pressure velocity (PV) is monitored through measurement of dynamic pressure inside a compressor throughout a stroke using a dynamic pressure transducer.

Ahmed et al. (2012) developed a method for fault detection and diagnosis of a multistage reciprocating compressor, using a PCA based method to various faults including suction valve leakage, inter-cooler leakage, loose belt drive and other possible failures in a reciprocating compressor. The method used Hotelling's T^2 and Q-statistics to compare the feature values from the time domain of the vibration signal. Ahmed et al. demonstrated that PCA based methods have the ability to detect single and multiple faults in a reciprocating compressor. The study also showed Q-statistic's superiority to Hotelling's test in detecting faults.

Farzaneh-Gord and Khoshnazar (2016) developed a numerical method to analyze a single stage reciprocating compressor for valve failures. This model takes piston movement, valve dynamics, and mass flow rate in the cylinder into account. The simulation results of this study agreed with experimental results, showing that malfunctions in suction and discharge valves will lead to decrease in compressor mass flow and increase in discharge temperature, with suction valve having more significant effects. A suction valve failure will additionally cause an increase in suction temperature. The study suggested that monitoring suction and discharge temperatures of the compressor can be effective in valve failure diagnosis.

Kostyukov and Naumenko (2016) used Russian methodological and standards base to develop a novel online health monitoring technique for reciprocating compressors under the control of vibration-Based diagnostics. This method enables the operator to use real time condition of the machine to plan use of spare parts and repairs, which results in an extension in the life span of the compressor.

Lu et al. (2016) proposed a method for centrifugal compressor fault diagnosis using thermal parameters of the compressor. This method is based on an adaptive trend extraction algorithm and a sliding window based matching strategy, and demonstrates enough robustness and efficiency to detect the failure in its early stages, preventing serious damage to the compressor.

Potočnik, Govekar (2017) developed a semi-supervised vibration-based classification and condition monitoring method for reciprocating compressors using a combination of feature extraction, PCA, and statistical analysis. This method addresses the problem in refrigeration appliances that despite the large database of vibration-based measurements available, there is no information extracted to define classes of operations for these compressors. The study

compares different classification methods and verifies that although linear methods such as DA and SVM are suitable candidates for industrial applications due to their simplicity, more complex problems and more general solutions are achieved through using nonlinear classifiers such as neural networks trained by Bayesian regularization.

2.2.3. Criteria for evaluation of diagnostic systems

Venkatasubramanian et al. (2003b) identified the desirable characteristics of a diagnostic system. A set of criteria is derived based on these characteristics that can be used to benchmark different diagnostic systems. However, the following concept must be considered when comparing diagnostic systems. The main task for a diagnostic classifier is to suggest hypotheses or faults the explain occurring abnormalities. These explanations should be complete, meaning that the actual faults should be a subset of the proposed fault set. On the other hand, the desirable resolution for the system aims to make the fault set to be minimal. Therefore, one should consider the trade-off in the accuracy of predictions between completeness and accuracy. Aside from this concept, there are ten characteristics that Venkatasubramanian et al. found desirable for diagnostic systems to possess.

1. **Quick detection and diagnosis**: Although it is desirable for the system to respond quickly in detecting malfunctions, this can be disruptive to the performance of the system during normal operation (Willsky 1976). A system too sensitive to can detect high frequency noise as failure, which leads to frequent false alarms during normal operation.
2. **Isolability**: It is expected of the diagnostic system to distinguish between different failures. This is in contrast with rejection of modelling uncertainties, meaning that a system with high degree of isolability usually performs poorly when it comes to rejecting modelling uncertainties.
3. **Robustness**: The system is desired to be robust to noise and uncertainties, enabling the performance to degrade gradually when a failure happens.
4. **Novelty identifiability**: In addition of the diagnostic system determining the process condition as normal or abnormal, it is also expected of the system to decide if the abnormality is caused by an unknown ‘novel’ malfunction. This feature is important because usually most of the historic data available for a process fall under normal operation zones. There is limited data available for different kinds of failure, and often some failure types are completely unknown as far as the historic data goes.
5. **Classification error estimate**: If a system is able to give an estimate on the classification error that can occur in the analysis, the recommendations by the system will be more reliable to the user.

6. **Adaptability**: It is very much desirable for a diagnostic system to be adaptable to changes in the process, whether it is due to disturbances, or environmental conditions such as production quantities and quality of raw material.
7. **Explanation facility**: The ability to explain the origin and propagation of the faults is an important factor for diagnostic systems. The justification of the recommendations by the system will lead to a more accurate on-line decision support system.
8. **Modelling requirements**: The modelling effort required for fast and easily deployed real-time diagnostic system should be minimal.
9. **Storage and computational requirements:** A balance is to be found between the complexity of the algorithms and the storage requirements of a diagnostic system.
10. **Multiple fault identifiability**: Due to interacting nature of most of faults, it is usually difficult to identify multiple simultaneous faults in a process. The patterns that occur when a combination of faults is happening is not necessarily detectable by a system trained to detect the faults individually. Additionally, designing separate models for different combinations of faults is not feasible for large processes.

This ten-part criteria will be the basis of evaluation for the fault detection method that is developed in this study.

2.3. Multivariate data analysis

Multivariate data analysis refers to *"all statistical techniques that simultaneously analyze multiple measurements on individuals or objects under investigation." (Hair 2006, p. 4)*. These methods are usually extensions of univariate and bivariate analysis. Some of the methods are designed to replace multiple steps of univariate analysis, while others are uniquely designed to solve multivariate issues (Hair 2006).

Due to the recent explosion in the information produced and available, as well as development of new technologies to store this data, there is a need for means to study this data and extract knowledge for decision making. Although some cases are possible to analyze using simple statistics, most of the data requires more complex methods and tools for knowledge discovery. Hence there is growing interest and investment in Multivariate statistical techniques as one these tools (Hair 2006).

In the following, three multivariate analysis techniques are described in detail. These methods are the building blocks of the novel fault detection method that is developed in this study.

2.3.1. Principal Component Analysis

Principal Component Analysis (PCA) is one of the most important and powerful methods for dimension reduction. It is a data-driven modelling technique that aims to transform the data into fewer dimension with minimum loss of data.

The method has been modified and reformulated over the years. There are two different approaches to PCA; on one hand there is Pearson's (1901) regression based school of thought, with focus on modelling properties of PCA. As the first inventor of the method, Pearson interpreted the method as explaining the variation with principal components. On the other hand, there is the multivariate statistical approach to PCA, as conceptualized by Hotelling (1951) around the same time as the idea of taking linear combination of variables was introduced. The variation of the principal components is stressed in this approach.

These two methods, although seemingly similar, have fundamental conceptual differences. Hotelling's approach, focuses on the specific direction of principal components. PCA in this approach functions as a method to define new axes that explain the most of variation. In Pearson's approach however, it is the subspace that plays the most important role, and the axes act as a basis for the subspace (Bro, Smilde 2014).

2.3.1.1. Formulation

PCA as usually formulated today follows an approach closer to that of Hotelling. Based on Jollife (2002), the following formulation is used to derive the principal components: Suppose $\mathbf{X}$ as a vector of p random variables. Since looking at the whole structure of variances and covariances of the variables is time consuming and not very useful, the alternative approach of looking for a few derived variables is suggested. These variables are to be found in a way that they preserve the most possible information given by the variances and correlations for covariances.

The first step is to look for a linear combination of the variables that have the maximum variance, this combination is denoted as $\boldsymbol{\alpha}'_1\boldsymbol{X}$, where $\boldsymbol{\alpha}$ is a vector of p constants, so that

$$\boldsymbol{\alpha}'_1\boldsymbol{X} = \alpha_{11}x_1 + \alpha_{12}x_2 + \cdots + \alpha_{1p}x_p = \sum_{j=1}^{p} \alpha_{1j}x_j \qquad (2.1)$$

Next, we look for another linear combination $\boldsymbol{\alpha'}_2\boldsymbol{X}$, which is uncorrelated with $\boldsymbol{\alpha'}_1\boldsymbol{X}$ and has the maximum variance. Repeating the same algorithm, we can find up to p vectors that are called the *Principal Components*. The goal of the algorithm however, is to account for all of the variance in data by using a number of principal components much less than p.

Generally, if a set of data with high number of variables has significant correlations among the variables, then the first few principal components will account for most of the variation in the original variables. In contrast, the last few principal components will belong to directions with little to no variation, identifying the closest to linear relationships in the original variables. An example of a dataset transformed into PCA space is shown in Figure 2. 2.

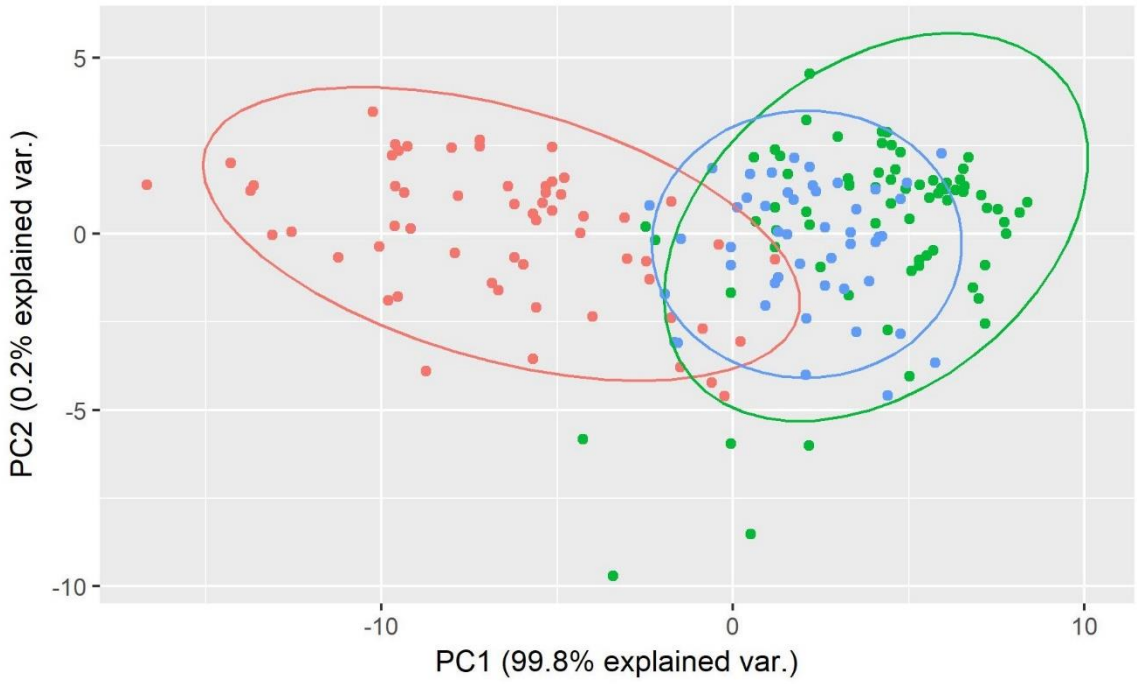

Figure 2. 2 - dataset with three classes of data transformed to the first two principal components

The approach to finding the principal components are described by Jollife (2002) as following. Considering $\boldsymbol{\Sigma}$ as the covariance matrix of our vector of random variables $\mathbf{X}$, this matrix consists of the variance of each element of $\mathbf{X}$ on the diagonal, and the covariances of every combination of variables in $\mathbf{X}$ on the other elements of the matrix. It turns out that each $\boldsymbol{\alpha}_k$ in $\boldsymbol{\alpha'}_k\boldsymbol{X}$ is the *kth* eigenvector of $\boldsymbol{\Sigma}$, corresponding to λ_k, the *kth* largest eigenvalue of $\boldsymbol{\Sigma}$. Additionally, the variance value for each PC is in fact their corresponding eigenvalue, but only in case of choosing $\boldsymbol{\alpha}$ to have unit length.

The following course of action is suggested by Jollife (2002) for deriving the principal components. First, $\boldsymbol{\alpha}_1$ is considered in such way that the variance of $\boldsymbol{\alpha'}_k\boldsymbol{X}$ is maximized. Knowing that $var(\boldsymbol{\alpha'}_1\boldsymbol{X}) = \boldsymbol{\alpha'}_1\boldsymbol{\Sigma}\boldsymbol{\alpha}_1$, it is clear that there is no finite $\boldsymbol{\alpha}$ that maximizes the expression and therefore another condition for $\boldsymbol{\alpha}$ has to be imposed. This condition is in fact the unit length as mentioned before, which is applied as $\boldsymbol{\alpha'}_1\boldsymbol{\alpha}_1 = \mathbf{1}$. In order to maximize the variance value under this condition the technique of Lagrange multipliers is used, where the maximized equations transform into

$$\boldsymbol{\alpha'}_1\boldsymbol{\Sigma}\boldsymbol{\alpha}_1 - \lambda(\boldsymbol{\alpha'}_1\boldsymbol{\alpha}_1 - 1) = 0 \tag{2.2}$$

Where λ is a Lagrange multiplier. This results in

$$(\boldsymbol{\Sigma} - \lambda\boldsymbol{I}_P)\boldsymbol{\alpha}_1 = 0 \tag{2.3}$$

With $\boldsymbol{I}_P$ as an identity matrix with dimensions *p*. From this equation, it is obvious that $\boldsymbol{\alpha}_1$ is an eigenvector of $\boldsymbol{\Sigma}$, and since for the first PC the largest variance $\boldsymbol{\alpha'}_1\boldsymbol{\Sigma}\boldsymbol{\alpha}_1 = \lambda_1$ is desired, this eigenvector corresponds to the first and largest eigenvalue of $\boldsymbol{\Sigma}$. It can be proven that the other principal components are also eigenvectors of $\boldsymbol{\Sigma}$, corresponding to eigenvalues in an orderly fashion.

A major application for PCA is in developing control charts for processes. The two more usual methods for developing such control charts based on PCA analysis are *Hotelling's* T^2 statistic and the *Q-statistic*, also known as the *Squared Prediction Error (SPE)* (Ketelaere, Schmitt).

2.3.1.2. Hotelling's statistic

Hotelling's T^2 statistic is describe as "an important tools for inference about the center of a multivariate normal population" (Willems et al. 2002, p. 125). It was first introduced by Hotelling (1931) as an extension on the t-distribution by Student (1908) to multivariate statistics.

The test is used in its practical form as following according to Penha (2001). First, it is necessary to decide how many of the principal components are to be retained for the test. A suggestion is to keep all the principal components with the value of their variance greater than 1. This number is denoted k. $\boldsymbol{P}$ is set to be the $p \times p$ loading matrix, resulted from the PCA, and $\boldsymbol{P}_k$ is the first k columns of that matrix. $\bar{\boldsymbol{x}}$ denotes the mean of all the data points. The T^2value for any point $\boldsymbol{x}$ in a p dimensional space is calculated

$$T^2 = (\boldsymbol{x} - \bar{\boldsymbol{x}})'\boldsymbol{P}_k\boldsymbol{\Lambda}_k^{-1}\boldsymbol{P}'_k(\boldsymbol{x} - \bar{\boldsymbol{x}}) \tag{2.4}$$

Where Λ_k is a diagonal matrix with dimensions k where the elements are the k largest eigenvalues of the data covariance matrix λ_1 to λ_k, which are also the variance values for the principal components.

$$\Lambda_k = \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_k \end{bmatrix} \tag{2.5}$$

T^2 is in fact the *Mahalanobis distance* of $\boldsymbol{x}$ in PCA space, a measurement proposed by Mahalanobis (1936).

If the total number of points in the dataset in denoted by n, then the control limit for Hotelling's T^2 is calculated

$$T_\alpha^2 = \frac{k(n^2-1)}{n(n-k)} F_{k,n-1}(\alpha) \tag{2.6}$$

Where F is $F_{k,n-1}(\alpha)$ is the $(1-\alpha)$ percentile of the F-distribution with k and $n-k$ degrees of freedom. This threshold assumes anomaly to be the α percentile outliers of the data in terms of distance to the center of the data in PCA space.

F-distribution is a continues probability distribution introduced by Snedecor (1989). The *probability density function* of F-distribution for a random variable x and degrees of freedom d_1 and d_2 is

$$f(x; d_1, d_2) = \frac{\sqrt{\frac{(d_1 x)^{d_1} {d_2}^{d_2}}{(d_1 x + d_2)^{d_1+d_2}}}}{x B(d_1/2, d_2/2)} \tag{2.7}$$

Where B is the *beta function*

$$B\left(\frac{d_1}{2}, \frac{d_2}{2}\right) = \int_0^1 t^{\frac{d_1}{2}-1} (1-t)^{\frac{d_2}{2}-1} dt \tag{2.8}$$

2.3.1.3. Q-statistic

Q-statistic is another control process defined in PCA space. It is defined as a Control procedure for residuals associated with PCA in the original paper (Jackson, Mudholkar 1979). Also described as the sum of squares of the residuals, the Q-statistic for the vector $\boldsymbol{x}$ is

$$Q = (\boldsymbol{x} - \overline{\boldsymbol{x}})'(\boldsymbol{I} - \boldsymbol{P}_k \boldsymbol{P}_k')(\boldsymbol{x} - \overline{\boldsymbol{x}}) \tag{2.9}$$

Q-statistic is in fact the quadratic orthogonal distance to the PCA space. The control limit for Q is calculated

$$Q_\alpha = \theta_1 [\frac{C_\alpha \sqrt{2\theta_2 {h_0}^2}}{\theta_1} + 1 + \frac{\theta_2 h_0 (h_0 - 1)}{{\theta_1}^2}]^{\frac{1}{h_0}} \tag{2.10}$$

Where

$$\theta_i = \sum_{j=k+1}^{p} {\lambda_j}^i \, , i = 1, 2, 3 \tag{2.11}$$

$$h_0 = 1 - \frac{2\theta_1 \theta_3}{3{\theta_2}^2} \tag{2.12}$$

And C_α is the $(1 - \alpha)$ percentile of the standard normal distribution.

Geometrically for a point in PCA space, Hotelling's T^2 is the distance between the point and the center of all the points in the retained principal components hyperplane, and Q-statistic is the normal distance between the point and the said hyperplane. A simplified visual representation of this is shown in Figure 2. 3 for a 3-variable dataset with 2 retained principal components.

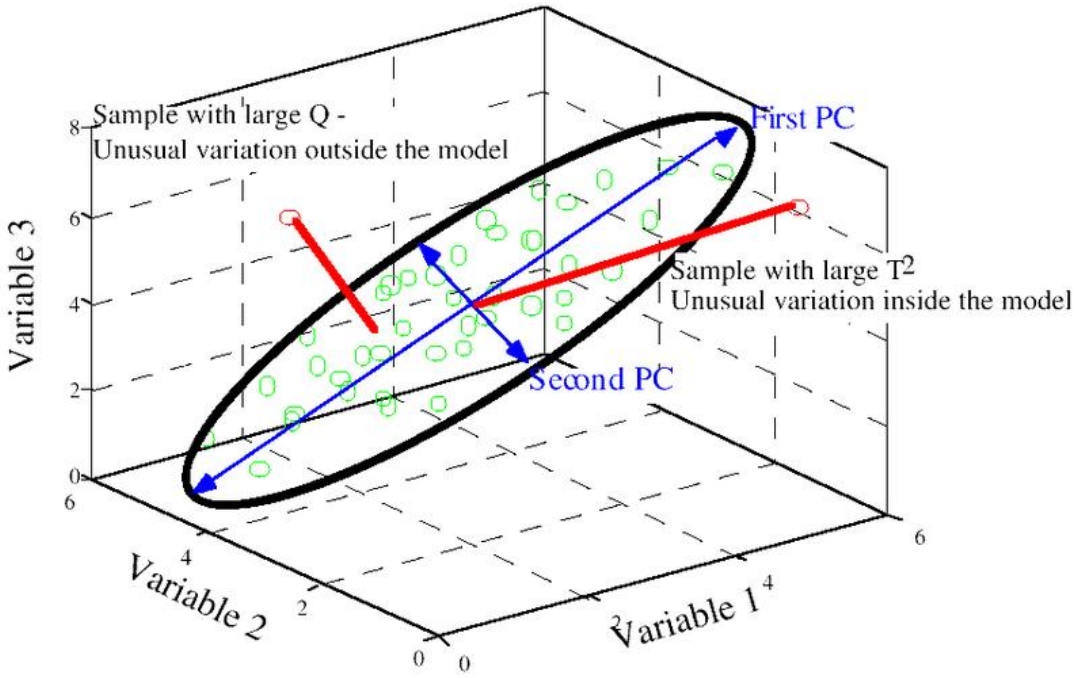

Figure 2. 3 - Hotelling and Q-statistics outliers representation in PCA space (Penha, Hines 2001)

2.3.2. Kernel Density Estimation

Kernel density estimation (KDE) is a non-parametric procedure used for visualization and regression purposes in data science. This method provides a simple and reliable estimate on the probability density function of random variables without the specification of a parametric model. In simple terms, KDE provides a simple way to find structure in a dataset without the need to assume a parametric model for the data. The main principles of this method, also referred to as Kernel smoothing techniques, can be extended to statistical problems in a wide range of applications such as signal processing, econometrics, engineering and medicine (Wand, Jones 1995).

KDE, as it is used today was first introduced by Rosenblatt (1956) as a set of nonparametric estimates of a Density function. At around the same time, Parzen (1962) developed the same method independently, presenting it as a method for estimation of probability Density function and mode. The creation of the method is therefore accredited to both authors.

KDE is a counterpart to *parametric regression* models. These models make assumptions on the functional for of the regression function. It can be in a linear, parabolic, or any other form. The choice of a specific parametric model can be based on different scientific or experimental reason. However, the limitations of a parametric model hold true regardless of its derivation

method. The rigidity of restricting the model to belong to one parametric family is the main limitation of parametric models. The chosen of the parametric model, if not done carefully, can lead to incorrect conclusions in the regression analysis and therefore to misinterpretation of the data (Wand, Jones 1995).

2.3.2.1. Univariate kernel density estimator

The method is first described in one dimension, called *Univariate Kernel Density Estimator*. Univariate KDE is most straight forward type of kernel estimators, making a thorough study of its properties possible (Wand, Jones 1995). given a sample $X_1, \dots, X_n$ with a continuous, univariate density *f*, the Kernel density estimator is

$$\hat{f}(x,h) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right) \tag{2.13}$$

Where K is kernel shape function and h is the bandwidth. In practice, the working principle of the method is that a kernel is placed on each of the data points, and the influence of the point is spread about its neighborhood. These influences are then added up to form the overall estimate. Different kernels can be used for different purposes, such as uniform, triangular, Gaussian, and *Epanechnikov* kernels. Figure 2. 4 shows a schematic of univariate kernel density estimate with Gaussian kernels.

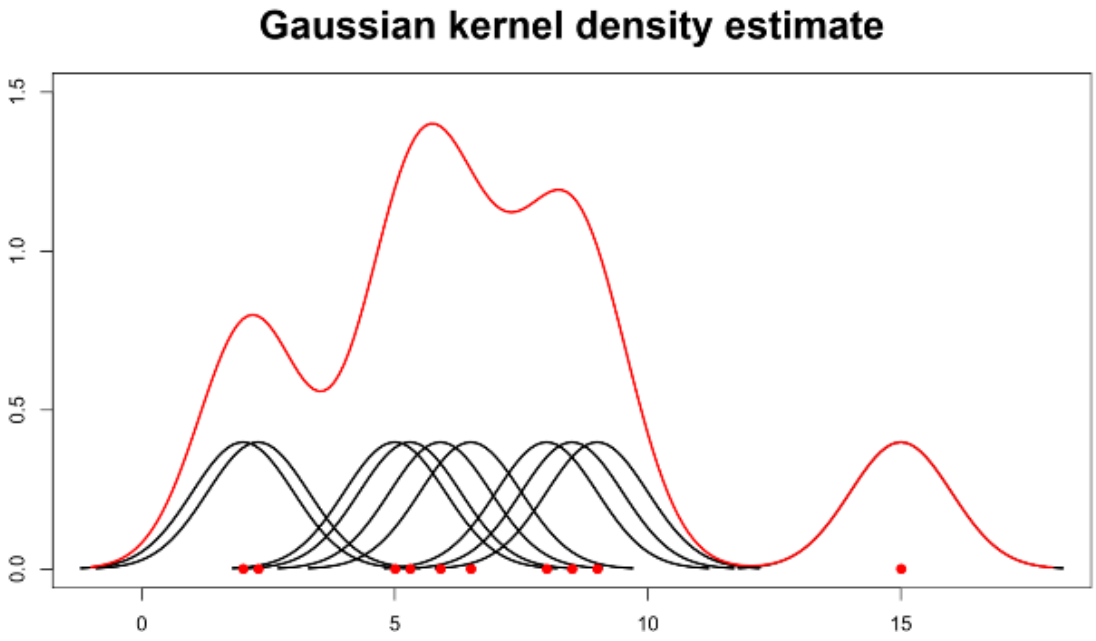

Figure 2. 4 - Univariate kernel density estimation using gaussian kernels (Wand, Jones 1995).

Wand and Jones (1995) showed that the shape of the kernel function is not a particularly important choice. However, the value chosen for bandwidth *h*, plays a very important role in the quality of the estimation. If the value of *h* is too high, then the curve will be overfitted, and in contrast a number too high for *h* results in an estimation that is not fitting to the true density of observations.

Another factor that influences the quality of the estimation is the shape of the kernel function K. The shape function K is usually defined as a symmetric and unimodal density. The reason for these choices, aside from the simplicity of interpretation, is a criterion for *admissibility* of kernel shape functions introduced by Cline (1988). Based on this criterion, a kernel density estimator is considered admissible, if it compared to other kernel estimators, produces the smallest value for mean integrated square error among all densities and sample sizes. This means the kernel must have nonnegative Fourier transforms that is bounded by 1.

Among all of the possible Kernels, the optimal Kernel shape is found by Hodges, Lehmann (1956) to be of the form

$$K^a(x) = \frac{3}{4}\{1 - x^2/(5a^2)\}/\left(5^{1/2}a\right)1_{\{|x|<5^{1/2}a\}} \qquad (2.14)$$

Where a is an arbitrary scale function. Epanechnikov (1969) described the optimality properties in density estimation setting for the simplest version of K^a with choosing $a = \sqrt{1/5}$. Often referred to as the *Epanechnikov kernel*, it has the formula

$$K^*(x) = \frac{3}{4}(1 - x^2)1_{\{|x|<1\}} \qquad (2.15)$$

The graph for this kernel is shown in Figure 2. 5 in comparison with uniform, Gaussian, and triangular kernels.

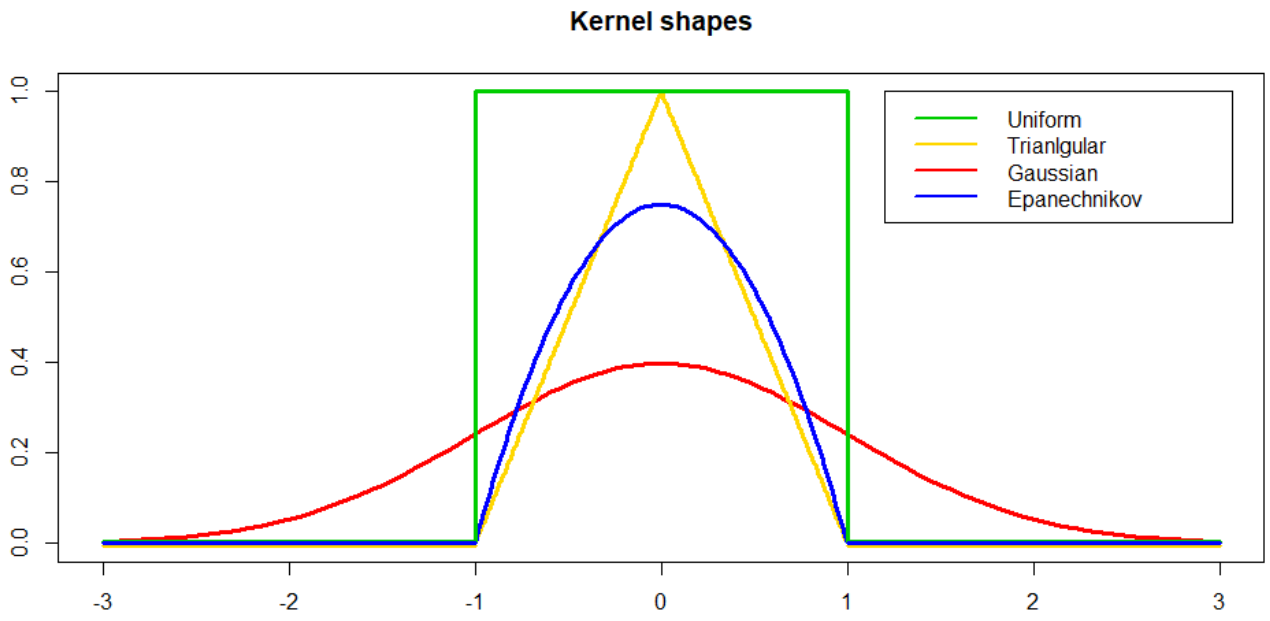

Figure 2. 5 - Comparison between some conventional kernel shapes

Although the Epanechnikov kernel has the optimal shape for kernel estimation settings, Wand and Jones (1995) showed the inferiority of other unimodal densities such as the Gaussian in estimation performance is not significant. Therefore, the choice of the Kernel is often based on other criteria such as computational efficiency. For example, uniform kernels are not very popular since the resultant estimate is piecewise constant. Based on such reasons, even the Epanechnikov kernel is considered unattractive, due to the discontinuous first derivative produced by the estimate.

2.3.2.2. Multivariate kernel density estimator

The extension of the univariate kernel density estimation to more dimensions ($x \in \mathbb{R}^d$) is called *The Multivariate Kernel Density Estimator*. There are challenges with extending this method to higher dimensions. First of all, the most general parameterization of the kernel estimator requires many more bandwidth parameters than the univariate case (Wand, Jones 1995). The other problem is that the variation in the data makes it necessary for the sample size to be very large for kernel estimation analysis to be relevant. Also referred to as the *curse of dimensionality* (Kárný, Warwick 1997), it can be shown that reasonable nonparametric density estimation proves to be very difficult in more than five dimensions.

In spite of all the challenges, Silverman (1998) has proved kernel density estimators to be powerful tools for analyzing bivariate structures. Scott (2015) even went on further to show

the important role of multivariate kernel density estimation in visualizing the structures of three- and four-dimensional datasets.

According to Deheuvels (1977) the extension in its most general form is

$$\hat{f}(x, \boldsymbol{H}) = \frac{1}{n}\sum_{i=1}^{n} K_{\boldsymbol{H}}(x - \boldsymbol{X}_i) \tag{2.16}$$

Where

$$K_{\boldsymbol{H}}(x) = |\boldsymbol{H}|^{-1/2} K\left(\boldsymbol{H}^{-1/2} x\right) \tag{2.17}$$

$\boldsymbol{H}$ is a symmetric positive definite $d \times d$ matrix called the *bandwidth matrix*, and K is a d-variate kernel function satisfying

$$\int K(x)dx = 1 \tag{2.18}$$

As mentioned, the choice of bandwidth in this method is complex and sensitive. Therefore, Cacoullos (1966) assumed a single bandwidth estimator by assuming $\boldsymbol{H}$ to be of form $h^2\boldsymbol{I}$, leading to the general formula

$$\hat{f}(x, h) = n^{-1}h^{-d}\sum_{i=1}^{n} K\{(x - \boldsymbol{X}_i)/h\} \tag{2.19}$$

The kernel function is usually chosen to be a d-variate probability density function. There are different techniques to generate multivariate density functions based on a symmetric univariate kernel. However, the most popular choice for K is the standard d-variate normal density (Wand, Jones 1995), which is in the form

$$K(x) = (2\pi)^{-\frac{d}{2}} \exp\left(-\frac{1}{2}x^T x\right) \tag{2.20}$$

The kernel estimator is formed by centering a bivariate kernel function on each of the data points. Then contours are formed on the biplot with the height of the contour averaged from the kernel density estimate (Wand, Jones 1995). An example is shown for a bivariate dataset in Figure 2. 6.

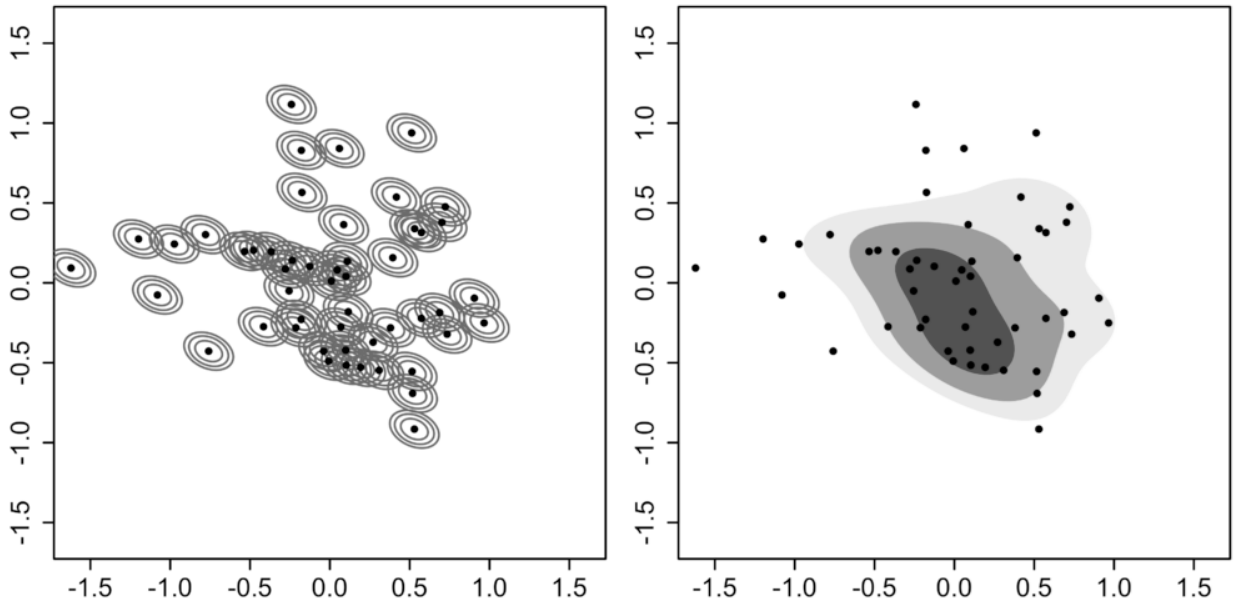

Figure 2. 6 - Bivariate kernel density estimation (Wand, Jones 1995)

2.3.2.3. Binned kernel density estimator

Computation proves to be an issue in practical calculation of kernel estimators. Considering the problem of obtaining a kernel density estimate over a space with M grid points $g_1, \dots, g_m$, a direct approach is possible due to explicit formulation of the kernel estimator (Wand, Jones 1995). The direct approach is calculating

$$\hat{f}(g_j; h) = n^{-1} \sum_{i=1}^{n} K_h(g_j - \boldsymbol{X}_i), \quad j = 1, \dots, M \tag{2.21}$$

This calculation requires nM kernel evaluations to compute the whole distribution, and therefore is computationally expensive (Wand, Jones 1995).When working with large samples, in order to increase the computational speed of the calculation, kernel estimator is replaced by approximations.

The main idea behind such approximations is to replace data with weight factors denoted by c_j, which is a representation of the amount of data that is near g_j. The *binned kernel density estimator (BKDE)* is one of the ways to approximate the kernel estimation. It uses a discrete convolution structure that can be computed quickly using the fast Fourier transform, which results in immense reduction in number of kernel evaluations (Wand, Jones 1995).

Supposing that all of data is contained in an interval $[a, b]$, after discretizing or *binning* the data based on the proximity of the data to an equally space grid, and storing the grid count values in $c_1, \dots, c_M$, The BKDE distribution is calculated (Silverman 1982)

$$\tilde{f}_j = \sum_{l=1-M}^{M-1} c_j - l\kappa_l \,, \quad j = 1, \dots, M \tag{2.22}$$

Where

$$\kappa_l = n^{-1} K_h \left(\frac{(b-a)l}{M-1} \right) \tag{2.23}$$

2.3.3. Artificial Neuron Networks

Artificial Neuron Networks (ANNs) are circuits designed for two distinctive purposes. First, which is the origin of the term, is an attempt to model the actual neurons in an actual brain to describe biophysical phenomena happening to the signals and identifying primitive elements of information process inside the brain. The other broader purpose, is to develop heuristically conceived devices and methods inspired by the biological function of simple components of the brain. Most of the studies tracing back to the early days of the field, focus on the former, whereas the present day studies tend to focus on development of information-processing devices (Kohonen 2001).

The study of ANNs is possible to undertake with three different approaches: *models*, *paradigms*, and *methods*. A *model* is a finite set of variables and their interactions that describe a real system, according to basic simplified laws of nature. A *paradigm* is an approach to simplify complex problems, with directing the selection of variables and setting up the models, as well as interpretation of data. In other words, a paradigm is a case study of a general theory. A *method* is a set of directions that is developed with sole focus on effectiveness in application, with no regards to whether it is developed based on a model or a paradigm, or neither (Kohonen 2001).

While the line between these approaches is often blurred in contemporary theoretical research, it comes down to two main aspects: the *scholars*, who are trying to understand the brain more

deeply, as opposed to *inventors*, who are focused on developing new technologies (Kohonen 2001).

Kohonen (2001) categorizes ANN in its pure form into three main groups: signal-transfer networks, state-transfer networks, and competitive-learning networks. Additionally, many mathematical statistics methods such as PCA and ICA (independent component analysis) are considered to be ANNs. Many ANN realizations have been proposed for statistical techniques (Oja 1992).

Competitive-learning networks are a category of ANNs that is based on competition between the cells. Every cell receives identical information, where they compete on that information. There is one winner cell, which acquires full activity. This results in all the other cells' activity being suppressed, making the winner the only cell that is learning the input. Through repetition of these competitions for every input, every cell develops a sensitivity for a different domain of signal values (Kohonen 2001).

Kohonen (2001) formalized an algorithmic form in order to create globally ordered maps of various sensory features onto a layered neural network. This algorithm, called *The Self-Organizing Map (SOM)*, is defined as *"an 'elastic net' of points (parameter, reference, or codebook vectors) that are fitted to the input signal space to approximate its density function in an ordered fashion" (Kohonen 2001, p. 86).* The definition suggests that like many other clustering techniques, SOM attempts to create abstraction, specifically through reduction of high dimensional data to two-dimensional visualization.

2.3.3.1. Classical Vector Quantization

SOM is built upon the idea of feature sensitive filter by competitive learning, first introduced in scalar form by Lloyd (1982) and in vector form by Forgy (1965). The idea, developed to the *classical vector quantization (VQ)*, is a standard technique in modern digital signal processing. VQ partitions the space of input data into a finite number of regions, each defined by a single model vector, known as the *codebook vector*. The goal in mind for choosing the codebook vectors is to minimize the mean distance of input data from the best-matching codebook vector (Kohonen 2013).

Considering the input data as n-dimensional vectors denoted by $\boldsymbol{x}$, codebook vectors by $\boldsymbol{m_i}$, and the index $\boldsymbol{c}$ denoting a particular vector $\boldsymbol{m_c}$, the one with the smallest distance from $\boldsymbol{x}$. $\boldsymbol{c}$ is found as

$$c = \underbrace{argmin}_{i} \{\|\boldsymbol{x} - \boldsymbol{m_i}\|\} \tag{2.24}$$

And the mean quantization error is defined as

$$E = \int_V \|\boldsymbol{x} - \boldsymbol{m_i}\|^2 p(\boldsymbol{x}) dV \tag{2.25}$$

Where $p(\boldsymbol{x})$ is the probability density of $\boldsymbol{x}$ and V is the data space. This function, although in a highly non-linear gradient-descent procedure, converges to a local minimum (Kohonen 2013). Additionally, Linde et al. (1980) showed that for a finite dataset, a batch computation method is possible to formulate.

2.3.3.2. Self-Organizing Maps

Kohonen (2001) formulated Self Organizing-Maps (SOM) as a successor to VQ, with the addition of spatial and global order to the SOM nodes (equivalent of codebook vectors in VQ). In order to elaborate the learning principles and the mathematics of the algorithm, Kohonen (2013) expresses underlying central idea as follows. *"Every input data item shall select the model that matches best with the input item, and this model, as well as a subset of its spatial neighbors in the grid, shall be modified for better matching" (Kohonen 2013, p. 53).*

The SOM method is formulated in two main algorithms; The Original, stepwise recursive algorithm, and the batch computation algorithm. The recursive method is only developed for technical reasons and comparison purposes with other methods. Therefore, the batch computation version is the method that is used for practical computations. The initial representation of the algorithm is however still more tangible using the stepwise form (Kohonen 2013).

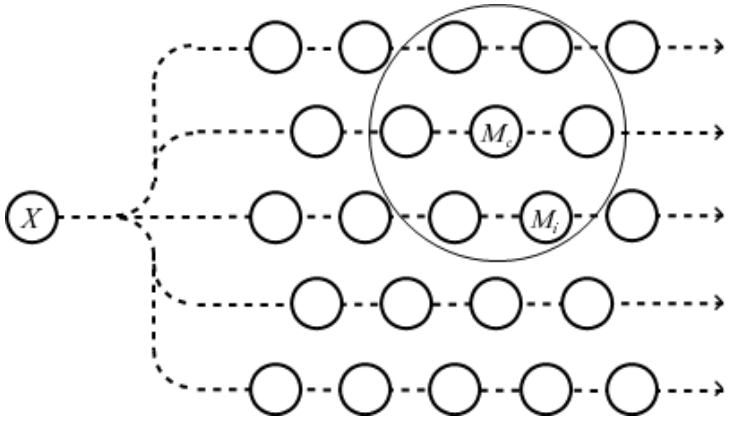

Figure 2. 7 - visual representation of a self-organizing map. M_c is the best matching unit among the set of models M_i that X is presented to, and the models in its neighborhood (larger circle) match better with X that the rest of the models (Kohonen 2013).

Figure 2. 7 Illustration of a self-organizing map. An input data item X is broadcast to a set of models Mi, of which Mc matches best with X. All models that lie in the neighborhood (larger circle) of Mc in the grid match better with X than with the rest.

The original, stepwise recursive SOM algorithm is first discussed. As seen in Figure 2. 7, the input data and the model vector are considered a sequence consisting t steps of n-dimensional vectors $\boldsymbol{x}$ and $\boldsymbol{m}_i$, in the form of $\{\boldsymbol{X}(t)\}$ and $\{\boldsymbol{m}_i(t)\}$, respectively. The formulation of the original SOM for finding the self-organized map is

$$\boldsymbol{m}_i(t+1) = \boldsymbol{m}_i(t) + h_{ci}(t)[\boldsymbol{x}(t) - \boldsymbol{m}_i(t)] \tag{2.26}$$

In this equation, $h_{ci}(t)$, the so-called *neighborhood function*, is the equivalent of a kernel in smoothing methods. Similar to VQ, the subscript c identifies the winner node, where

$$c = \underbrace{argmin}_{i}\{\|\boldsymbol{x}(t) - \boldsymbol{m_i}(t)\|\} \tag{2.27}$$

In every recursive step, first $\boldsymbol{x}(t)$ selects a winner node according to (2.27), and the model at the winner node and the nodes in its spatial neighborhood are updated according to (2.26).

The function $h_{ci}(t)$ plays the most important role in self organization of the algorithm, as it determines the rate of changes in node positions. There are many choices for the form of $h_{ci}(t)$, such as a circle around the winner, or in its most simple form, $h_{ci}(t) = 1$. A popular choice for this function is

$$h_{ci}(t) = \alpha(t) * \exp[-sqdist(c,i)/2\sigma^2(t)] \tag{2.28}$$

where $\alpha(t)$ and $\sigma(t)$ are monotonic decreasing scalar functions of t, and $sqdist(c,i)$ is the square of geometric distance between c and i in the grid. $\sigma(t)$ is chosen in a way that is large in the beginning, around half of the diameter of the grid, and it gradually decreases in a manner that after around 1000 it is a fraction of it, but never reaches zero.

For the algorithm to converge to a stable state, it should be shown that the values of $\boldsymbol{m}_i(t+1)$ and $\boldsymbol{m}_i(t)$ will be equal for t leading to infinity, while the neighborhood function is non-zero. Comparing this to (2.27) results in the requirement of

$$\forall i, \qquad E_t\{h_{ci}(t)[\boldsymbol{x}(t) - \boldsymbol{m}_i(t)]\} = 0 \tag{2.29}$$

Where E_t is the mathematical expectation value operator over t. The independent value of $\boldsymbol{m}_i(t)$ for convergence is denoted by $\boldsymbol{m}_i^*$, and it can be shown that

$$\boldsymbol{m}_i^* = \sum_t h_{ci}\boldsymbol{x}(t) \Big/ \sum_t h_{ci} \tag{2.30}$$

The equation (2.30), although implicit, due to dependence of c to $\boldsymbol{x}(t)$, is considered as the basis for the iterative solution of $\boldsymbol{m}_i(t)$, or the batch computation algorithm. This equation is simplified by substituting $\boldsymbol{x}(t)$ with mean value of points that are closest to each node

$$\boldsymbol{m}_i^* = \sum_j n_j h_{ji}\boldsymbol{x}_{m.j} \Big/ \sum_j n_j h_{ji} \tag{2.31}$$

Where n_j and $\boldsymbol{x}_{m.j}$ the number and the mean vector of the inputs that have the model $\boldsymbol{m}_i^*$ as their closest, respectively.

SOM can organize itself regardless of the initial values for $\boldsymbol{m}_i$. However, since the algorithm is often desired to converge as quickly as possible, an optimal choice of initial values is a matter of discussion. A regular two-dimensional sequence of vectors along a hyperplane resulted from the first two principal components are suggested by Kohonen (2001).

The batch computation algorithm of SOM is as follows. In every step, each input vector is compared to all of the model vectors, adding to the sublist of the winner node. To simplify, the neighborhood function here is considered to have value 1 for the neighborhood of each node, and value zero for the other nodes. Then, the mean of every vector present in the sublist corresponding to every node, as well as the neighborhood nodes is calculated. This value is calculated for every node. These mean values then replace the old values of $\boldsymbol{m}_i$, therefore completing one step.

The mean that is calculated in this algorithm is called *generalized median*. It is defined as *"…the item that has the smallest sum of distances from the other items in the set" (Kohonen 2013, p. 58)*. This set of tasks is iterated, updating the node positions in every step, until the values reach a steady state, where they do not change in position anymore with every iteration. In practice, the means are formed as *weighted averages*. A schematic of the process is shown in Figure 2. 8. The batch computation algorithm is proved to converge by Cheng (1997).

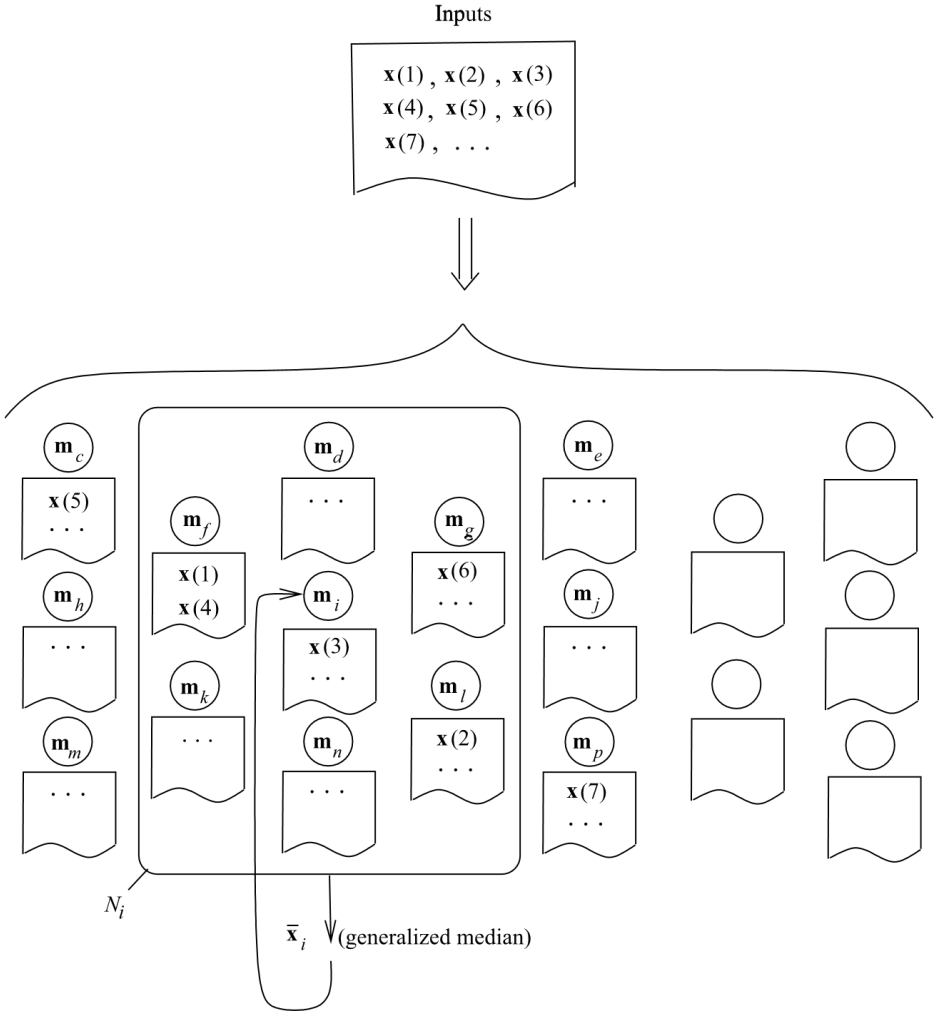

Figure 2. 8 – SOM batch process. The input x(t) is placed in a sublist corresponding to its best matching unit, and the new value of the models are calculated as generalized medians over the neighborhood sublists (Kohonen 2013).

SOM is one of the most studied and cited algorithms in recent science. There are many variations and modifications suggested for the algorithm, such as recursive self-organizing maps (Voegtlin, Dominey 2001), self-organizing time map (Sarlin 2013), and Growing Self-Organizing Map (GSOM) (Alahakoon et al. 2000).

2.3.3.3. Growing Self-Organizing Maps

Dynamic self-organizing maps with controlled growth, Also known as *Growing Self-Organizing Maps (GSOM)*, is a feature map model proposed by Alahakoon et al. (2000). The motivation for developing such algorithm is described as overcoming the limitation of the original SOM algorithm, namely the need for determination of the structure of network prior to performing the algorithm (Fritzke 1991), as well as the incompleteness of the topology preservation that is provided by SOM. Another limitation of SOM according to Alahakoon et al. is lack of awareness on the structure of the data to the user. This results in difficulties in

setting the size of the network, and realizing if the resulting cluster structure is in fact accurate in describing the structure of the data.

Alahakoon et al. (2000) argue that these problems can be solved if the shape and the size of the network are determined during the training of the network. Therefore, the GSOM algorithm was developed as the following.

In the first step, the algorithm is initialized. The weight vectors corresponding to the starting nodes are defined and the *growth threshold (GT)* is calculated based on the user requirements. This is achieved through a value called the *spread factor (SF),* which is a number between 0 and 1 that is defined by the user and determines the GSOM independent of the dimensionality of the input. GT is calculated from SF and acts as a threshold for initiating node generation. The higher the GT value, the less the map is allowed to spread out.

The second step is called the *growing phase*. It starts with presenting the input to the network. Similar to SOM, the node with closest weight vector to the input vector is recognized as the winner. Afterwards, the weight vector adaptation is applied to the neighborhood vectors, including the winner itself. This neighborhood is smaller compared to that of SOM. Also, the learning rate is reduced exponentially over iteration, which is another difference between GSOM and SOM. The weight adaptation is performed using the following formulation

$$w_j(k+1)\begin{cases} w_j(k), & j \notin N_{k+1} \\ w_j(k) + LR(k) \times (x_k - w_j(k), & j \in N_{k+1} \end{cases} \tag{2.32}$$

Where $LR(k)$ is the learning rate at kth iteration, and it is approaching zero for $k \to \infty$. $w_j(k)$ and $w_j(k+1)$ are weight vectors and N_{k+1} is the neighborhood of the winning node.

The error value for a node consists of the accumulation of the distances between the weight vector and all the input vectors that are corresponding to the said node. If this value for a node has a significant part in the total error, the area that is identified with this node is under represented. Therefore, a new node is generated in the neighborhood of the said node. Error distance is defined

$$E_i(t+1) = E_i(t) + \sqrt{\sum_{k=1}^{Dim} (v_k - w_{i,k})^2} \tag{2.33}$$

Where i denotes the node that the function is calculated for, t signifies the time or iteration, Dim is the number of variants in the input data, and w and v are input and weight vectors, respectively. This error distance accumulates for all the nodes to

$$QE = \sum_{i=1}^{N} E_i \tag{2.34}$$

Where N is the number of the nodes in the network. QE acts as a measurement to determine the need for a new node. If a node's contribution to QE is high, a new node is created in its neighborhood to share the load.

The process of adding node only happens to *boundary nodes*, the ones that have at least one neighboring position without a node. If a boundary node is chosen for addition of nodes, then every free neighboring position is filled with a new node as seen in Figure 2. 9. Although this process creates redundant nodes, it is still more practical than finding a proper position for the new node. The redundant will never win and therefore will be removed after a few iterations. In case a non-boundary node is chosen due to high error, the weight vectors are redistributed in its neighborhood.

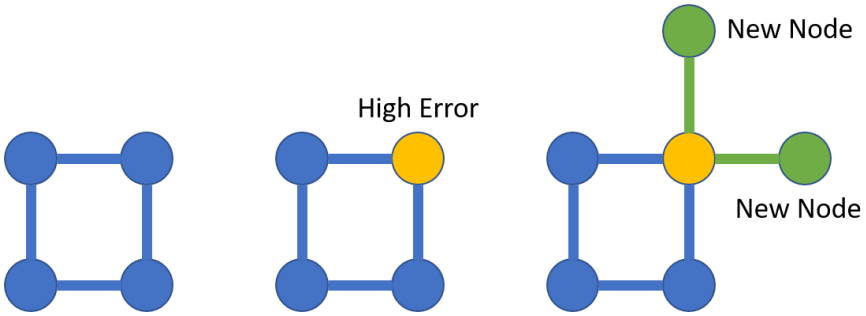

Figure 2. 9 - Node generation process in GSOM (Alahakoon et al. 2000)

Since the older nodes around the new node have already been partly organized, assuming a random weight for the new node will introduce non-matching weight vectors. To avoid that, the weight of the new node is calculated to merge with the flow of the older nodes according to guidelines by Alahakoon et al. (2000).

The growing phase is iterated until all the input values have been presented to the algorithm and the node growth is near stable. At this point the algorithm enters the *smoothing phase*¸ where the learning rate is reduced and the weights for the winner nodes and their neighborhoods are adapted. The introduction of new nodes stops at the beginning of the smoothing phase. The

smoothing phase and subsequently the algorithm converges when the error values of the nodes reach very small values (Alahakoon et al. 2000).

The theoretical advantages of GSOM compared to the original SOM and other similar algorithms are described by Alahakoon et al. (2000) as learning rate adaptation, localized neighborhood weight adaptation, and error distribution of non-boundary nodes.

3. Development of fault detection method

This chapter is dedicated to in-detail description and evaluation of the novel fault detection method that is developed in this study. The main aim of this method is to be used as a basis for asset health monitoring. Proper health monitoring can result in optimizing maintenance schedules as well as avoiding unplanned downtimes.

Predictive maintenance in general has always been a subject of interest to chemical and process industries due to its money saving potential. Asset health monitoring specifically plays an important role in improving overall plant reliability, and therefore has received growing attention in the industry in the recent years. However, this attention is focused on a very small group of assets called *critical assets*.

Critical assets in chemical industry are those that are either critical in terms of safety and security, or the ones that are financially sensitive and have an important role in the profitability of the plant. These assets are usually monitored using highly specialized software technology, whereas many other assets of less importance are not monitored at all. The data for these assets are usually stored in a historian device where no further analysis is carried out. These assets, if monitored collectively, have a potential to improve process drastically by reducing downtime and expenses of maintenance. The fault detection method presented in this study is developed with special interest in all the assets that are not monitored in today's industry, in specific to offer a generic monitoring solution for these assets.

In order to be applicable to a wide range of assets, the method has to be developed with a data-driven approach. As opposed to model-based methods, this method does not require any specific measurement or information on the distribution of data and relations between variables. Instead, this method relies on the with the process engineer to determine and identify the sensitive variables and process states, as well as normal operation zones and failure modes.

The basic assumption underlying any data-driven method is that the relations between the measurements differ in normal and abnormal behavior. More generally, different states of a process can be traced in correlations between the measurements of the process. Any process is assumed to operate under normal conditions most of the time, which results in mostly similar relationship between measurements; therefore, searching for broken or different relationships between the variables will lead to abnormal periods of operation. In any visualization of data

there are dense areas with points showing normal operation, as opposed to more parse areas for outliers, which point to anomalies.

There are two main approaches in data-driven methods with regards to training data and pre-determined information. In the first approach, some periods of time in the process history are determined by the process expert as normal operation periods. The training of algorithms with this approach are handled with the use of the normal operation data, and resulting monitoring system utilizes control charts to identify the points as normal or abnormal. A schematic of this approach is shown in Figure 3. 1.

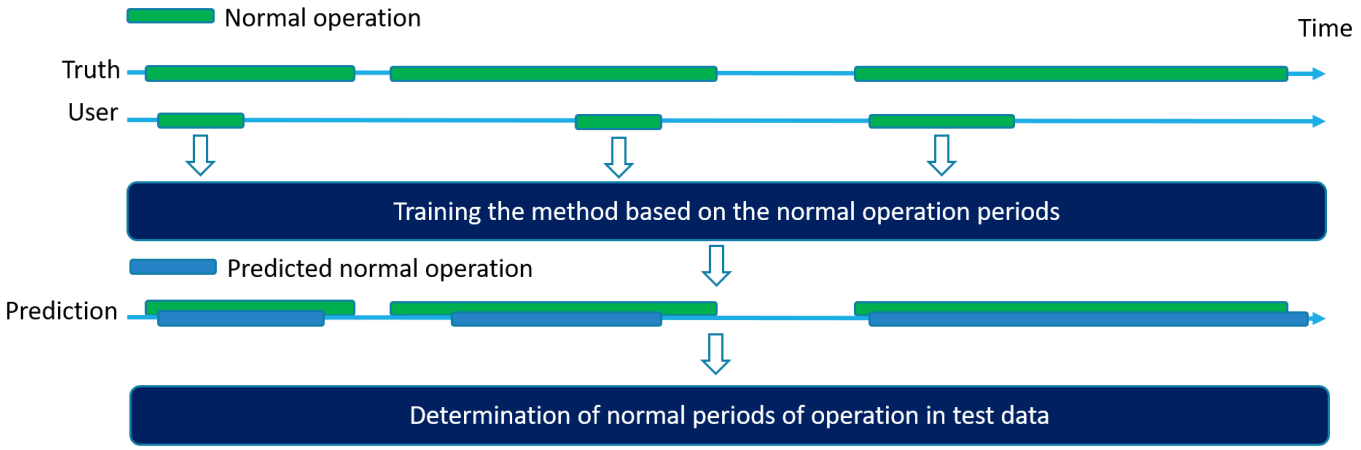

Figure 3. 1 - Normal data training approach

The second approach, assumes various operation zones that are considered normal, as well as different failure modes instead of general abnormal behavior. Methods based on this approach utilize clustering methods to separate the data into different groups. This approach is visualized in Figure 3. 2.

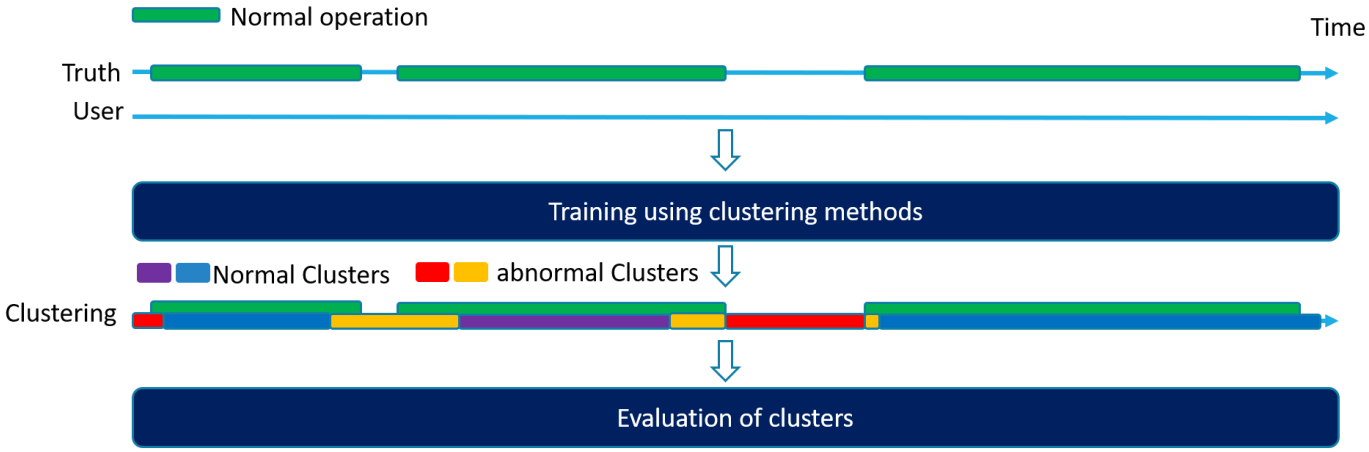

Figure 3. 2 - Clustering training approach

clustering methods are unsupervised techniques for grouping similar observations within a dataset. Here the unsupervised technique refers to a problem finding hidden structure within unlabeled data. In clustering, the identity of clusters is not pre-determined by the data scientist, therefore these methods are categorized as unsupervised (Dietrich et al. 2015).

Although the first approach performs with more stable results, it fails to explain different regimen of failure, which is an important diagnostic result. In addition, methods based on this approach are unable to detect the addition of novel failures and new normal operation zones. The clustering approach in data-driven methods makes up for these shortcomings, with the price of difficulty in interpreting results and making robust, and therefore are the basis for the fault detection method developed in this study. These methods also do not require determination of periods before training phase.

Seeing as the sensor data from assets in chemical industry usually consist of numerous variables, the process of clustering using the conventional techniques is somewhat problematic. Based on the aforementioned "curse of dimensionality" phenomenon, as the number of dimensions increases, the performance of ordinary clustering techniques degrades. A way to circumvent this problem is to use dimensionality reduction techniques, such as PCA in the linear data, and SOM based methods in nonlinear cases. These techniques are studied in more detail in Section 2.3.

The fault detection method described in this study is robust and adaptive. It truly functions in the multivariate space, making use of the latest developments in self-organizing maps. This method makes monitoring of highly non-linear data in multiple regimen systems possible, since it does not make any assumption the distribution of the data.

Another characteristic of the fault detection method is that this method is designed to perform the analysis with no regards to the type of the measurements available. Although many methods for monitoring any specific asset is developed and available to use, these methods usually require a specific set of measurements for each asset. On the other hand, these assets are often not equipped with the sensors necessary to provide the measurements for the methods, especially if the asset is not a central part of the process. It should be noted that failures in these assets can be as harmful to the process as the central assets.

According Himmelblau (1978), when carrying out the fault detection tasks using a computer, is that an important point for the algorithms to function properly is the requirement of predefining the possible malfunctions to the system. However, if an operator is monitoring the

process and diagnoses a period of process as faulty, the two stages of definition and classification do not have to carried out consequentially, as they do in the case of complete automated monitoring. This is an important point taken into consideration in developing this method. Therefore, close attention to the analysis by the operator will be beneficial in defining novel faults and unexpected anomalies.

3.1. Fault detection method

In this chapter, the novel method of data-driven fault detection for asset monitoring is presented in detail. A sample data frame used to illustrate the results. The data frame is one of many samples that is included in R, called 'wine'. These data are the results of a chemical analysis of 178 different wines grown in the same region and derived from different cultivars in Italy. The analysis determined the quantities of 13 constituents found for these wines. The correlations between the variables can be seen in Figure 3. 3.

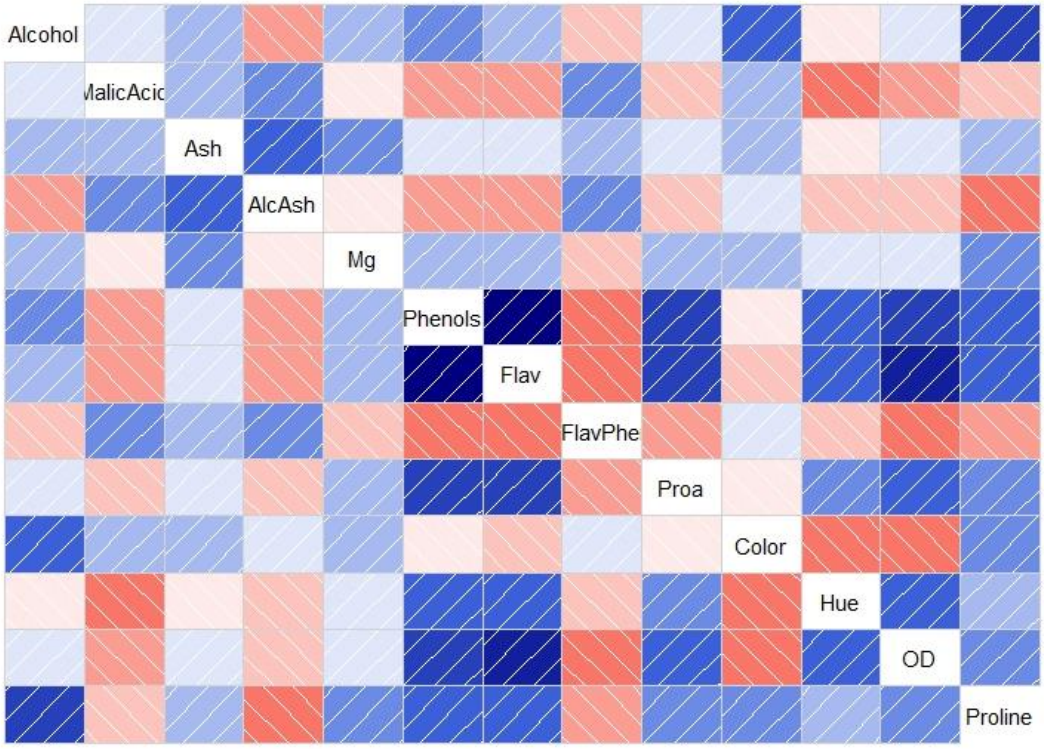

Figure 3. 3 - Wine data correlogram. Color blue denoted positive correlation and red denotes negative correlation. The intensity of the colors is an indication of the value, with higher values of correlation having more intense colors.

The algorithm is presented in a step-by-step approach.

3.2.1. Pre-processing and parameter definition

In the beginning, the data is introduced to the algorithm. The ideal data for this algorithm consists of a data frame consisting of data points that are used for training the algorithm, as well as a vector of the same length containing the date and time for the measurement of each data point. Additionally, a data frame containing some data points that signify known failure periods can be introduced to system to verify the results.

As pre-processing, the data frames are scaled to have a zero mean and standard deviation one for each column (variable). This step results in the elimination of the influences of physical nature of measurements in data variation. Therefore, the data is in a physical sense dimensionless, and every variable in the data is of the same scale and can be treated the same manner.

Some of the analysis parameters are defined in this step as well. These parameters needed for the analysis are

1. Threshold for PCA control charts denoted by α (implemented in the code as $a = 1 - \alpha$), which has a default value 0.05.
2. BKDE anomaly score threshold denoted by **AT**, with a normal value 3.
3. Spread factor for GSOM, with a range between 0 and 1 and a default value 0.9.

It should be noted that due to formulation of these techniques, the value of these parameters usually does not affect the analysis noticeably, and therefore the default values are assumed to provide the desired results. However, optional fine tuning is possible in order to optimize the clustering to explain the physical phenomena.

3.2.2. Kernel Density Estimate

In the first step of the analysis, a series of 2D binned kernel Density Estimates are performed, and the distribution results are stored for later purposes. There will be a distribution for every possible combination of two variables in the data frame. For a data frame with p variables, these distributions will be stored in $\hat{f}_k$, where $k \in \{1, \dots, \binom{p}{2}\}$. For example, in the wine sample data, there will be $\binom{13}{2} = 78$ distributions produced and stored.

As it was mentioned in Section 2.3.2, it is required to define two bandwidth values for the BKDE analysis. For a data set containing n points each with p variables, that has every point of each variable stored in $\{\boldsymbol{Y}_i, i \in (1, \dots, p)\}$ these values are set as

$$h_i = Standard\ dev.(\boldsymbol{Y}_i)/n^{1/6}$$

Linear binning is used to obtain the bin counts and the Fast Fourier Transform is used to perform the discrete convolutions. For each x_1, x_2 pair the kernel is centered on that location and the heights of the kernel, scaled by the bandwidths, at each data point are calculated. The output $\hat{f}$ value at every location is the normalized sum of these heights. A bivariate Gaussian kernel is used for the algorithm.

A 256×256 grid is set for the estimation, meaning that the probability value will be calculated for 256 equally spaced points in each direction. A measurement called anomaly score denoted by AST is defined for every point in each distribution as

$$AST = -\log \frac{\hat{f}(x_1, x_2)}{\max(\hat{f})}$$

In order to visualize the BKDE distributions, they are plotted in the form of contours based on their AST values with the data points. These plots are helpful in visualizing the operation zone of a process, especially if the training data only contains the normal operation data. An example for wine sample data is shown in Figure 3. 4. Such heatmaps can be produced for every combination of variables.

The heatmaps produced by kernel estimation, as shown in Figure 3. 4, are bounded within visible boundaries. As mentioned before the BKDE is calculated within a grid, that is defined by the algorithm based on the limits of the data points. The estimation is not calculated outside of this grid, and that is the reason for visible boundaries of the maps.

The heatmaps show a probability of points being located all around the plot with contours and colors, which can also be used as a basis for evaluating the anomaly of a point, given all the points in the training set are considered normal.

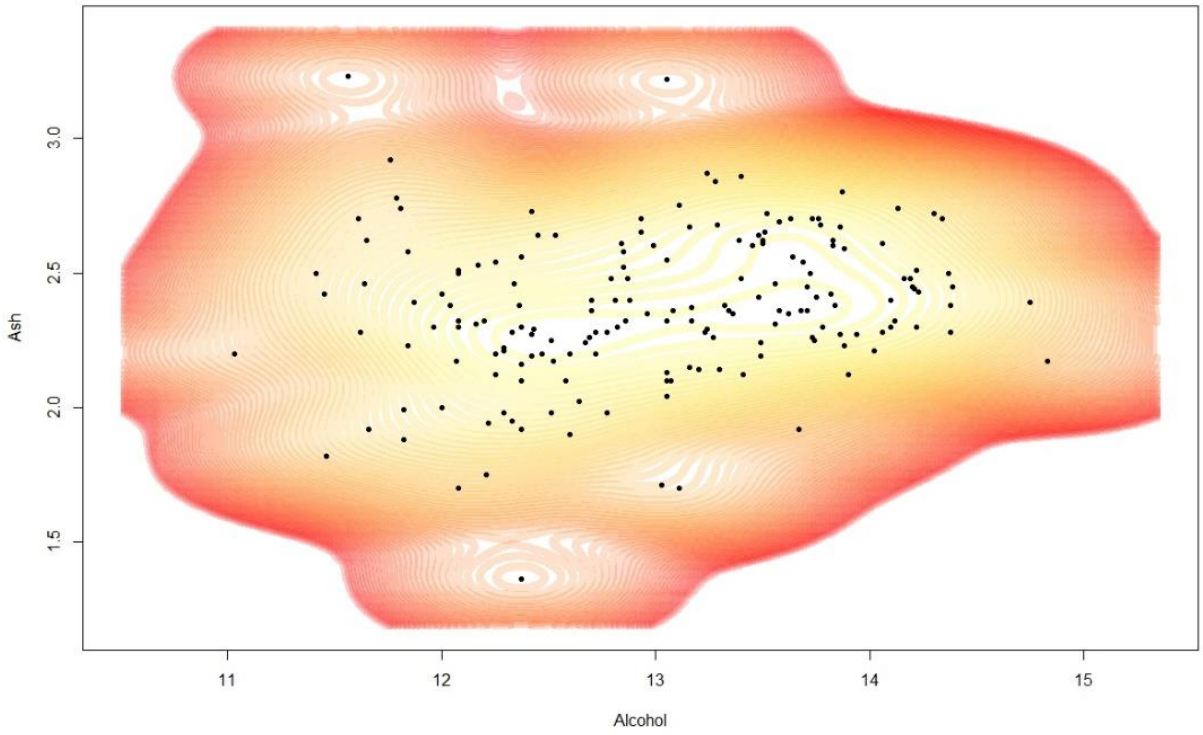

Figure 3. 4 - BKDE based heatmap for wine data

The BKDE distributions are stored for later comparisons and analysis.

3.2.3. Growing self-organizing map and clustering

In this step, the growing self-organizing maps algorithm is solved for the scaled data frame. The only input needed for the calculation is the Spread factor, as described in Section 2.3.3. The training progress for the wine sample data is shown in Figure 3. 5. A discontinuity is visible in the mean distance to unit value at the point of transition from growing phase to smoothing phase. The reason is the sudden change in the learning rate during the transition from growing phase to smoothing phase, as described in Section 2.3.3.

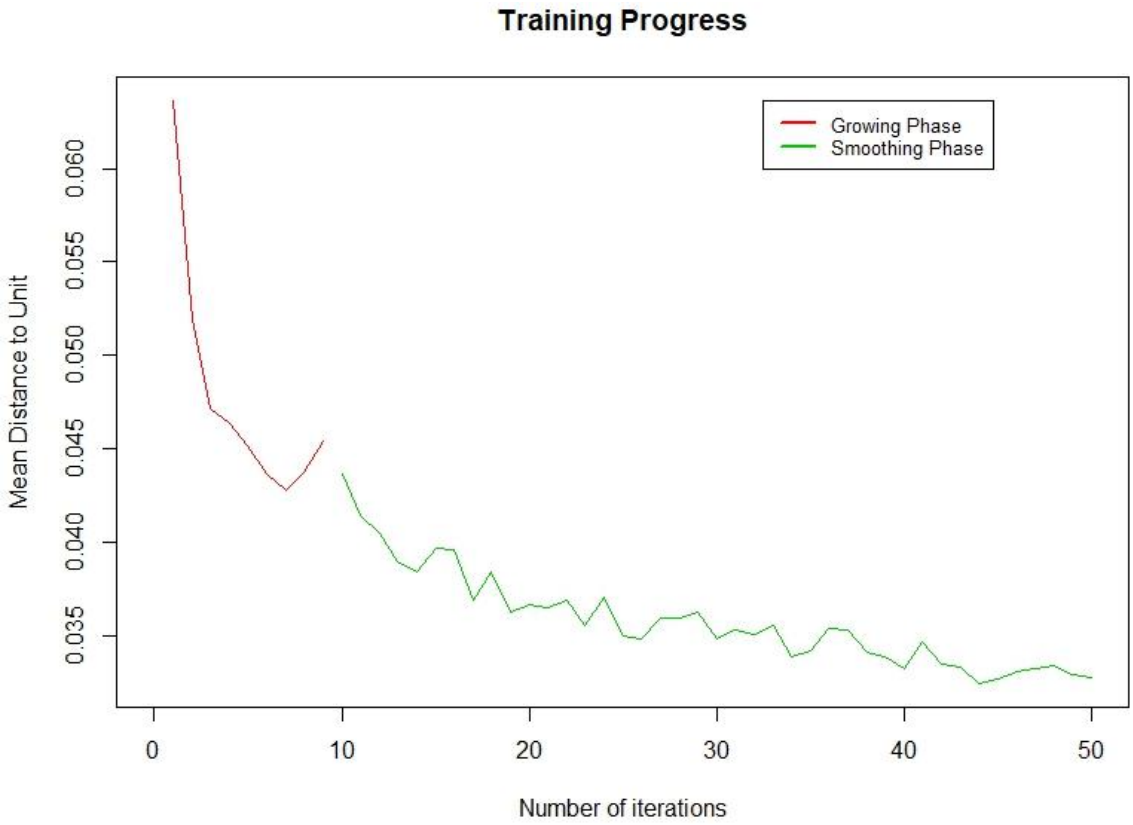

Figure 3. 5 - GSOM training progress for wine data

Figure 3. 6 shows the output of the GSOM calculation in the form of the positions of the GSOM nodes, as well as the number of points that have determined each node as their best matching unit. As can be seen, the number and the position of the nodes in GSOM are arbitrary, as described in Section 2.3.3.

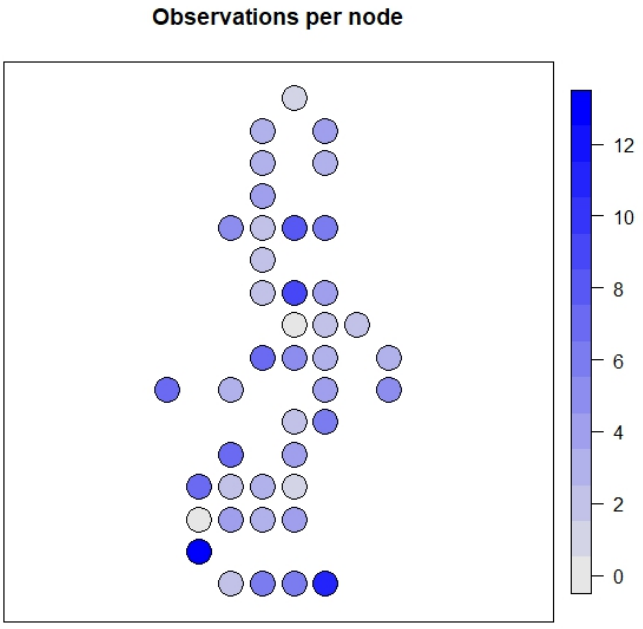

Figure 3. 6 - Wine data mapped to the GSOM

In the next step, GSOM will be categorized into a number of clusters denoted by NC. There are different ways to determine the number of clusters that is suitable for a data frame. Best would be to inquire the subject matter expert on the number of the operation zones in the process and the number of known failure modes. Then, if the training data contains normal operation only, the number of clusters will be equal to the number of operation zones since the purpose of clustering the data is in fact to find the operation zones. If the training data contains both normal operation zones as well as some failure mode data, then the number of clusters equals the number of normal operation zones with the addition of the number of known failure modes.

However, this information might not be available for a process, especially in the first time that data from a process is being analyzed. Therefore, a generic method for determining the number of clusters is suggested base on K-means clustering. Although the final clustering of the data is based on hierarchical clustering method, K-means provides a quick and feasible measurement for comparing the hypothetical cases of the clustering. In this method, the sum of squares for every group in a K-means clustering is calculated. These values are added to result one value called *Within Cluster Sum of Squares (WCSS)*. The value of WCSS is compared for different cases of clustering based on K-means in a diagram seen in Figure 3. 7 for wine test data.

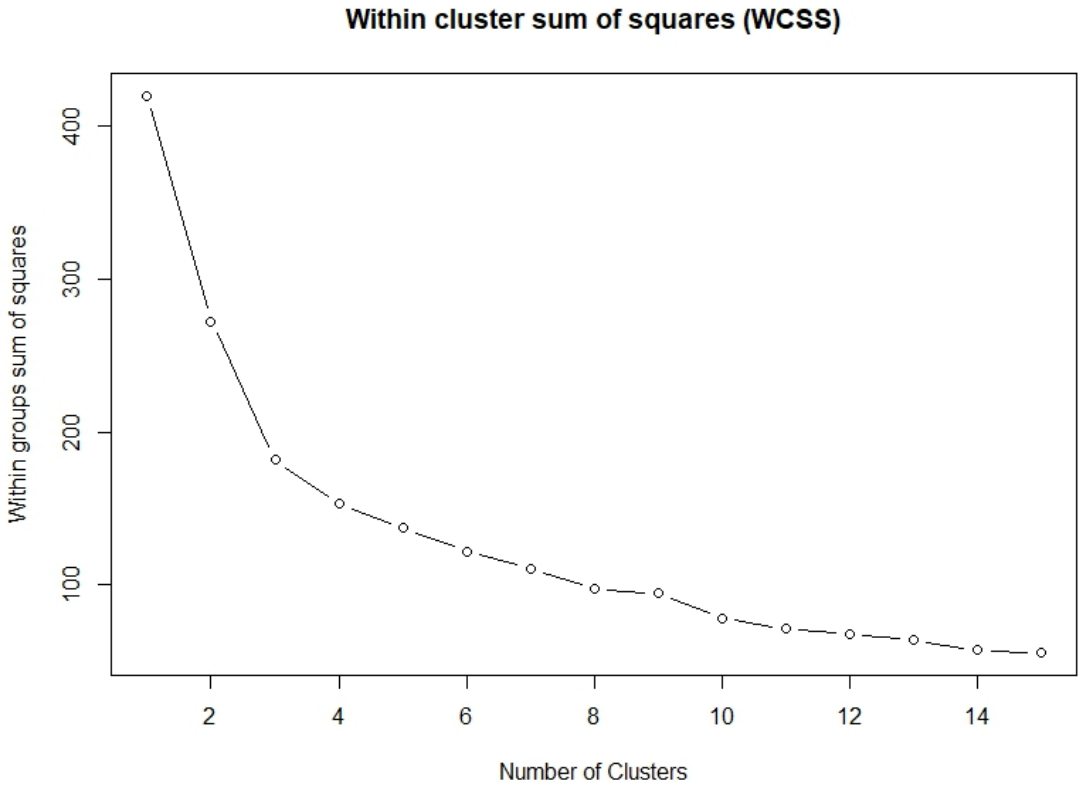

Figure 3. 7 - Estimation on the optimal number of clusters for wine data

A point with high difference in the trend before and after (elbow) on the diagram is a suitable candidate for clustering. Here for the wine sample data, three clusters are deemed suitable by this method.

Although the number of clusters is determined by K-means method, the actual clustering of the GSOM nodes, and subsequently the data is done using the hierarchical clustering method. In this method, first a function computes and returns the distances between the rows of the GSOM nodes positions matrix. This procedure produces a so-called dissimilarity structure, which is the basis for hierarchical clustering. Initially, each object is assigned to its own cluster and then the algorithm proceeds iteratively, at each stage joining the two most similar clusters, continuing until there is just a single cluster. At each stage an adapted measurement for the distances between clusters are recomputed. The result of such algorithm can be visualized in a dendrogram, as shown for the wine sample data in Figure 3. 8.

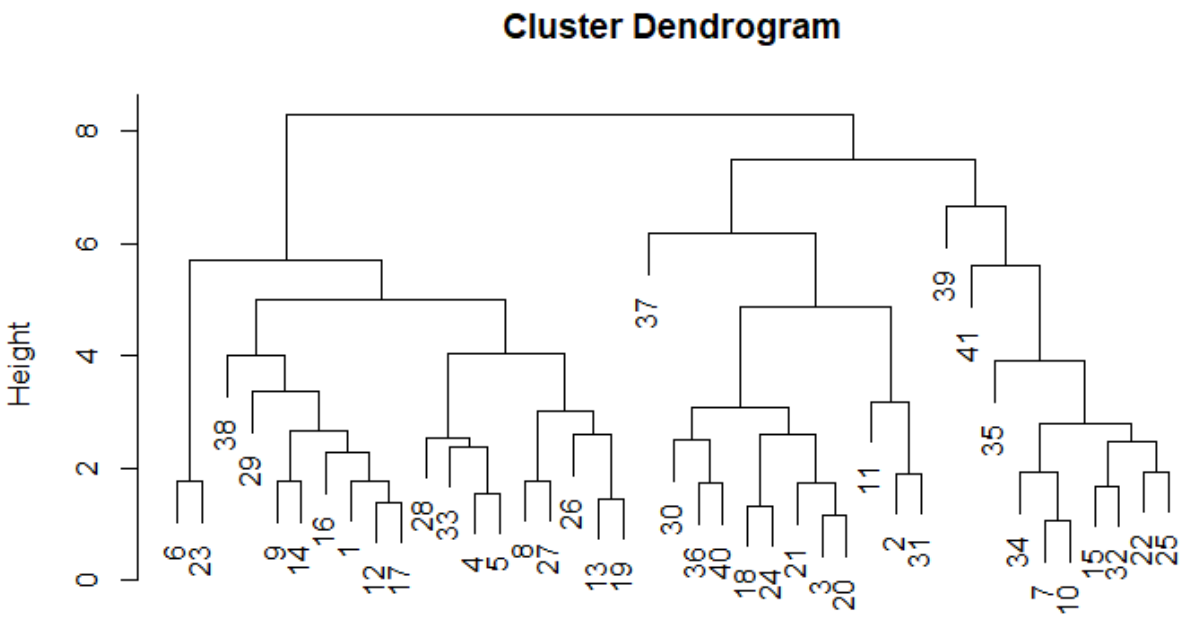

Figure 3. 8 – Dendrogram of GSOM nodes for wine data. The GSOM nodes are indicated by assigned numbers

This clustering structure is then cut into groups, where the number of groups are specified by NC. The result of clustering the wine sample data is shown in Figure 3. 9.

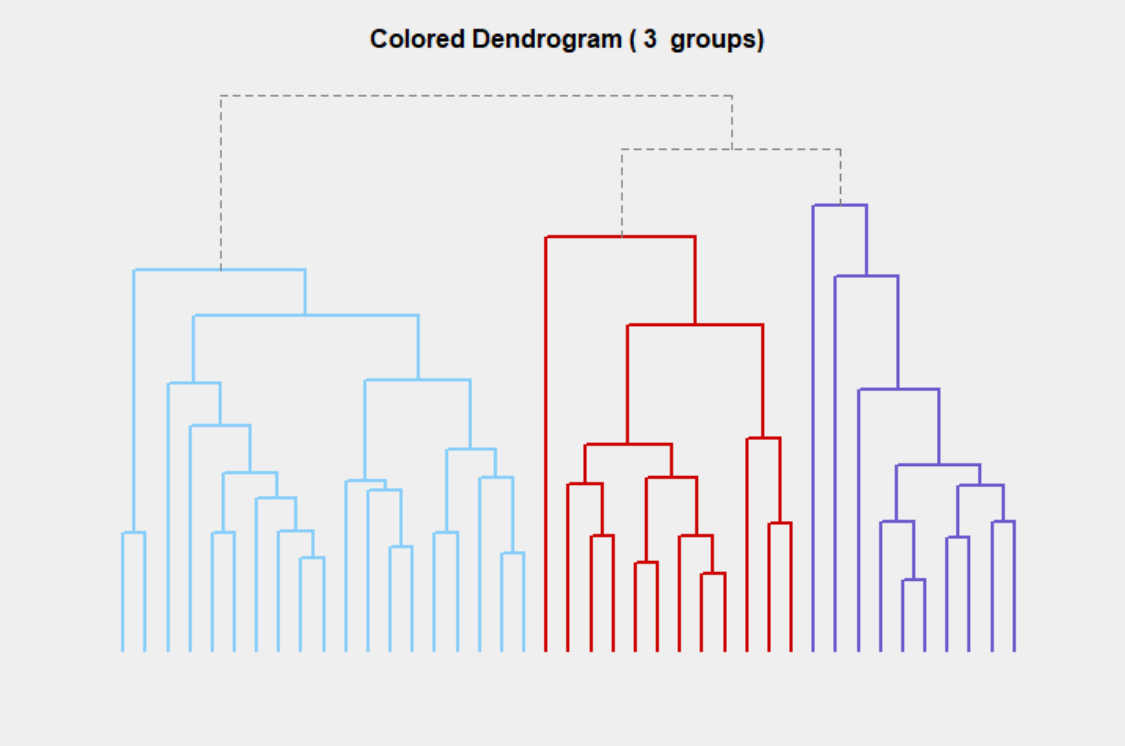

Figure 3. 9 - Hierarchical clustering of GSOM nodes for wine data

The position of the nodes with regards to the data points can be seen in two selected variables from the dimensions of the data frame in Figure 3. 10. The positioning of the nodes is in a way that more dense areas get more nodes. the clustering however, is more related to the distance between the nodes. It should be considered that this plot only shows two variables out of 13 that is measured in the wine sample data. Therefore, the multi-dimensional distance between the nodes of different clusters might not be visible. Same plot can be produced for every combination of the variable. However, visualization of such high-dimensional data is problematic and requires some imagination. Perhaps considering the multi-dimensional distance between nodes is best perceived if regarded as scalar values, therefore the clustering can be viewed as a categorization process based on these values.

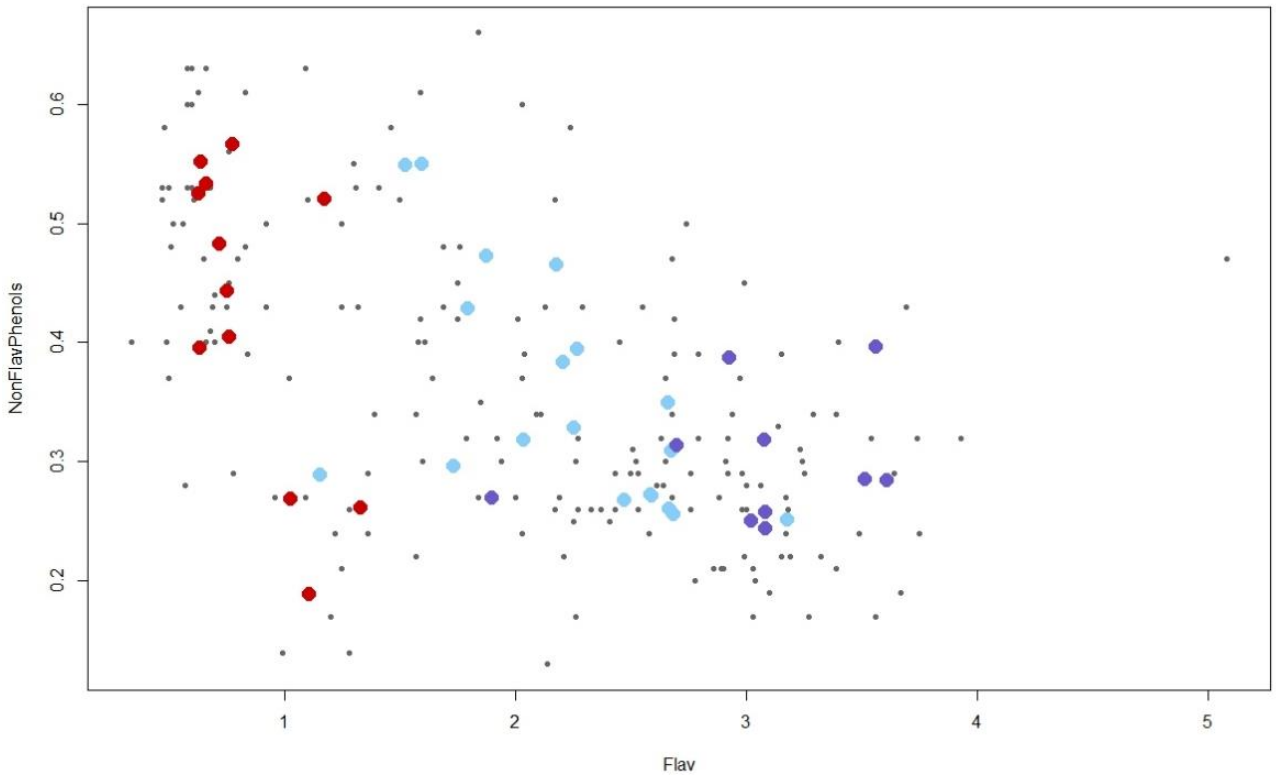

Figure 3. 10 - Position of GSOM nodes in comparison with wine data points. Each color signifies one of the clusters

This clustering is based on the distance between the GSOM nodes. To further illustrate this point, the same procedure is applied on the results of a SOM on the same data. The SOM grid size is chosen to be as close to the GSOM as possible. For the wine sample data, a 7×6 grid with hexagonal topology is chosen to match the 41 node GSOM results. The point count for each node plot is shown in Figure 3. 11.

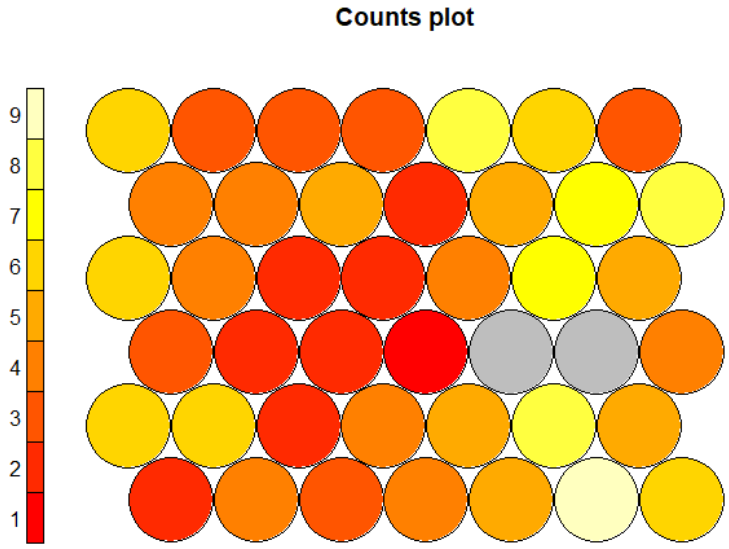

Figure 3. 11 - Wine data mapped to the SOM

As mentioned before, a major difference between SOM and GSOM, which is the order in the position of nodes in the grid can be seen when comparing the two count plots. This makes the SOM plots more suitable for visualizing the node distances, especially with regards to the clustering. The result of clustering the SOM nodes based on the same hierarchical method used in GSOM clustering for wine sample data is shown in Figure 3. 12.

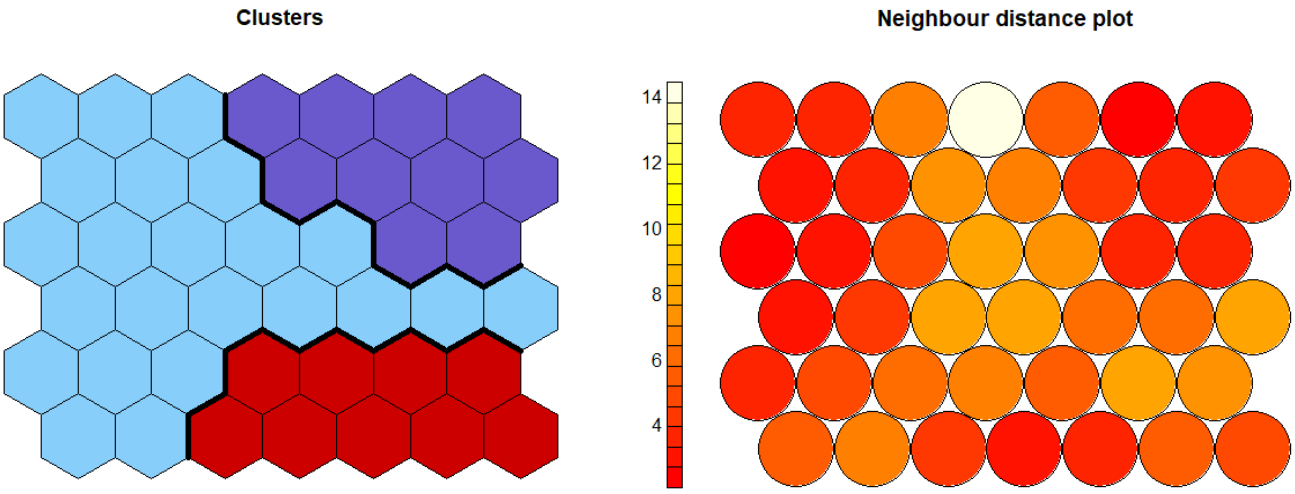

Figure 3. 12 - SOM clustering boundaries and Distance between SOM nodes

Figure 3. 12 visualizes the positioning of clusters on the nodes grid. Additionally, a comparison can be made between the clusters, and the distances between the nodes in the grid, verifying the clustering as a distance based method. The borders between the clusters signify an increased distance between the boundary nodes of each cluster.

3.2.4. Categorization of clusters

Considering that the clustering is performed using the general method, and there is no information available on the physical nature of the clusters with regards to the process operation zones, this step attempts to classify the clusters into two groups, normal operation and failure mode, based on a data-driven approach. Among various methods to evaluate the data from each cluster for abnormality, the most effective way is studying the heatmaps created by BKDE in Section 3.2.2. To make this evaluation practical, first the center of each cluster is found by calculating the arithmetic mean of the nodes in the cluster. This point is then placed on the BKDE distributions and is assigned an AST for every combination of variables. These values show how the cluster centers are positioned with regards to the correlations of each set

of two variables. A threshold denoted by *AT* is defined for a correlation to be considered broken in case of *AST* larger than the threshold value. The result of this method can be visualized in a binary board with every correlation, showing whether the correlation is held true or it is broken. Such diagram is shown for wine sample data in Figure 3. 13.

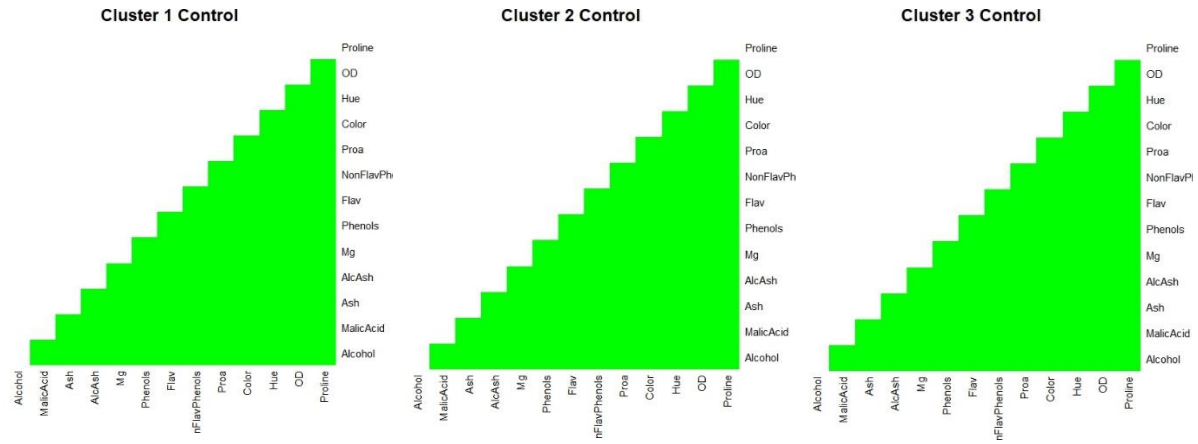

Figure 3. 13 – Correlations binary boards for wine data clusters

In this example, every correlation is held within the boundaries defined by BKDE for every cluster. As an illustration of an abnormal case, the hypothetical case of clustering wine data into 5 clusters, as shown in Figure 3. 14, produces a cluster with some abnormal correlations.

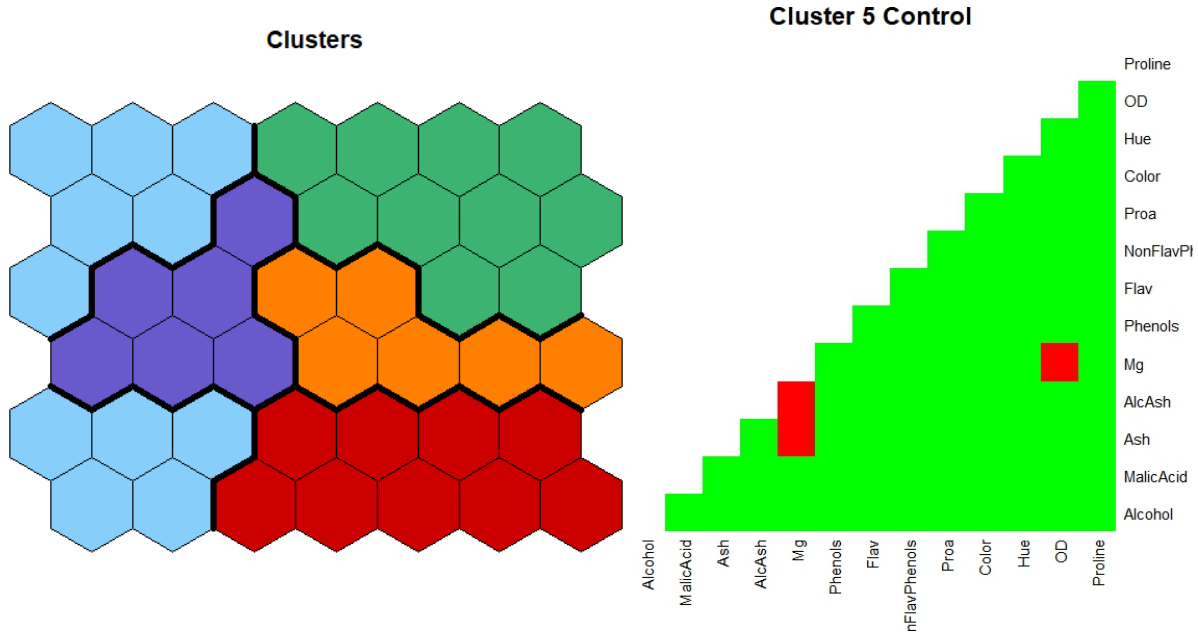

Figure 3. 14 - Clustered SOM map and correlations binary board in the hypothetical case of 5 clusters for wine data

The red lights on the binary board shows the correlation between the variable Mg and three other variables to be out of the boundaries defined by the BKDE heatmaps. A biplot relating to one of the mentioned correlations is shown in Figure 3. 15. It is visible that the position of the single node that is categorized as cluster 5 is an outlier compared to the rest of the GSOM nodes.

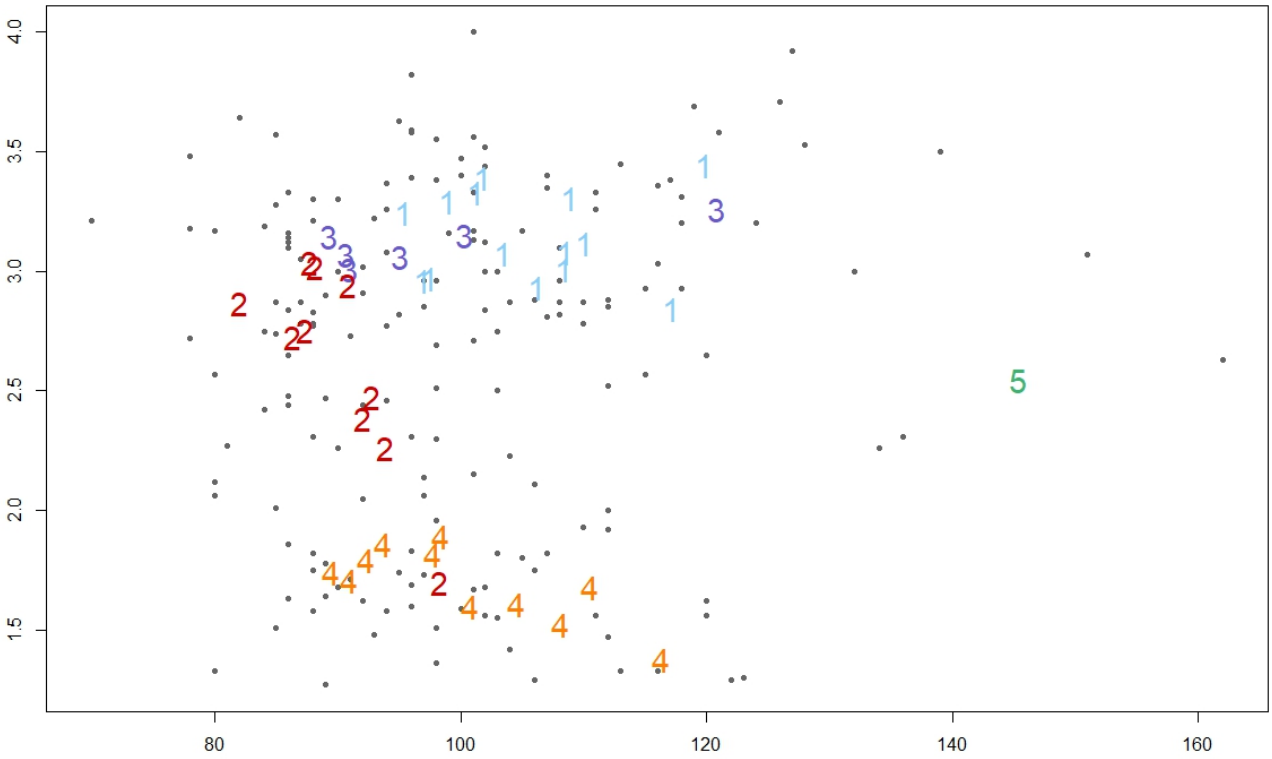

Figure 3. 15 - Position of GSOM nodes for 5 clusters in wine data. The nodes are shown with numbers of their cluster

Although this result can be assumed as an indication to the abnormality of this cluster, it is always recommended to have a subject matter expert decide which cluster are to be considered normal operation.

Supplementary information is possible to provide for aiding the decision-making process. One useful method is principal component analysis on the data frame, with color coding the clusters in the PCA results as well as the control charts. The PCA is solved for the data frame, and the principal components are determined with their corresponding standard deviations. The properties of the principal components for the wine test data is shown in Figure 3. 16.

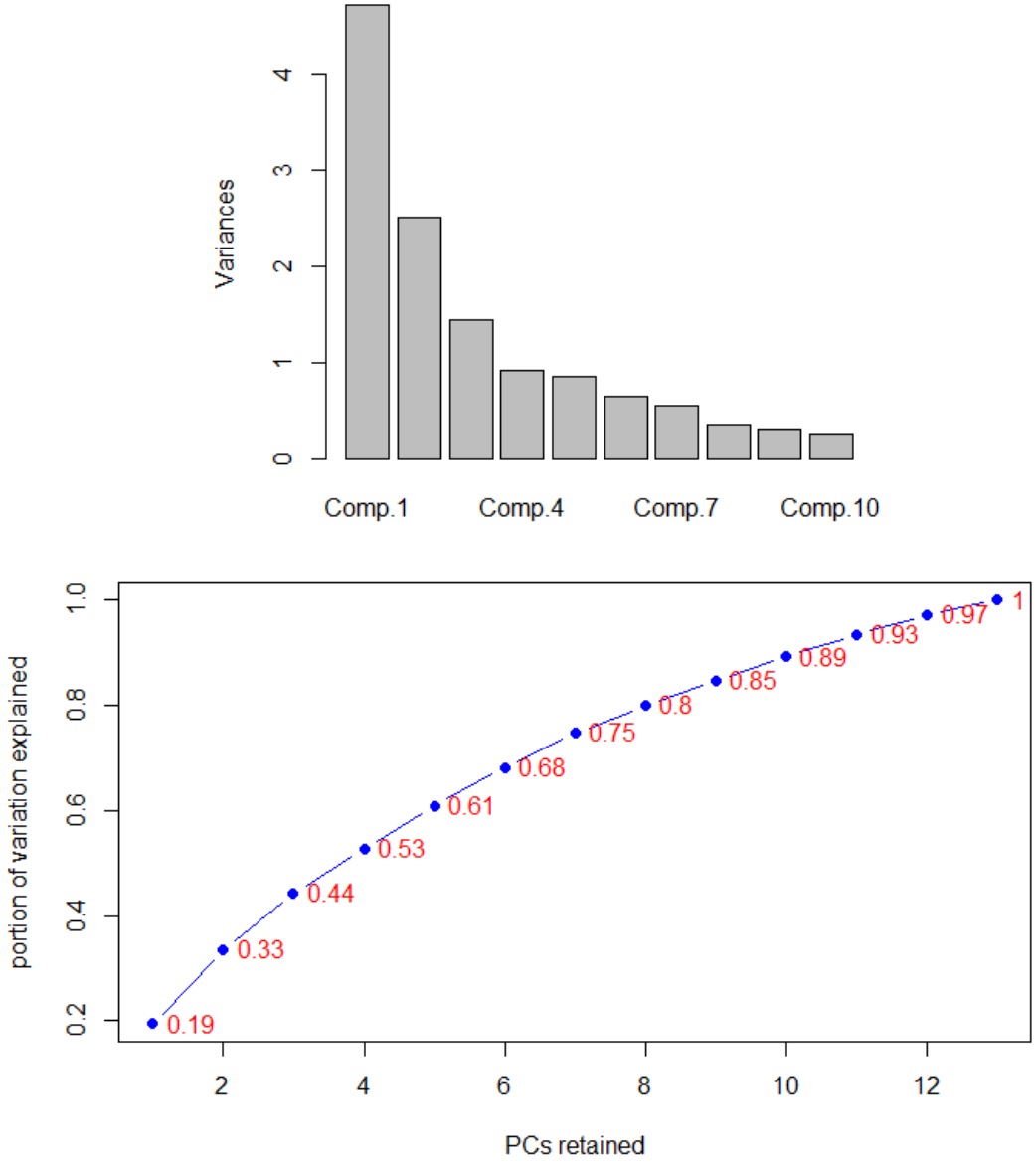

Figure 3. 16 - PCA summary for wine data. Variances of principal components and their accumulative sum

In order to determine how many principal components to retain for the calculations related to the control charts, a generic suggestion is used in this method, which is to retain the principal components with standard deviation value larger than one. Figure 3. 17 shows a total of 3 principal components that fit this criterion for wine test data, and the biplots of the correlations between these principal components with the data points colored with regards to their cluster.

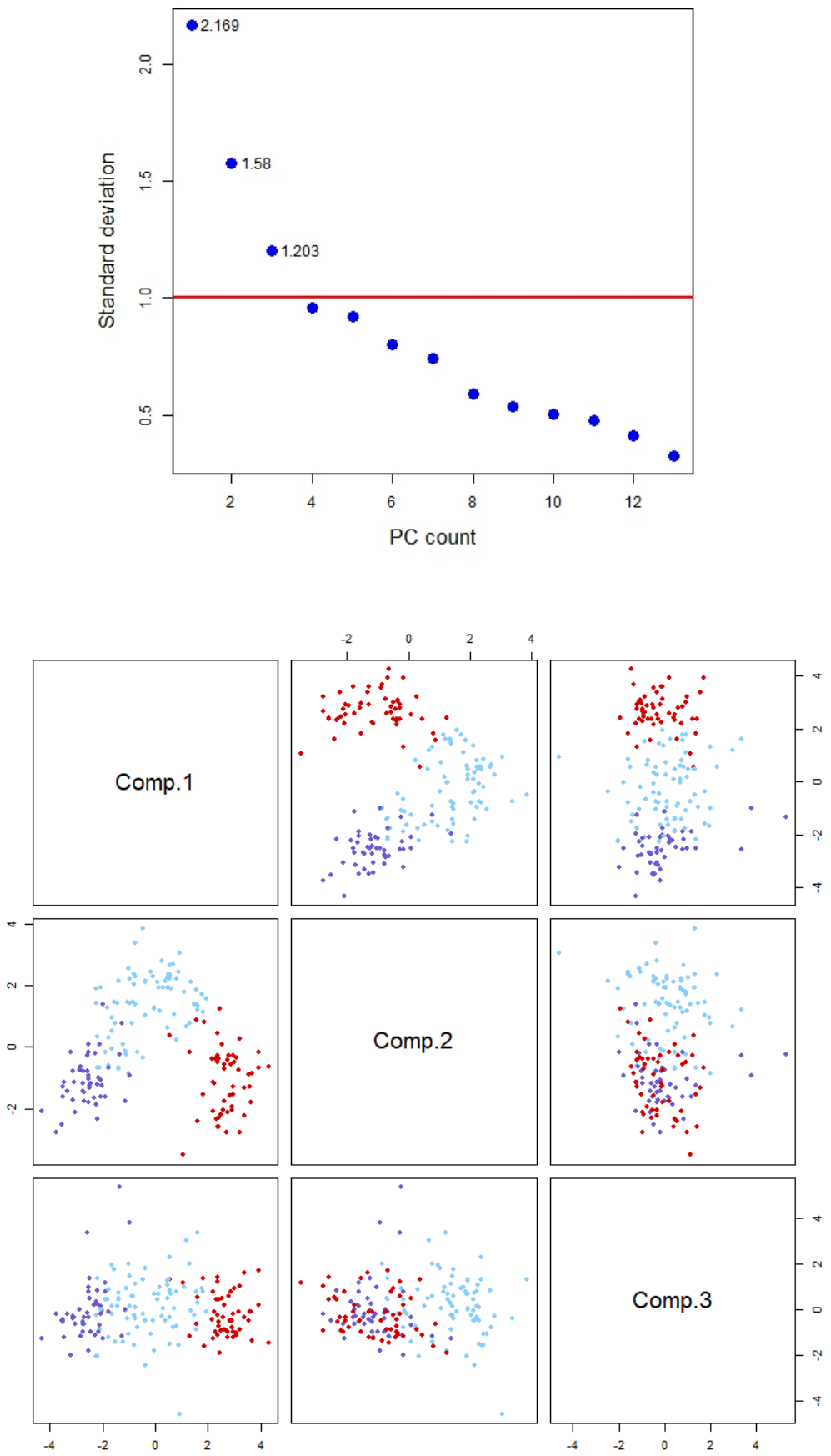

Figure 3. 17 - Retaining principal components and their biplots for wine data

A closer look at the first two principal components shows a clear separation in different clusters. This confirms that the different zones of operations as determined by GSOM clustering transforms to PCA and is also detectable in the PCA space. However, this is not true for more complex datasets, which is usually the case with chemical process data.

In the next step, T^2 Hotelling and Q-statistic control charts can be developed, since the number of retained principal components as well as PCA scores and variations are available. A value for threshold of the control charts, denoted by α is used to calculate a maximum score threshold. The Hotelling's and Q-statistic control charts is shown in Figure 3. 18.

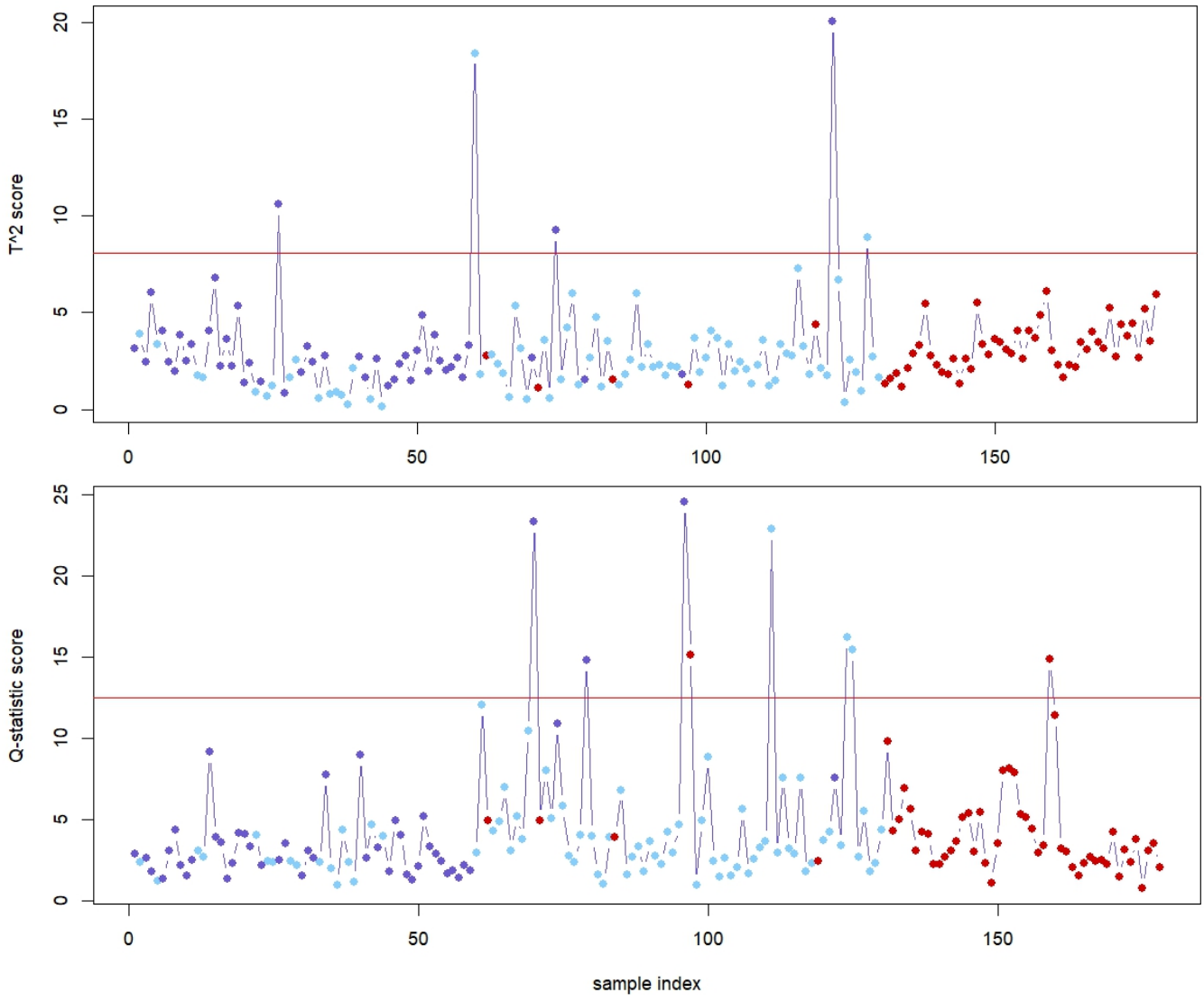

Figure 3. 18 - PCA control charts test results for wine data. Hotelling's T^2 and Q-statistics anomaly scores with different clusters marked with different colors

The anomaly scores are normalized on a logarithmic scale to transforms the control value for both of methods to zero, making a comparison possible. Such comparison is made for wine test data in Figure 3. 19.

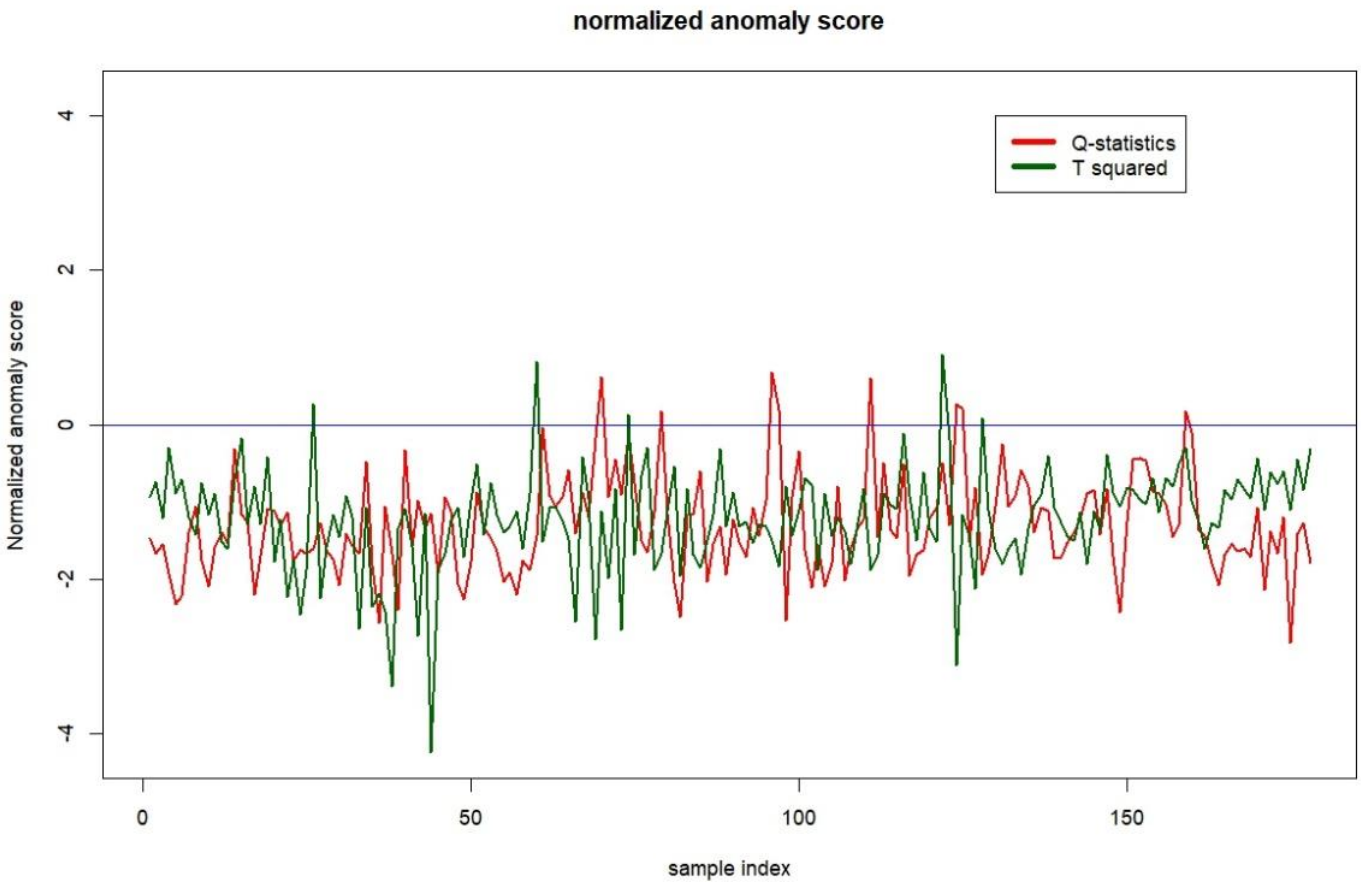

Figure 3. 19 - PCA control charts comparison for wine data

It is shown in Figure 3. 19 the results of the two tests do not necessarily agree in this case, but in a larger scale data, they show more similarity in trends.

The control charts based on PCA are useful tools in determining the abnormality of the clusters.

3.2.5. Decision making within clusters

At this point in the analysis, the clusters of the data are categorized into normal operation and failure modes, either by user specification or based on the binary board method, as described in Section 3.2.2. In this step, the clusters are analyzed individually based on their category.

First, a category is defined for all the data points that are considered abnormal. Every data point that belongs to an abnormal cluster already belongs to this category. On the other hand, there

might be some additional data points that have a high distance to the center of an operation zone, despite being categorized in a normal operation cluster. In order to find these points, PCA control charts are developed for every normal operation cluster individually. These control charts are allowed to have different threshold parameters, and the number of principal components retained for each cluster may vary with regards to the variation Explained by each PC in a particular cluster. The Q-statistic control charts for the first cluster of wine test data is shown as an example in Figure 3. 20.

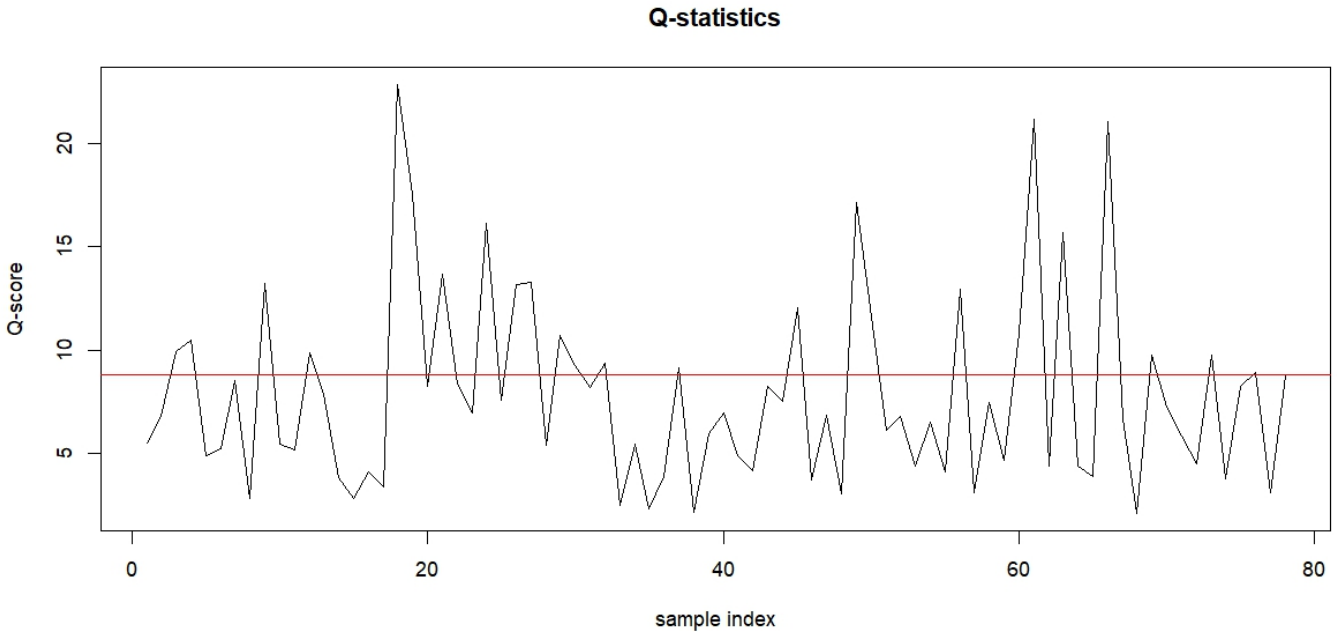

Figure 3. 20 - Q-statistics anomaly scores for wine data cluster 1

There is one point that needs to be considered while developing the control charts for clusters. If a cluster is defined as normal operation, but the number of points inside it is less than the number of variables in the data frame, it is not possible to calculate the PCA for said cluster. Therefore, this cluster must be recategorized to abnormal altogether. This abnormality is due to the scarcity of the data points that are located in positions inside that cluster, as opposed to other abnormal clusters, where the abnormality is the result of correlations being broken between the variables compared to normal data. In the end, if the user still declares such cluster as normal, then more data points should be provided for that operation zone in the training data frame.

The data points of each cluster are compared to their corresponding control charts, and in case of anomaly scores higher than the threshold value, these points are added to the abnormal points group. This process completes finding the abnormal points all through the data frame.

Additionally, the BKDE heatmaps are recalculated, this time including only the normal operation data clusters. Figure 3. 21 shows the abnormal points in the training data frame marked with color coding based on their cluster for wine test data on top of the corresponding BKDE heatmaps in a selected dimension. It is visible that the points that have a considerable distance to the center of clusters as well as the outliers in these dimensions are selected by the algorithm.

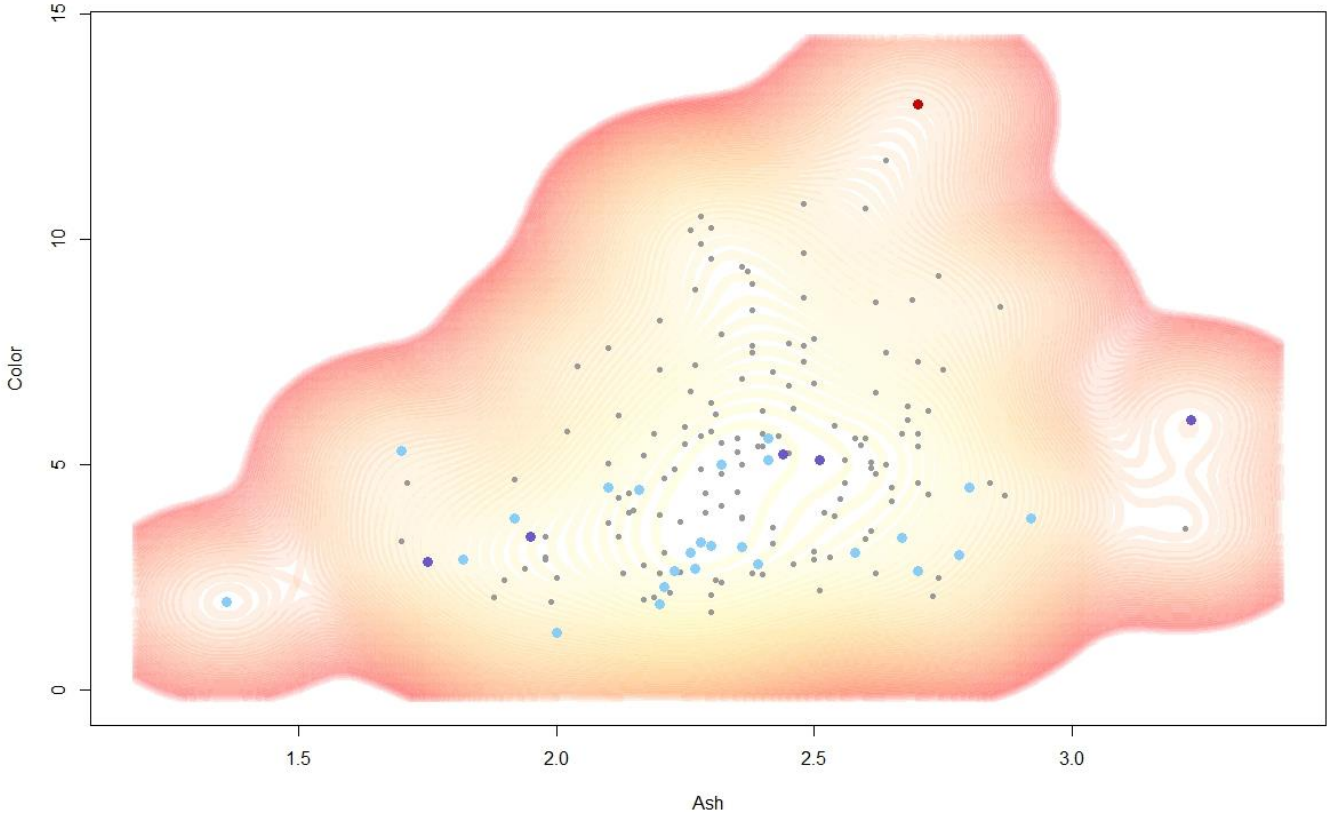

Figure 3. 21 - Anomalies as detected by the fault detection method for wine data. Abnormal points are highlighted in the color that signifies their clusters.

A flowchart for the training algorithm is shown in Figure 3. 22 that summarizes the fault detection method. The R script for performing the method is included in Appendix.

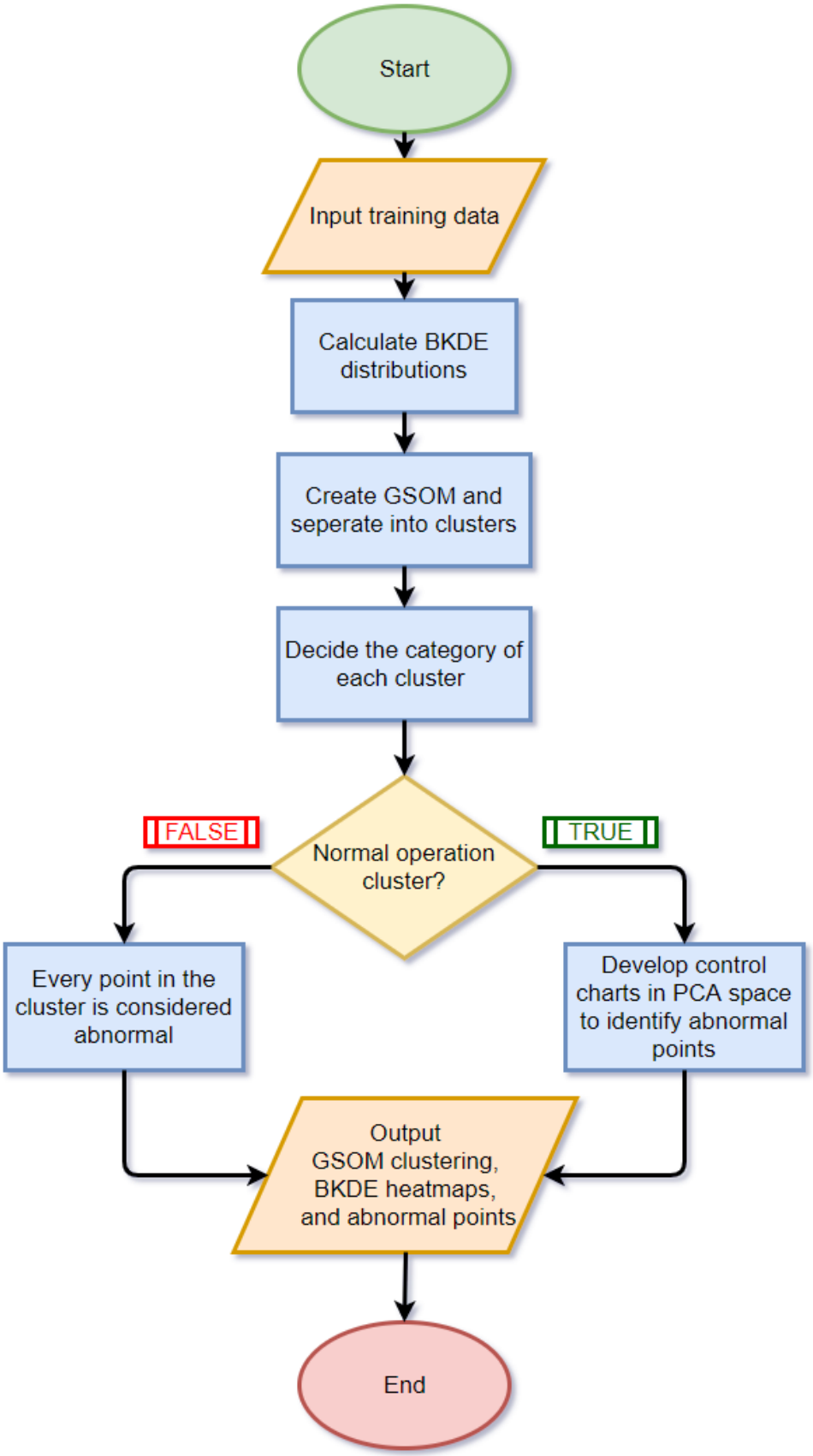

Figure 3. 22 - The fault detection algorithm.
Standard flowchart symbols and notation (Myler 1998, pp. 32–36)

3.2. Monitoring assets using the Fault detection method

The fault detection method, as described in Section 3.1, can be used as a basis for an online monitoring system. The strategy for such monitoring is described in this section. For the purpose of monitoring, it is necessary to define a training set of data, from which the first set of clusters and control charts parameters is derived. The GSOM grid with node positions, as well as clustering information, as well as the BKDE distributions and PCA space data is saved as the result of the training.

In online monitoring based on this method, every new point introduced to the algorithm, is first mapped to the already existing GSOM grid, where the point is assigned to one GSOM node, also known as the Best Matching Unit (BMU). Since the BMU belongs to one of the clusters that are defined in the training, the point subject to monitoring will subsequently be assigned to that cluster.

In the next step, depending on the nature of the cluster that the point is assigned to, different strategies will be set to motion for the point. Same as the training phase, if the point belongs to a failure cluster, the operator is immediately alarmed about the abnormal state of the process. Otherwise, the point is in normal operation cluster, and PCA based control charts should determine the state of the process.

A point that is assigned to a normal cluster should go through the anomaly detection based on PCA control charts. The point is first transformed to the PCA space, and then the Hotelling's and Q-statistic scores are calculated for the point. If these scores surpass the thresholds set by the operator, this point, although lying in a normal operation zone, is a subject of interest.

A high anomaly score for a point in a normal operation zone might be an indication of the process on the onset of or moving towards a failure mode. Due to the value of predictive maintenance and early detection of failures, such point is worth being examine further to find out which cluster they are approaching. A way to achieve that is by calculating the distance of the point to the center of every cluster. Aside from the cluster they belong to, the closest cluster to the point is a good candidate for the state that the process is approaching. If the point is approaching another normal cluster, the possibility of a shift between normal operation modes is most probable. In contrast, a point that is approaching a failure cluster should alarm the operator for the possibility of a failure that is about to happen.

A subject of interest for monitoring systems is their ability to distinguish a novel fault, meaning a failure or an anomaly that has never happened before. In the failure detection method described here, a novel failure is depicted as a new cluster. Therefore, a mechanism is expected to be put in place for adding clusters in case of new failures.

Naturally if a new failure is diagnosed by the operator in any way, a re-training of the method should take place with inclusion of the data related to the new failure and adding one to the number of clusters. However, supposing a monitoring system based on this method in place, there might be no other way to detect a new failure and this system has to offer a strategy for such situations.

A good approach for detecting a novel failure using this method is applied when the state of the process is determined to be in anomaly or failure, meaning the point indicating the online state of the process is in a failure cluster. In this situation, the distance between the point and the center of its cluster can be an indication of the nature of failure. If this distance is too high, it is possible that this point belongs to a state different than the failure mode of its cluster, and it is assigned to this cluster simply due to the lack of a better option. The threshold for this distance is a parameter that should be defined by the operator.

Rate of re-training is another characteristic of monitoring systems. Aside from the necessary re-training in the case of a novel failure, regular re-training with new data measured is recommended. An important point to consider for retraining is that since there are a few parameters that are chosen at random by the GSOM parameters, the clustering results will differ even when repeating the algorithm for the same dataset. This ambiguity of the algorithm can be prevented by using reproducible pseudo-random number generation. This way for a process, GSOM will always be calculated with the same random numbers.

A suggestion for the monitoring strategy is suggested in a flow chart in Figure 3. 23.

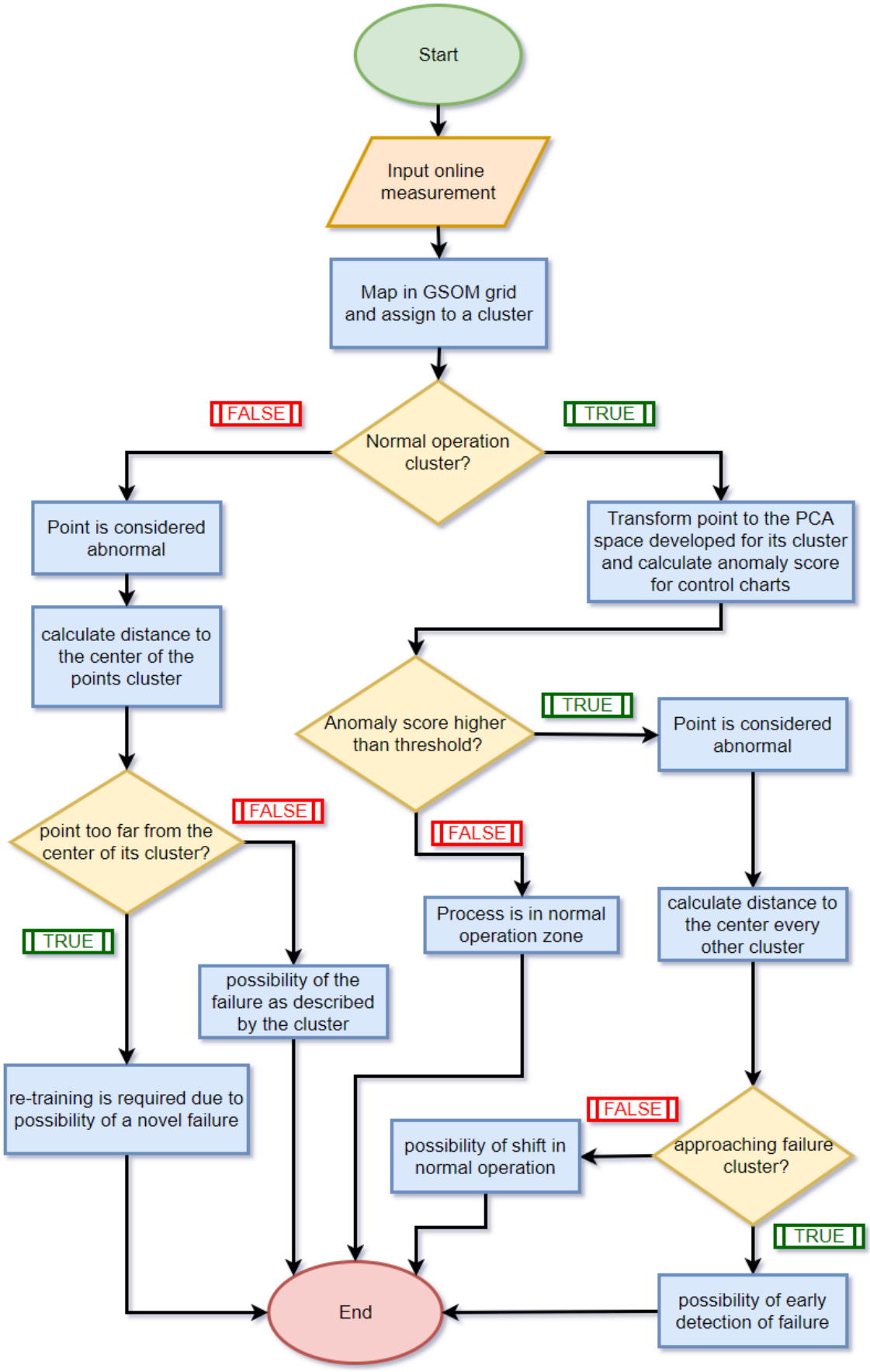

Figure 3. 23 - Monitoring algorithm based on the fault detection method

3.3. Evaluation of the Fault detection method

The criteria for evaluation of diagnostic systems, as developed by Venkatasubramanian et al. (2003b) and described in Section 2.2.3 is used to evaluate the fault detection method. This evaluation is presented from ten aspects. According to Venkatasubramanian et al. these characteristics usually will not be met by any single diagnostic method. However, benchmarking methods by a standard set of criteria will provide useful information in terms of reliability of solution, generality and efficiency in computation in comparison with other methods.

1. Quick detection and diagnosis: Monitoring assets using the fault detection method presented in this study is extremely quick in computation. The calculations necessary for diagnosing a monitored point takes a fracture of a second, which is more than satisfying for the type of assets that the method aims to monitor.

The downside to such sensitive methods is the possibility of detecting noise in data as failure. However, due to incorporation of GSOM in this method noise in one or even a few of the measurements will not have a significant effect on the classification results.

2. Isolability: Being based on self-organizing maps, this method is a classification method in its core. The ability of the method to distinguish is highly dependent on the process characteristics, as well as the information available on the clusters created by the method. However, given proper process parameter definition the method is in theory able to isolate every possible failure in different clusters.

Seeing as the conventional statistical tools used in the industry, such as PCA control charts, provide no way of declaring a cause or a classification for a fault, this method can be beneficial in this aspect.

3. Robustness: The robustness of this method can be explained by studying the building blocks of the algorithm individually. GSOM as the backbone of the method is its best aspect with regards to robustness since it does not include any assumptions on the distribution of the data in its algorithm. BKDE also portrays acceptable robustness due to its non-parametric approach. On the other hand, PCA is prone to corrupted entries due to its dependence on variation of data, making it the weakest link of the method with regards to robustness.

4. Novelty identifiability: A strategy is suggested in Section 3.2 for detection of novel faults. Although it is not verified to produce results as it is described in this study, this strategy results in novelty identifiability for the method, at least in theory. This is only possible due to classification power and robustness of GSOM algorithm, whereas more conventional methods such as BKDE and PCA are not able to distinguish between different classes of abnormal behavior.

5. Classification error estimate: The method as presented in this study does not include calculations regarding classification error estimate. The GSOM classification in the method has no probability model underlying, therefore it cannot be used to calculate error estimate. In fact, BKDE is the only part of the method that could be used to develop estimates due to its probabilistic nature.

6. Adaptability: The adaptability of this method can be assured by introducing a *forgetting factor* in re-trainings, meaning that by systematically omitting some of the oldest data from the training set of data at every iteration of re-training the algorithm will adjust to the changes in process properties. This approach is possible for any machine learning based method, therefore it does not give this method a significant advantage.

7. Explanation facility: This characteristic of diagnostic systems is in complete disagreement with the model-free data-driven approach that is considered for developing this method. In order to provide any explanation on the physical aspect of the process, it is necessary to integrate information about the specific process in the algorithm, making the method specific for the said process. This is against goal that was in mind when developing the method. This method is designed to point the user in the right direction in trying to explain the origin of the fault themselves.

8. Modelling requirement: This method has been developed completely data-driven, with no dependence on physical models or any information about the process it will describe, making the modelling requirements minimal for training and deployments.

9. Storage and computational requirements: Significant progress in computer technology has made storage and computing power easily available. A typical personal computer has the capacity to carry out the calculations of this method, same as most of conventional fault detection and diagnosis methods.

10. Multiple fault identifiability: It is possible that a multiple fault process state lies in a distinctive position in the multidimensional space that is happened to be picked up by GSOM

as a distinctive cluster, different form every individual fault cluster. Aside from this unlikely occurrence, this method classifies every point in one and only one cluster, therefore the method in and of itself is not able to distinguish multiple faults.

This characteristic is not to be considered a major disadvantage of this method, since this problem often proves to be extremely complex due to interactions between different faults.

4. Case Study

In order to illustrate the performance of the proposed method in more detail, it is applied on real process data from a Polyurethane (PU) production plant. The data is provided by Huntsman corporation, a global manufacturer and marketer of differentiated chemicals.

The process that is the subject of this study is Hydrogen chloride compression, which is a side step in the process line of PU. The compression is handled using reciprocal compressors, making them the target asset for fault detection and monitoring. The chemical process of PU production, as well as physical description of reciprocal compressors is presented in the next sections, followed by the results of applying the method on the data from such process.

4.1. Process Description

Polyurethanes process is an important asset within Huntsman Corporation. Huntsman Polyurethanes is a global leader in MDI-based polyurethanes. There are Huntsman production facilities in the US, the Netherlands and China.

Diphenylmethane diisocyanate (MDI), as the principal raw material for production of PU is produced in two plants at the Rozenburg site, located near Rotterdam, the Netherlands. These plants blend and react MDI and polyols with various additives to produce materials for specific applications, mainly PU production. The resulting PU foams are optimized for industries ranging from automotive industries to consumer products such as sports footwear.

4.1.1. Chemical Process

Polyurethane (PU) is a polymer composed of organic units joined by carbamate (urethane) links. It is in the form of extremely lightweight rigid or flexible cellular foam matrix with superior insulating properties. By definition, all of PU materials have at least one urethane group (-NH-CO-O-) in their structure, as shown in Figure 4. 1.

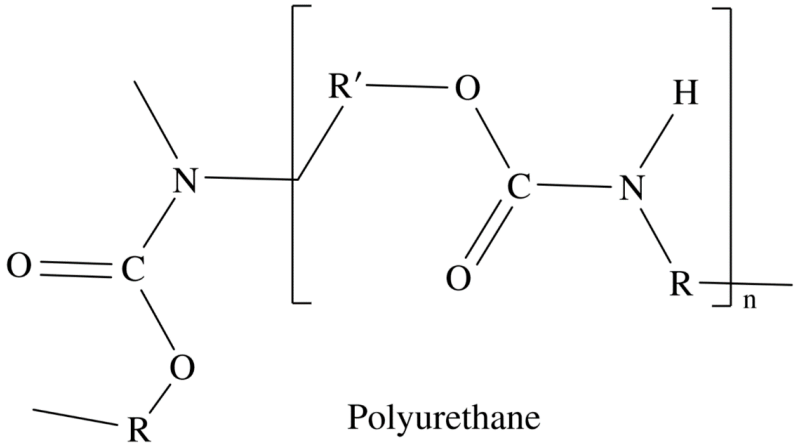

Figure 4. 1 - Polyurethane group (Sonnenschein 2015)

The main areas of use for polyurethanes according to Boustead (2005) are:

1. The furniture and mattress
2. The automotive industry
3. the consumer sectors
4. The construction industry
5. Refrigeration engineering

The biggest areas of use for Rigid PU products are the latter two. These products are widely used for conserving energy in housing and commercial properties, and food supply chain; keeping products at the right temperature in refrigerated vehicles, chiller cabinets and refrigerators.

Seeing as about half the energy used in the life of a building is for heating and cooling, effective insulation is a major priority. Compared to other building materials, polyurethane insulation has the highest thermal resistance (R-values) at a given thickness and lowest thermal conductivity. Use of such material helps meet advanced energy codes, contributes toward green building certifications and provides comfort to the building occupants; Therefore, addressing some of the emerging global megatrends.

The main building blocks of PU materials are polyols and isocyanates. Polyols are defined as the polymer backbones containing nominally two or more hydroxyl groups (OH). The polyols content of the PU is responsible for flexibility and softness of the material, whereas the diisocyanates are responsible for stiffness of resulting PU (Sonnenschein 2015).

The presence of $\mathrm{N}=\mathrm{C}=\mathrm{O}$ triad determines a chemical as *Isocyanate*. The electrophilic carbon and the relatively nucleophilic nitrogen of this group results in a chemical suitable for

addition of active hydrogen molecules, which is a property desired in production of PU (Raspoet, Nguyen 1998).

Isocyanates are characterized by high reactivity and versatility. This has led to the wide variety of applications that polyurethane materials have in chemical industry (Delebecq et al. 2013). There are two primary Isocyanates used in PU production: toluene diisocyanate (TDI), used for producing flexible PU, and *methylenediphenyl diisocyanate (MDI)*, used in rigid PU and insulation foams (Sonnenschein 2015).

MDI, with chemical formula ($OCN.C_6H_4.CH_2.C_6H_4.NCO$) as visualized in Figure 4. 2, is the most common Isocyanate used in production of PU. The largest volume uses for MDI are rigid foams, adhesives, sealants, coatings, elastomers, and flexible polyurethane foam.

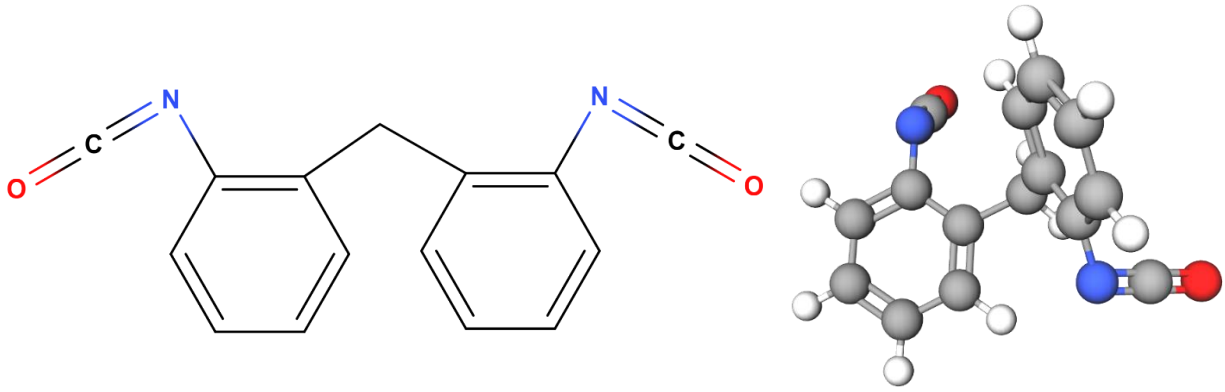

Figure 4. 2 - Molecular structure and 3D ball-and-stick model of MDI molecule

During the industrial process of MDI production as shown in the Figure 4. 3, 1 kg of MDI is produced from 0.8 kg of aniline and 0.9 kg of phosgene producing 0.6 kg of waste HCl (Sonnenschein 2015).

Hydrochloric acid as a byproduct of the MDI process, is also used as a catalyst in the formalin aniline process. The HCl omitted from the reaction, with the carbon monoxide taken off at the top of the reaction column. This composition needs to be treated to the desired condition for re using in the process. The treatment is mainly compression of the gas to the pressure high enough for use as catalyst, and to pipe the excess amount out of the process line.

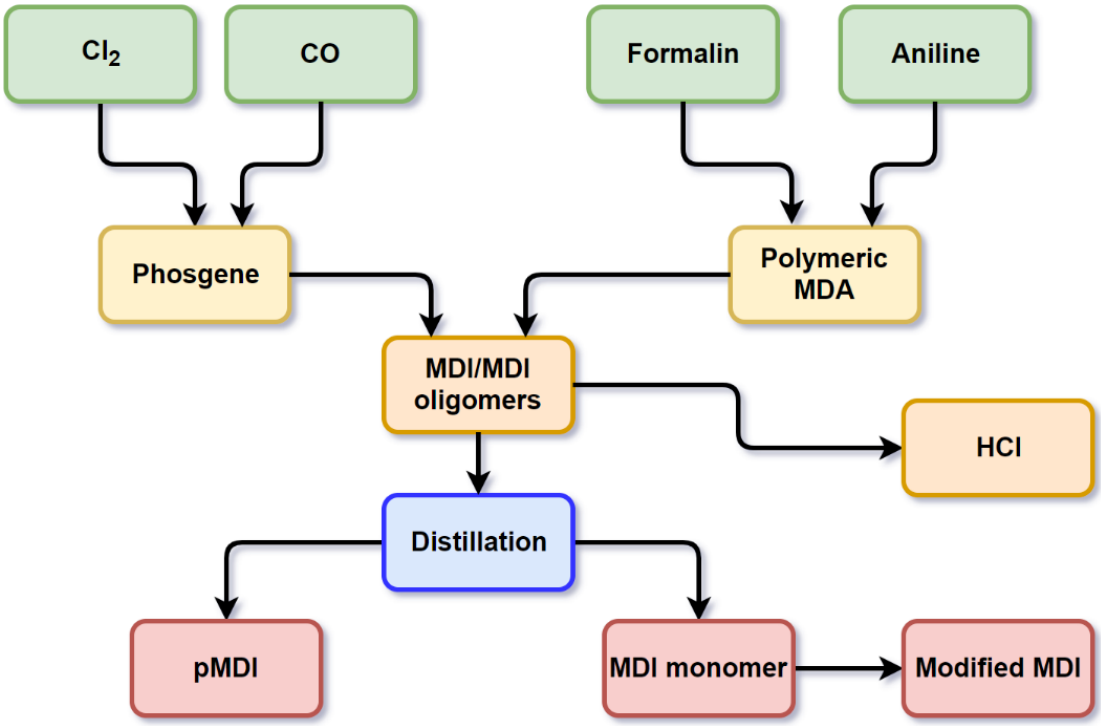

Figure 4. 3 - MDI production process flow (Sonnenschein 2015)

4.1.2. Reciprocating compressors

Compressors are mechanical devices that have the purpose of increasing the differential pressure of a gases. (Bloch, Hoefner 1996).

Industrial compressors are categorized in two main groups: Positive displacement compressors and dynamic compressors. Positive displacement compressors function based on confining the gas in a closed space, and then increase the pressure by reducing the volume of the said space. The compressed gas is then discharged through pipes out of the working space. On the other hand, dynamic compressors function based on increasing the velocity of the gas, and then converting the gained energy from momentum into pressure (Bloch 2006).

Positive displacement compressors have a wide variety of configurations and geometries, among which piston equipped reciprocating compressors and helical screw rotating compressors are by far the most commonly used in the industry.

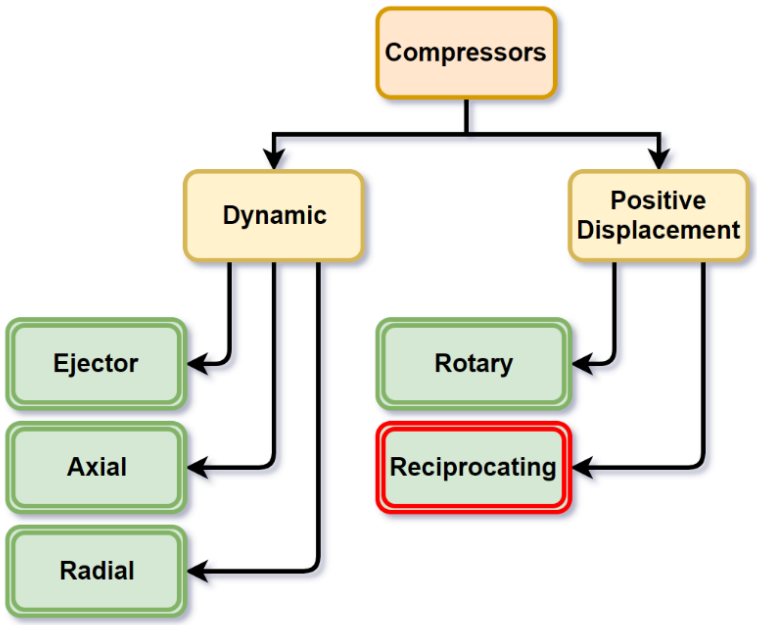

Figure 4. 4 - Classification of compressors (Bloch, Hoefner 1996)

Reciprocating compressors are the most common type of gas compressors. This is due to high pressure ratio that these compressors are able to produce at low mass flow rate, as well as the cost efficiency.

A reciprocating compressor setup comprises a cylinder-piston system with spring loaded inlet and exhaust valve, discharging into a receiver, where the gas with high pressure is stored. A typical setup can be viewed in Figure 4. 5.

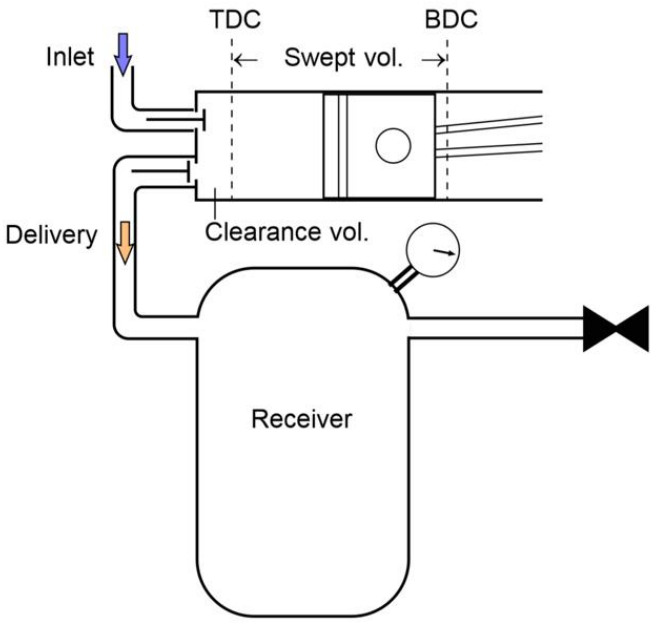

Figure 4. 5 - Reciprocating compressor set-up (Bloch, Hoefner 1996)

The furthest position of the piston crown from the valves and the closest it gets to them, are called Bottom Dead Center (BDC) and Top Dead Center (TDC), respectively. In practice, there is always a clearance between TDC and the cylinder top. This volume of gas is never delivered, and expands during the piston reverse stroke. Therefore, the volume of gas entering the cylinder during induction is always less than the swept volume by the compressor.

Reciprocating compressors follow a four-stage thermodynamic cycle, as the p-V diagram shows in Figure 4. 6.

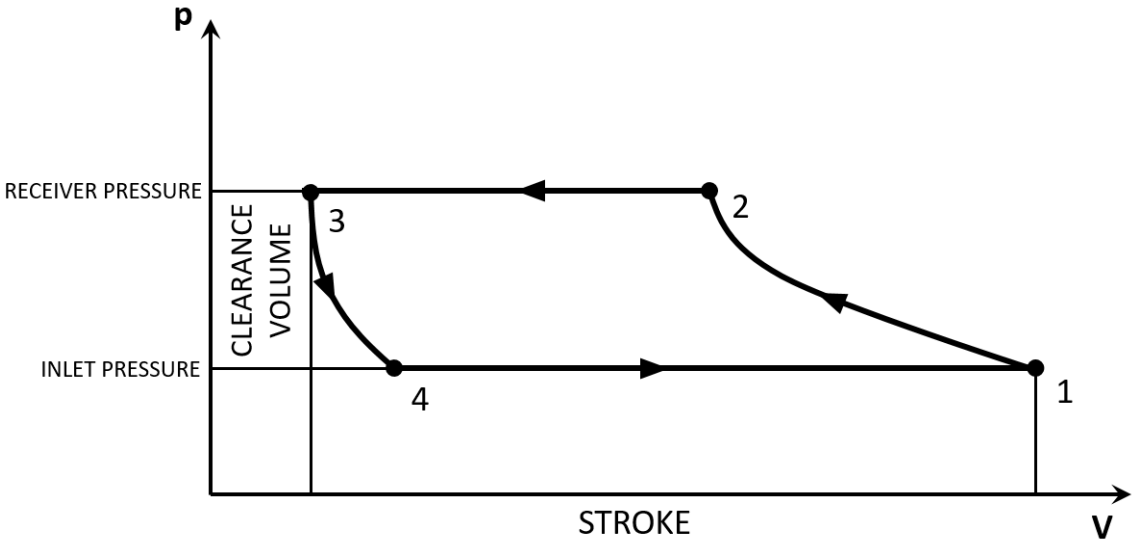

Figure 4. 6 - pV diagram for reciprocating compressor cycle (Bloch 2006)

These stages are as follows:

1- **Compression (1→2):** Starting at point 1 in Figure 4. 6, the cylinder is full of gas and ready to compress, and at this stage both valves are closed. The stroke of piston compresses the gas to the high-pressure state. Both valves stay closed.
2- **Delivery (2→3):** This is the stage where the gas pressure has reached that of the receiver. At this point, the discharge valve opens, letting the gas exit the cylinder while the piston completes its stroke. At the end of this stage, only the gas trapped in the clearance volume is left in the cylinder.
3- **Expansion (3→4):** The third stage marks the completion of the piston stroke. The discharge valve closes, and the piston moving back will increase the volume of the now trapped leftover gas (Expansion). As a result, the pressure decreases to the intake pressure value.
4- **Induction (4→1):** As soon as the pressure inside reaches the intake pressure, the inlet valve opens, letting gas inside the cylinder until the end of pistons reverse stroke (Induction), completing the cycle.

Any setup for a single reciprocal compressor comes with certain limitations, including pressure ratio, discharge temperature, power consumption, and effect of clearance volume (Bloch 2006). Based on these limitations, and in order to reach higher pressure ratios with minimized power consumption, it is common use a *multiple stage compression* setup.

A multiple stage compression setup consists of a number of compressors (usually between two and four) that are serially connected, with a separate cylinder for each stage and intercooling of the gas in between stages. The pistons usually move with the same speed, and are connected to a single rotating apparatus. Each of these can be studied as a single compressor, with the inlet properties equal to the outlet of the intercooler stage before it. A p-V diagram of the combined stages of a two-stage compressor can be seen in Figure 4. 7.

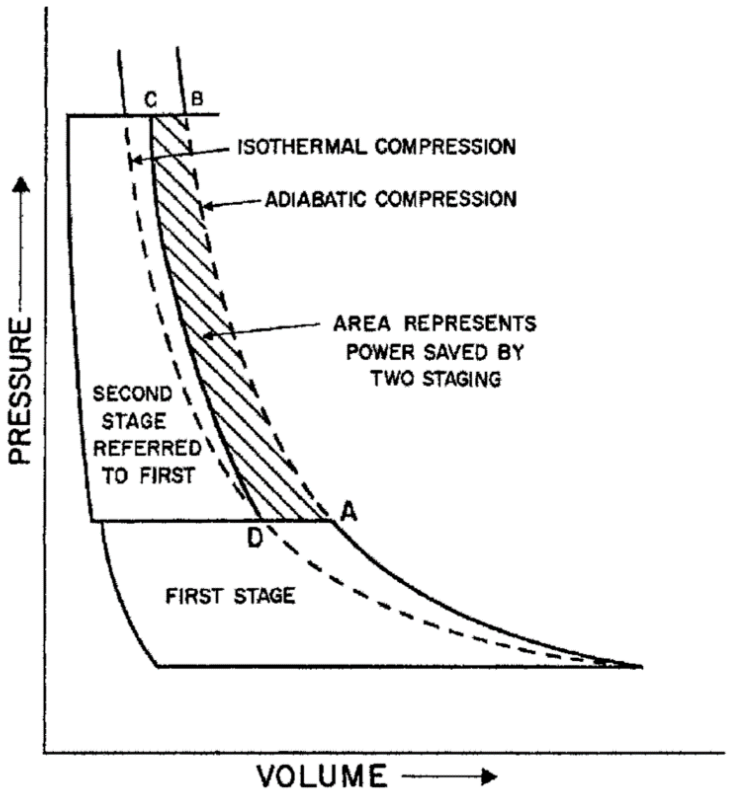

Figure 4. 7 - pV diagram for multistage reciprocating compression (Bloch 2006)

It is desirable to cool the gas after any stage to the same temperature as it was before that stage of compression, therefore minimizing the work required for the next stage.

There is an ideal relation between the pressure ratios of each stage to minimize the power required for the whole setup, in a case where the cooling is exactly to the isothermal point. In this approach, the total compression ratio is distributed to the stages in a way that the pressure ratio of every stage is equal.

4.1.3. Huntsman set-up

The MDI required to for PU process is produced by reacting phosgene with di-amino di-phenyl methane (DADPM). These two building blocks are produced in the following reactions

$$2C_6H_5NH_2 + CH_2O \xrightarrow{HCl} \underbrace{NH_2.C_6H_4.CH_2.C_6H_4.NH_2}_{DADPM} + H_2O \quad (4.1)$$

$$CO + Cl_2 \xrightarrow{C} \underbrace{COCl_2}_{phosgene} + Heat \quad (4.2)$$

DADPM and phosgene are then reacted inside a phosgenation chamber with monochlorobenzene (MCB) as solvent

$$DADPM + 2COCl_2 \xrightarrow{50°-110℃} \underbrace{NCO.C_6H_4.CH_2.C_6H_4.NCO}_{MDI} + 4HCl \quad (4.3)$$

The product of this reaction containing MDI, MCB, and HCl has to be distilled for the MDI to be usable in PU production. Through a series of columns, the different by-products of the process are separated from MDI. HCl with the excess carbon monoxide is taken off at the top of a phosgene absorption column, and then sent to compressors to increase pressure. The set-up contains 5 parallel reciprocating compressors, each with three stages.

4.2. Results of analysis based on the fault detection method

In this section, the results of the method developed in the earlier chapter is studied on the compressor sensor data from the MDI production line from Huntsman plant.

The data frame that is the subject of this study contains 48 measurements throughout the 5-compressor setup, that is recorded with the intervals of one measurement every ten minutes over the course of 15 months.

The measurements include the input and output pressure of the whole setup, as well as input and output temperature and inter-stage temperature values for individual compressors. There are additional values describing the cooling systems of compressors, and finally, measurements in the electric rotary parts of the compressors, which are used to verify the state of the compressors the check whether they are operational. A measurement of the MDI production rate indicates the operation of the whole plant. MDI has positive values, and for very small values of it the plant is considered shut down.

Additional to the measurement information, some supplementary information regarding some periods of time is provided by Huntsman. That includes some instances of unplanned shut downs in compressors, which indicates a failure, as well as some periods of time that due to changes in process parameters, there are fluctuations in the measurements. These information as well as periods leading to a switch between the operation of two compressors are represented as color codes in the first part of the analysis.

4.2.1. Pre-processing

In the beginning, it is necessary for the whole data frame altogether to go through an analysis in order to reveal the multi-regimen nature of it. The MDI rate measurements enables the filtering out of the data for when the plant is down. It is important to filter out the data for these periods of time, because individual compressor sensors are recording measurements while the plant is down, and these values should not be included in the analysis, as they bear no physical relevance.

An important step in pre-processing of the data is scaling. In this step, every variable goes through normalization and standardization process individually, meaning that the arithmetic mean of all of data points in each variable is subtracted from each values observation, and then these values are divided by the value of standard deviation of all the data for each variable. The result is a data frame where every variable has a zero mean and standard deviation equal to one. The result of this process is making every variable dimensionless. Any real process data frame has data with different measurements of different properties, therefore the data is of different units. For example, the data from compressor has temperature measurements, pressure measurements, and other variables that have different units. These units and their ranges are not subjects of interest in data driven analysis, so by scaling the data the only characteristic of the data remaining is the gradients between different instances of measurements.

After filtering out the plant down times, a PCA is solved for the whole data frame to visualize the distribution of the data. The results for the standard deviation of the principal components show a high variance in different dimensions of the data. As seen in Figure 4. 8, the first 10 components have a standard deviation greater than 1, with 6 of them larger than 2. These results prove PCA to be incompetent in discovering sufficient knowledge from this data frame. A biplot of the first two principal components is shown in Figure 4. 8. This figure illustrates the complexity of the data and the multi-regimen nature of it.

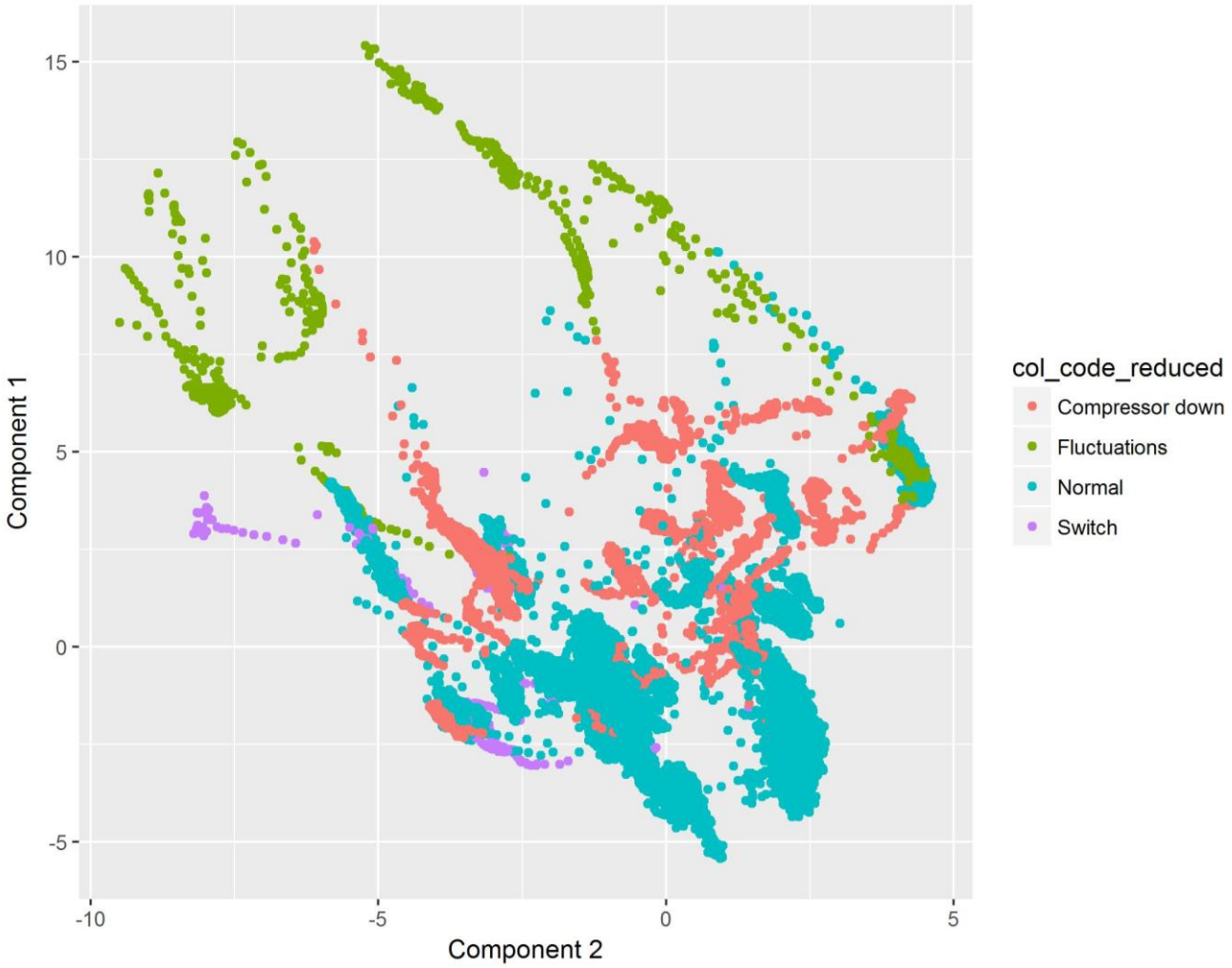

Figure 4. 8 – PCA on Huntsman data for all compressors

In order to elaborate on the nature of the data better, 5 periods of time that are considered normal operation for the plant are selected and combined to form a new data frame. This data frame contains some periods of time that indicate the plant working within the boundaries of normal operation. PCA on this data shows the different zones that a normal operation for this data can have, as seen in Figure 4. 9.

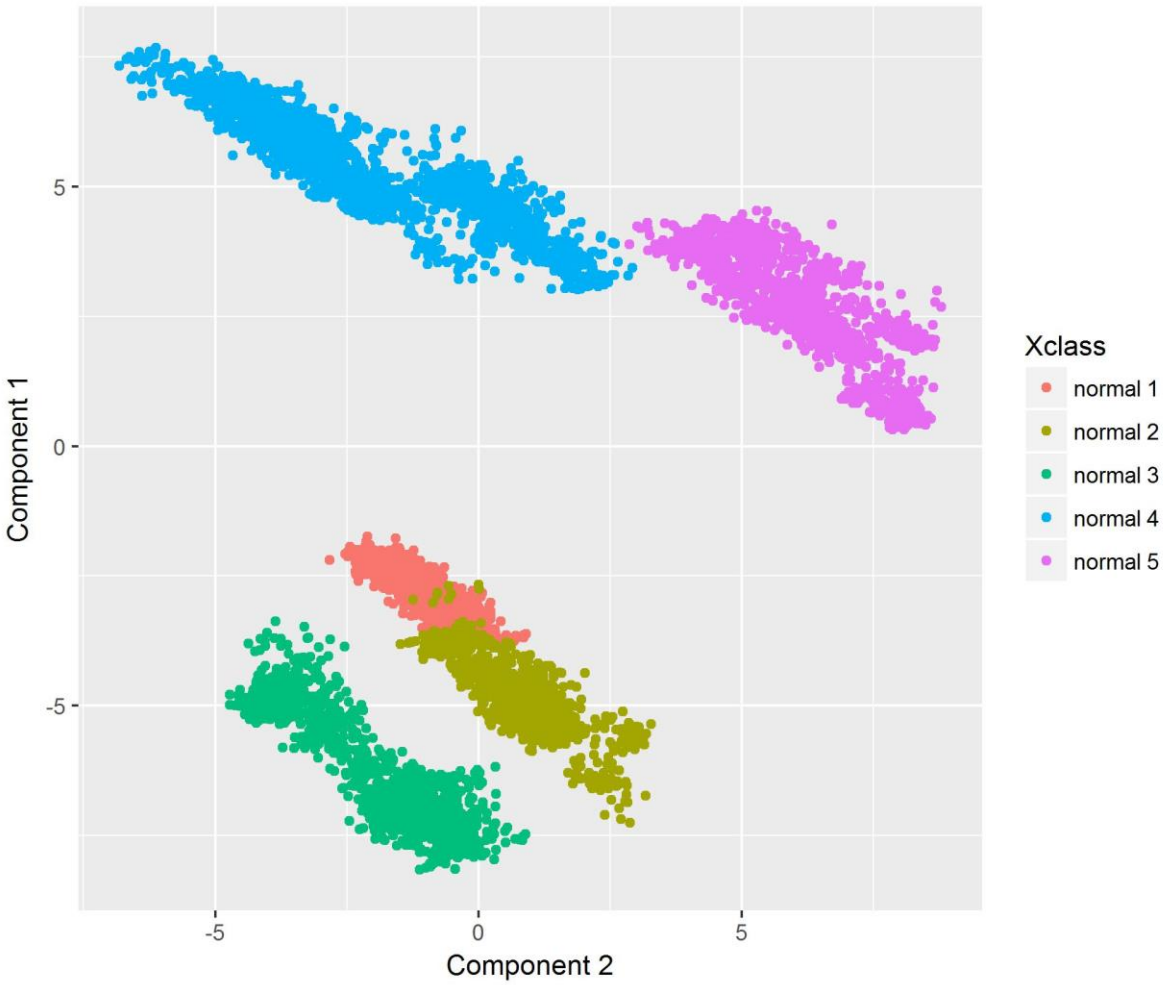

Figure 4. 9 - Different periods of normal operation of Huntsman data in PCA space

Such significant differences in instances of normal operation can be traced to the different combinations of compressors that might be operating in different periods of time. Since there are 5 compressors, of which 4 are usually operational, a switch in the compressors can shift the operation zone drastically. Therefore, it is reasonable to separate the data relating to each compressor and analyze them individually.

The grouping of data for each compressor includes some common measurements, such as input and output pressure and MDI rate, as well as individual measurements such as inter-stage temperatures and fluid flows into the cooling systems. A measurement relating to the electric current of the motor verifies the operation of the compressor. The periods of time where this measurement indicates the compressor to be shut down is filtered out of the data, based on the same reasons mentioned for the filtering of MDI rate low values.

The results of the analysis for one of the compressors is shown in this section. The same analysis is applicable for every compressor in the set-up. Figure 4. 10 shows the correlogram

for the sample compressor. The figure shows high correlation between the temperature measurements, denoted by T in their tag, and less correlation between the rest of the measurements. This is explained by the physical relation between the inter-stage temperatures of the flow, and the natural correlation between these values.

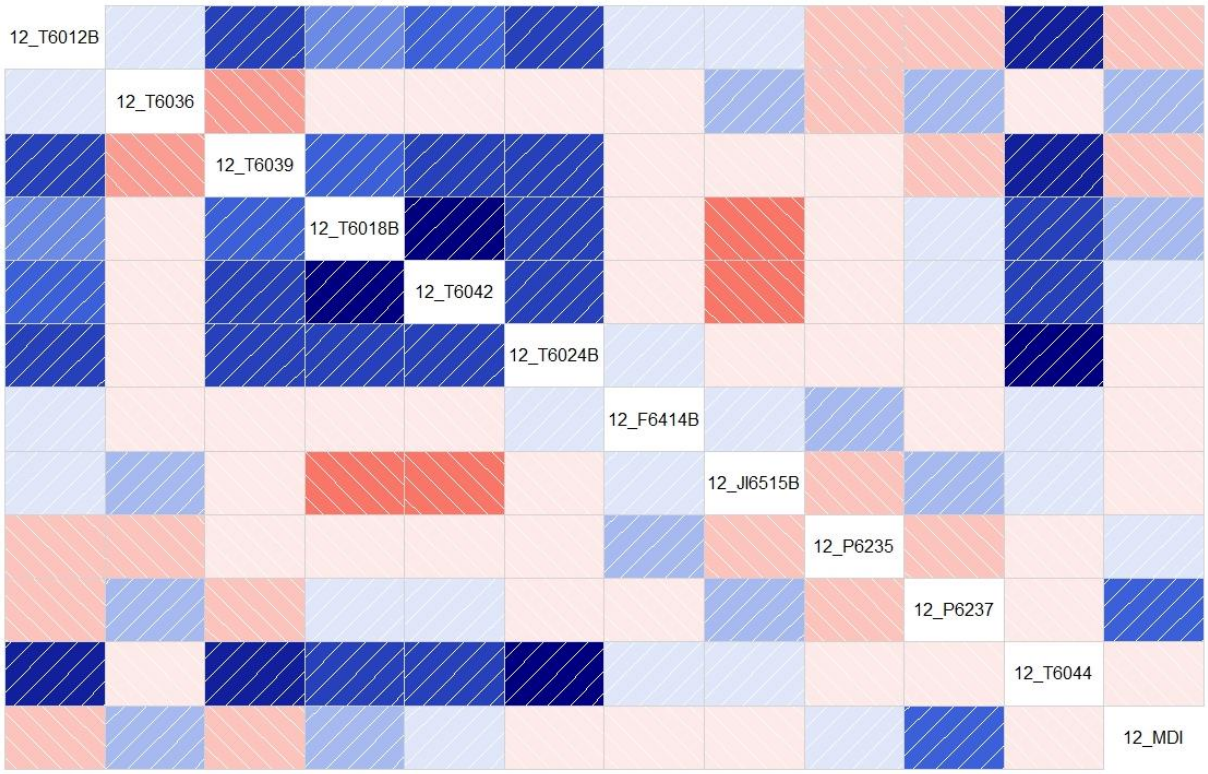

Figure 4. 10 - Correlogram for Huntsman single compressor data. Color blue denoted positive correlation and red denotes negative correlation. The intensity of the colors is an indication of the value, with higher values of correlation having more intense colors.

4.2.2. Kernel Density Estimate

Following the algorithm described in Section 3.1, the first step in analyzing the data is calculation of the BKDE distributions. The values for probability distribution of points in every combination of measurements are stored for later comparison. Plots such as Figure 4. 1 can be produced to illustrate the BKDE distributions. The contours are drawn based on the calculation of the $\hat{f}$ within a 256×256 grid that contains all the data points, as described in detail in Section 2.3.2.

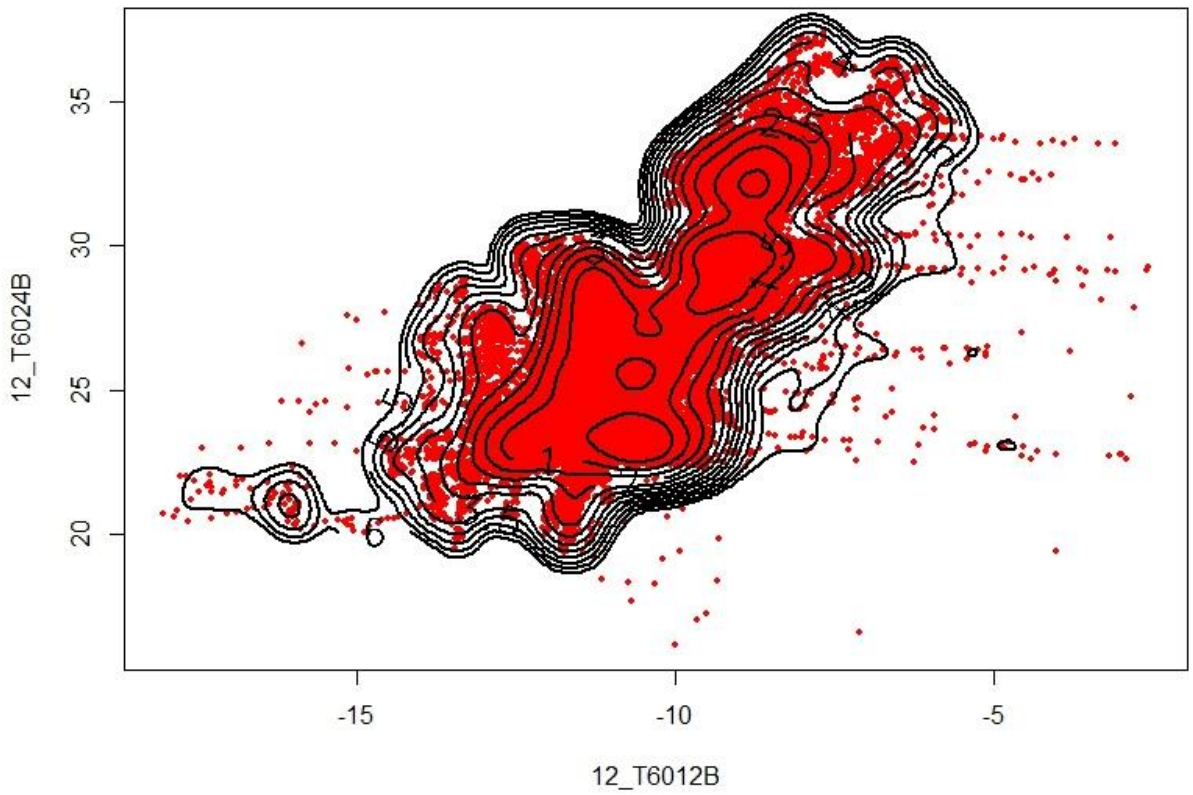

Figure 4. 11 - BKDE contours based on anomaly score for Huntsman single compressor data

The advantage of such estimation, is that there is no assumption taken into consideration for the distribution of data and the method is completely insensitive to the distribution of data. This is contrasting the more usual methods such as PCA, where the accuracy of the method is reliant on the distribution of data. A data with closer to multivariate gaussian distribution will produce more reliable results in PCA. The data points distribution in the PCA space for the first two principal components is shown in Figure 4. 12.

It can be seen in Figure 4. 12 that although most of the data, in fact more than 95% is populated around the center in the PCA space, there are extreme outliers that have high distances to the rest of the points. Additionally, it can be observed that the first two principal components only explain about 53% of the variation in the original data, which is not sufficient for in-depth analysis.

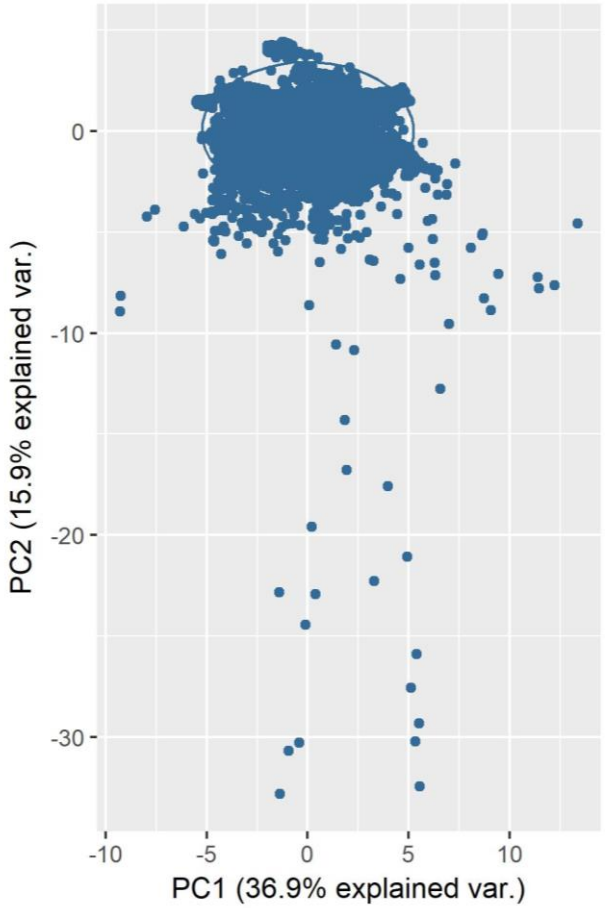

Figure 4. 12 - Huntsman single compressor data in PCA space

4.2.3. Growing self-organizing map and clustering

The next step of the method is mapping the points in GSOM, following Section 3.2.3. The spread factor is set at 0.9, which results in 160 nodes for this data frame. The training plot for this mapping is shown in Figure 4. 13.

The figure shows that the growing phase, where new nodes are generated, continues for 11 iterations of the algorithm. The rest of the process up to 50 iterations is the smoothing phase, where no new nodes are generated, and the algorithm is calculated the same as the original SOM.

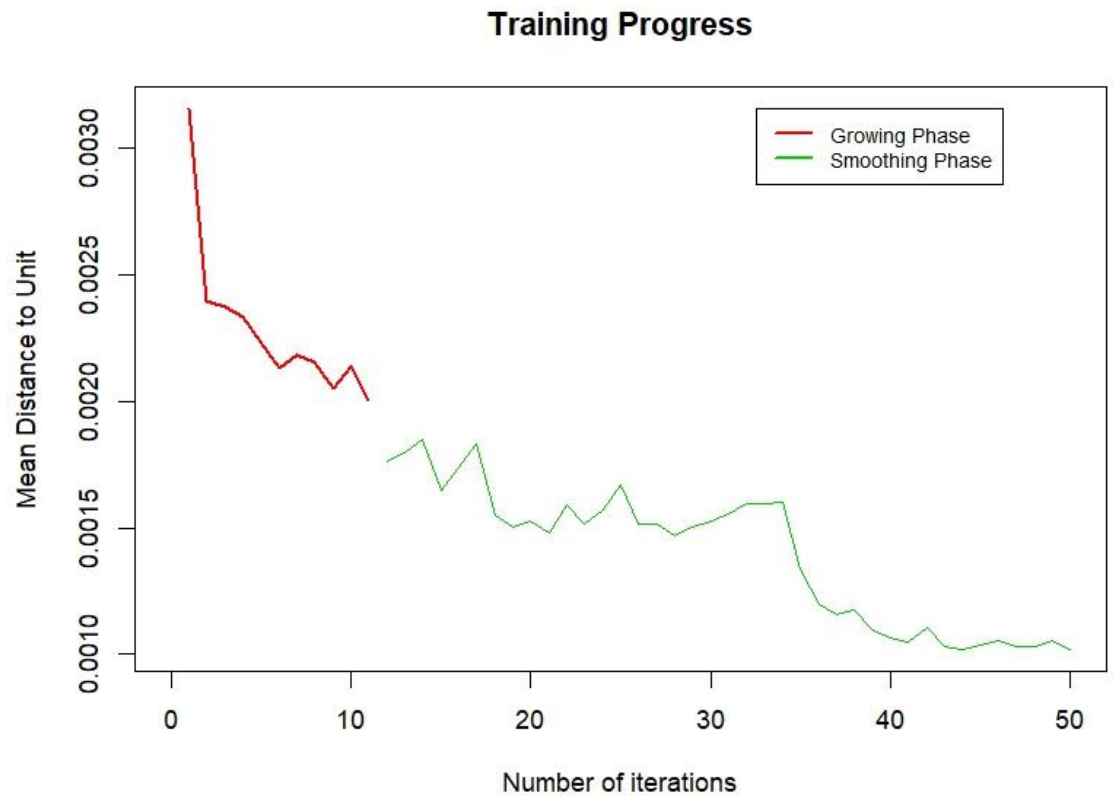

Figure 4. 13 - GSOM training progress for Huntsman single compressor data

A parallel mapping using SOM is calculated with the number of the nodes set to be equal to the result of the GSOM. Since the GSOM resulted in 160 nodes, a grid of 16×10 is defined for the SOM analysis. The training plot for the SOM is shown in Figure 4. 14.

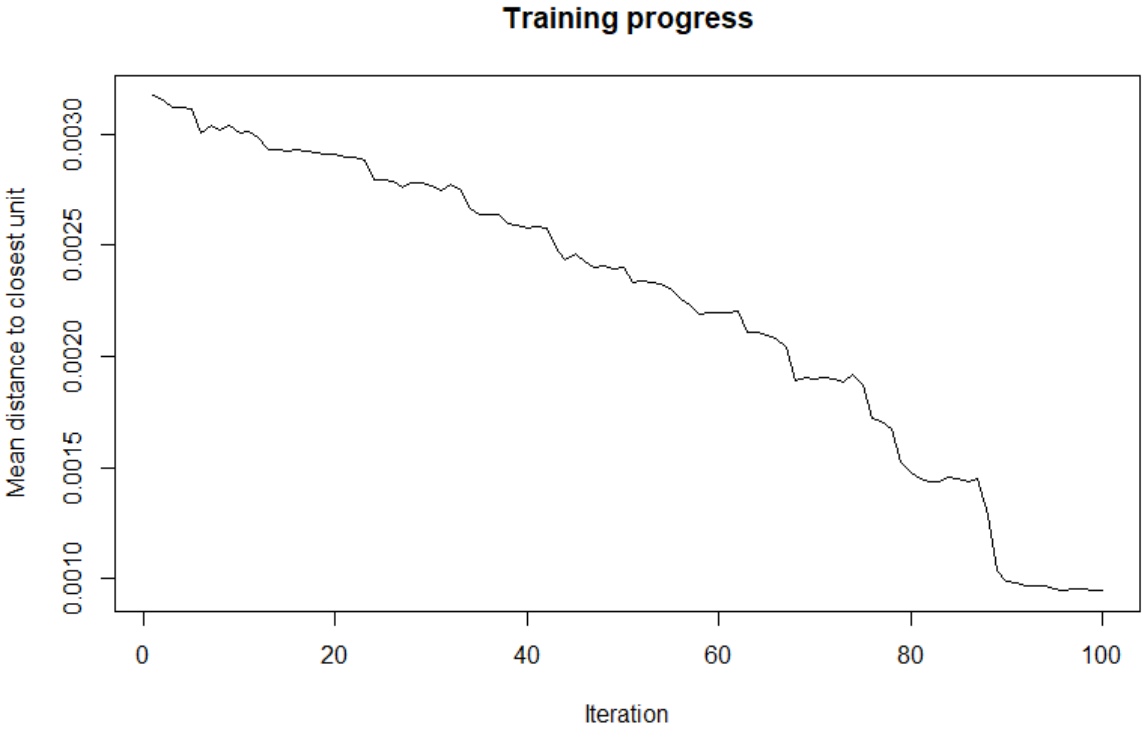

Figure 4. 14 - SOM training progress for Huntsman single compressor data

By observing the two training plots, it is possible to verify that the GSOM algorithm reaches a value of mean distance to unit around 0.001 after less than 50 iterations, which is achieved by the SOM algorithm in around 90 iterations. Therefore, one of the advantages of GSOM over SOM, the speed of convergence, is confirmed.

Another important diagram for the mappings is the count plot, where the arrangements of nodes in an organized map as well as the number of data points allocated to each node is illustrated. This plot for SOM is shown in Figure 4. 15.

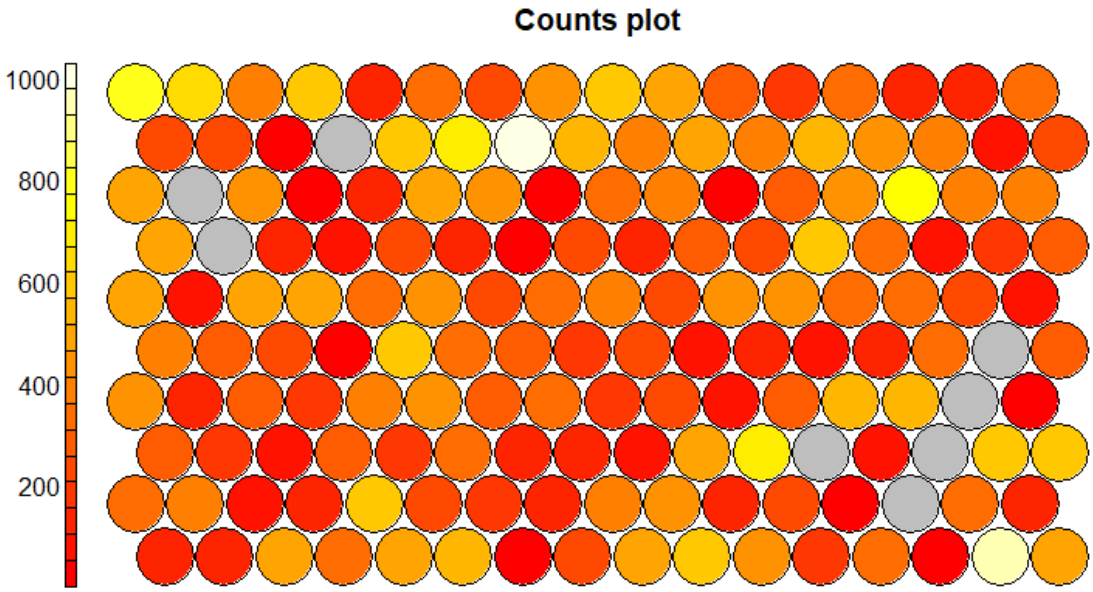

Figure 4. 15 - Mapping of Huntsman single compressor data in SOM

Figure 4. 16 shows the arrangements and the counts of points for the GSOM with same number of nodes. In comparison with SOM, the lack of order in the position of nodes can be noticed.

It is noteworthy that these arrangements are arbitrary, resulting from some of the analysis parameters that are chosen at random in GSOM algorithm; therefore, repeating the analysis on the same data frame, will result in different outputs in terms of node arrangements and even the number of nodes.

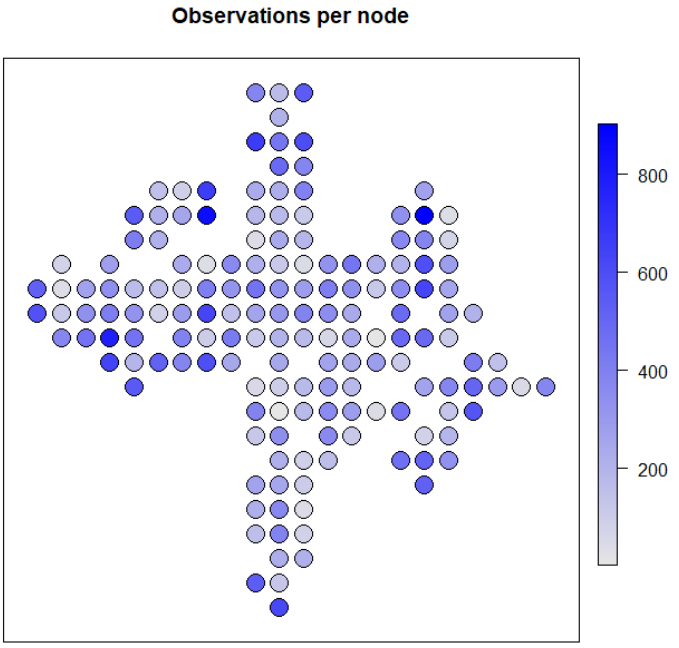

Figure 4. 16 - Mapping of Huntsman single compressor data in GSOM

An example of the repeated test for the same data frame is shown in Figure 4. 17. This mapping results to 167 nodes, and the different arrangement of nodes in comparison with the first mapping is clearly visible. From this observation it can be surmised that the original SOM has better reproducibility compared to GSOM. However, this difference in the arrangement of nodes in the GSOM grid does not cause a major difference in the position of nodes in the multivariate space of input data, therefore the reliability of the method is not challenged by these differences.

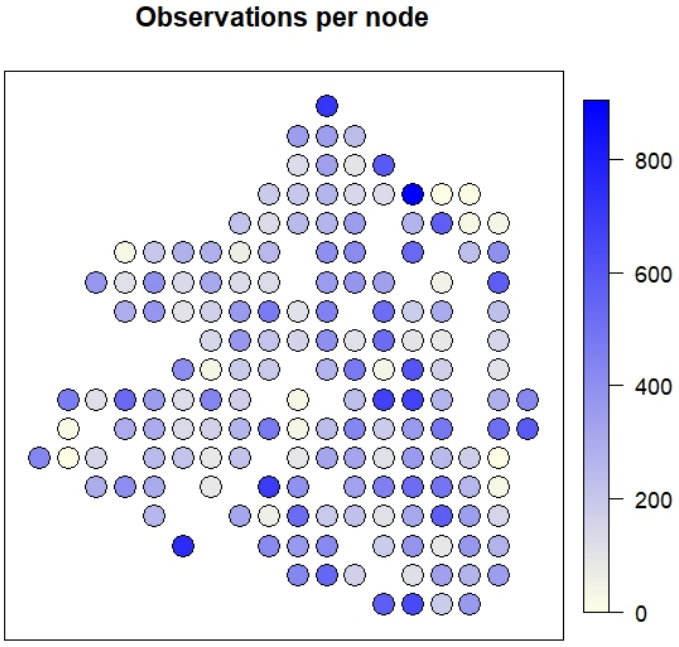

Figure 4. 17 - Mapping results for retraining of Huntsman single compressor data in GSOM

Another comparison between SOM and GSOM can be drawn from the accuracy of the node positioning with regards to the data points density in input space. In order to illustrate in two dimensions, the position of SOM and GSOM nodes are studied in combinations of two measurements in biplots. Figure 4. 18 is an example of these plots.

Although the results of SOM and GSOM as seen in Figure 4. 18 agree for the most part, one major short-coming of the original SOM in comparison to GSOM is visible in this figure. The region between the two big clusters of points is left out by GSOM nodes, as there are no data points in this region. In contrast, due to high density of points above and below this region, a number of nodes are placed in areas with no real data points around by SOM. This inaccuracy, mentioned by many studies about SOM, is verified by observing the positioning of the nodes in Figure 4. 18.

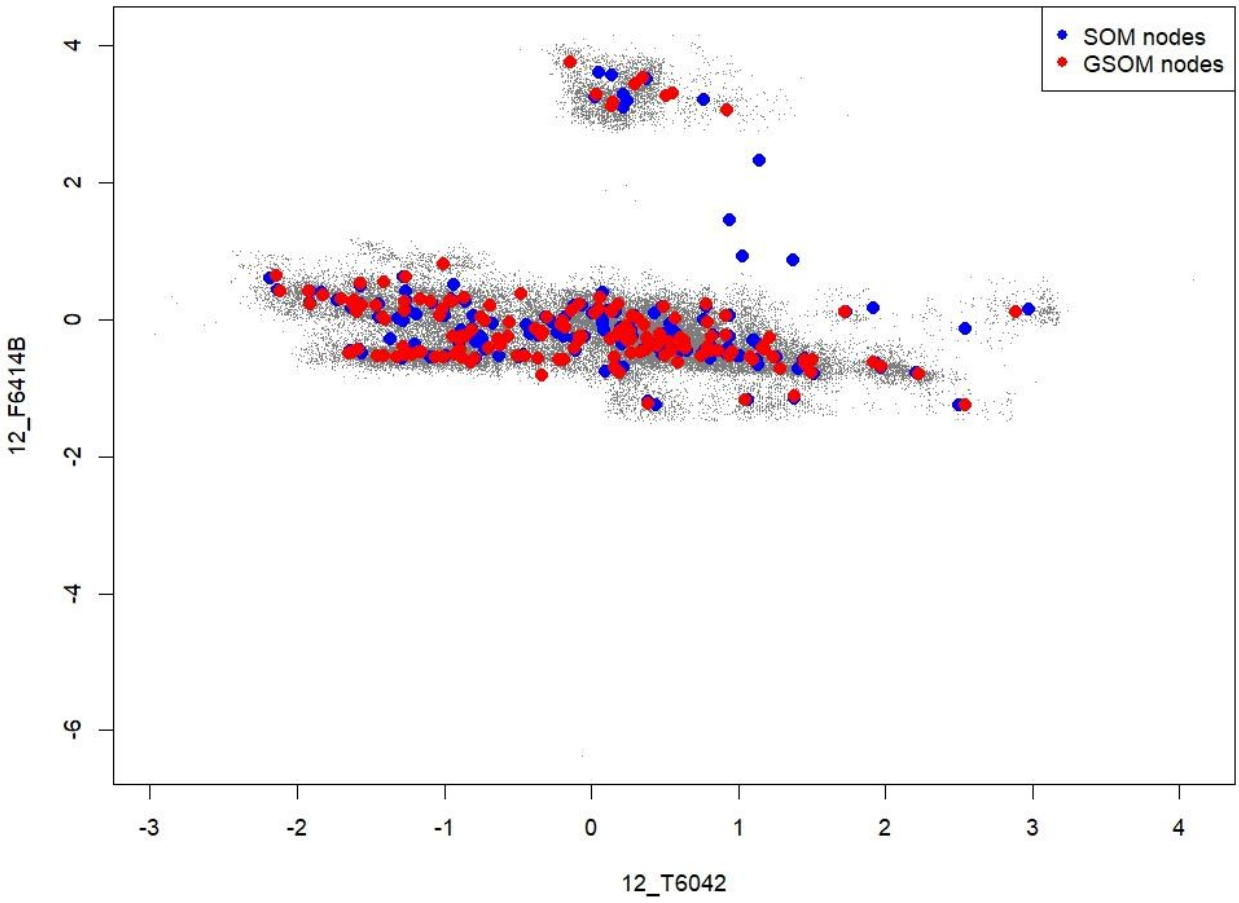

Figure 4. 18 - Comparison in node positions between SOM in GSOM for Huntsman single compressor data

The next step according to the algorithm is clustering the GSOM nodes and subsequently the data points. Although it is recommended to consult the subject matter expert on the number of clusters, a generic method for selecting the number of clusters is here presented. This method, as mentioned in the algorithm is based on K-means clustering. A parameter called *within cluster sum of squares* is calculated for every potential case of clustering based on K-means method. These values are shown in Figure 4. 19.

It can be observed in Figure 4. 19 that the trend of decrease in the value of WCSS shows changes in 3, 6, and 8 for the number of clusters. Naturally, the largest number is selected for clustering the data, since it could possibly be able to reveal more information on the operation zones than the other cases.

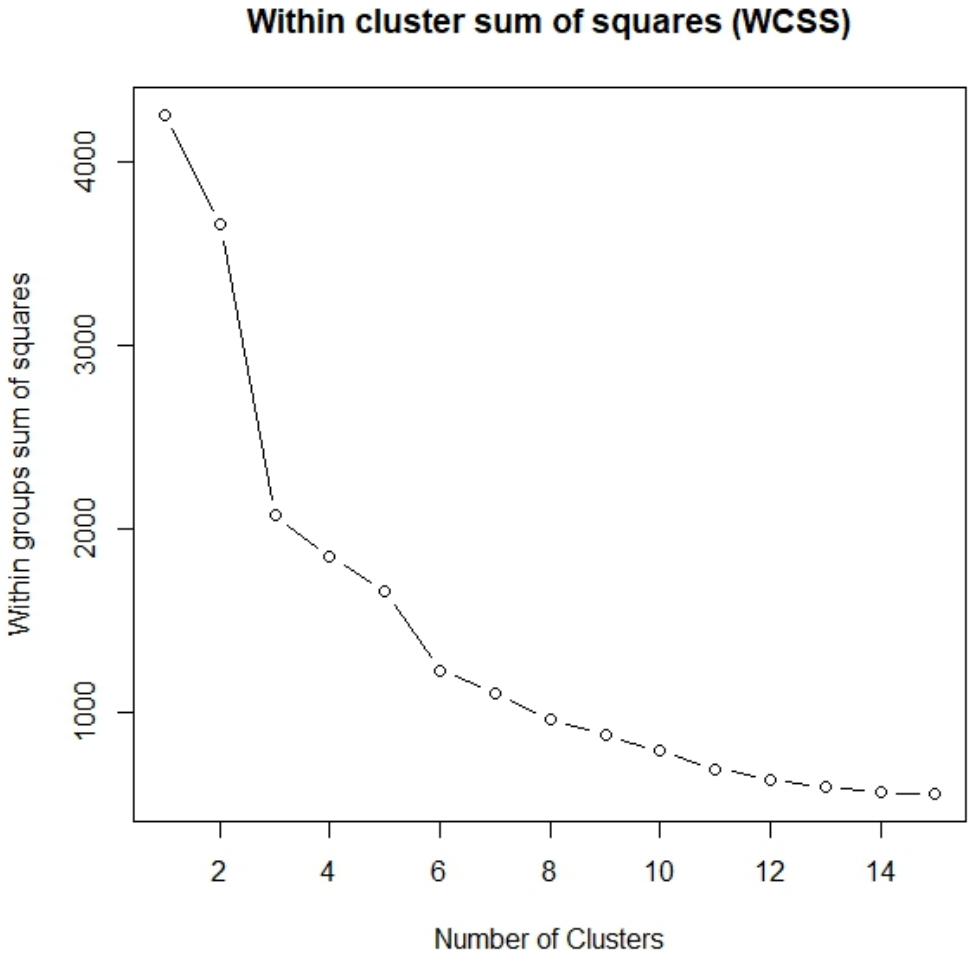

Figure 4. 19 - Estimation on the optimal number of clusters for Huntsman single compressor data

The hierarchical clustering is made as described in the algorithm. The nodes of GSOM are grouped into 8 clusters based on their distance. The data points follow the node that was determined for them as the best matching unit (BMU), and are be allocated to a cluster of their

BMU. In order to visualize the clustering result, the same procedure is done for SOM results. Due to the order in the SOM two-dimensional grid, the clustered nodes will form regions with clear borders. The clustered SOM grid is shown in Figure 4. 20.

It can be seen in Figure 4. 20 that some clusters, such as clusters 4, 7, and 8 only have one node in SOM belonging to them. It should be notes that this is not an indication to the number of actual data points that belong to these clusters. By comparing Figure 4. 20 with Figure 4. 15, where at least 500 points can be seen to belong to the one node that comprises cluster 7.

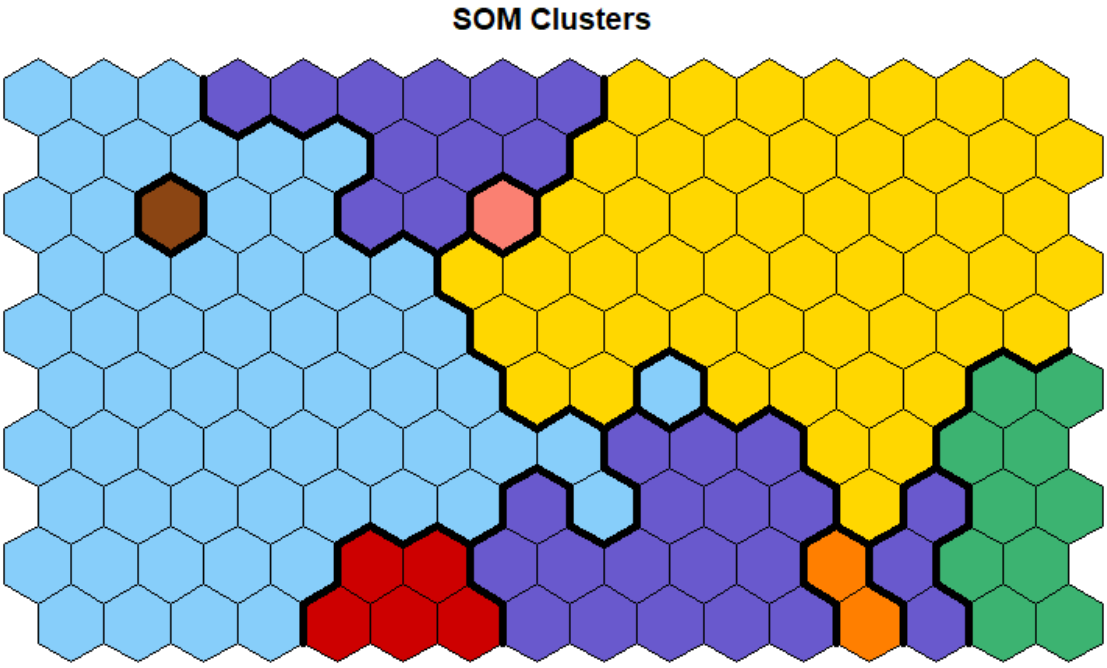

Figure 4. 20 - Clustering of SOM for Huntsman single compressor data

The colors assigned to each cluster is illustrated in Figure 4. 21.

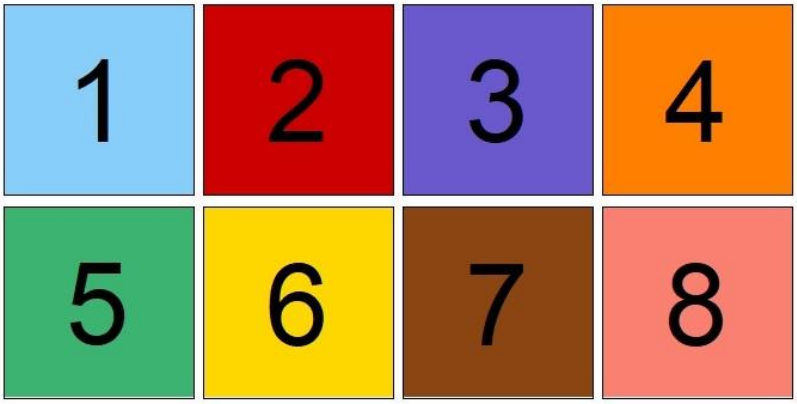

Figure 4. 21 - Colors and cluster numbers relation

The GSOM clustering will also have nodes categorized in 8 clusters, an although a visual representation of the clustering such as Figure 4. 20 is not possible for GSOM, this clustering

will be the basis of the algorithm. This is due to the advantages of GSOM as described earlier. The GSOM clustering is visualized through the color categorization of a dendrogram which is the result of hierarchical clustering of the distance matrix from the GSOM nodes, as shown in Figure 4. 22. In this figure each number belongs to a GSOM node, and the colors determine the cluster that the node belongs to with regards to Figure 4. 21.

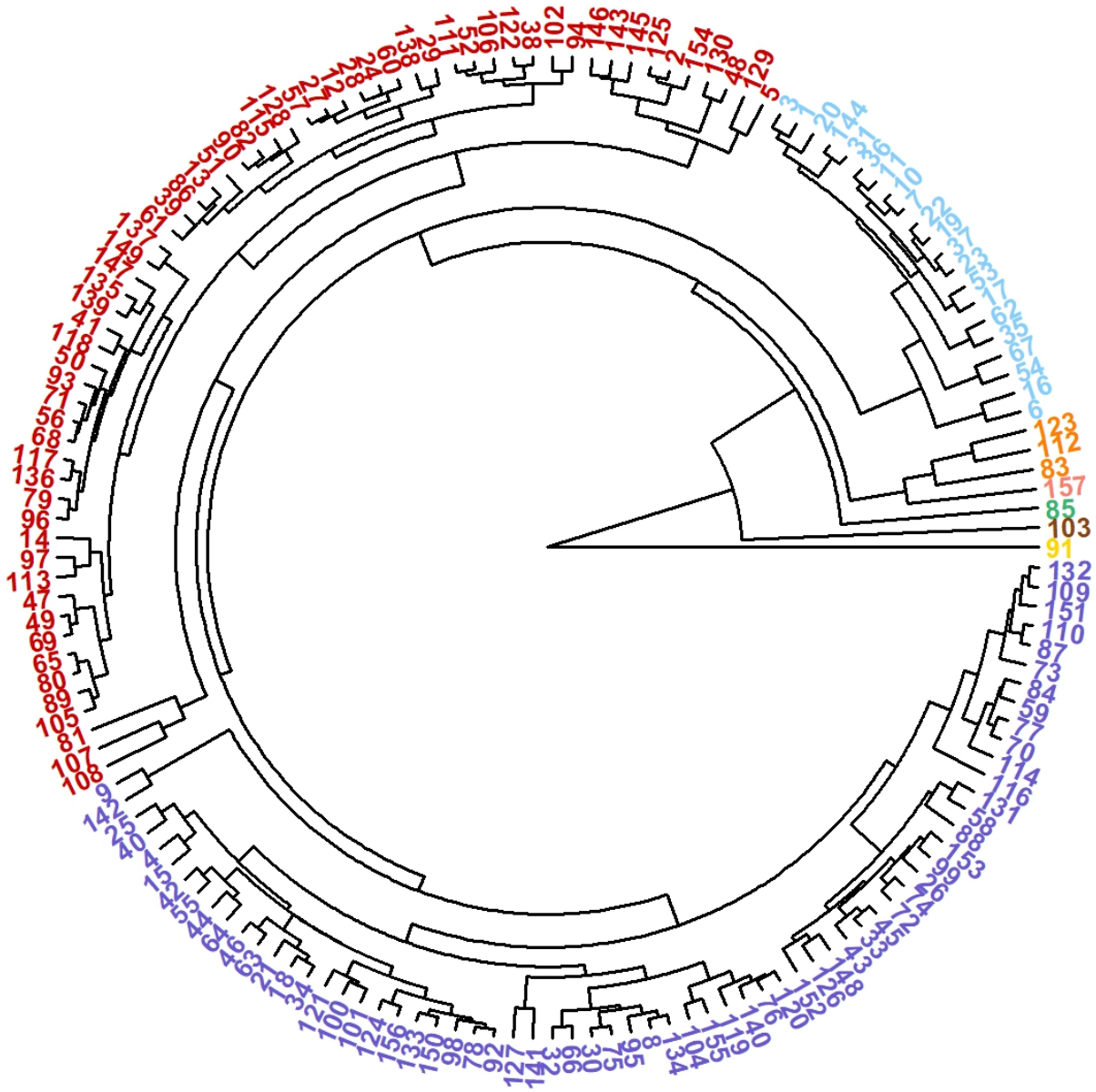

Figure 4. 22 - Hierarchical clustering dendrogram of Huntsman single compressor GSOM

4.2.4. Categorization of the clusters

The next step is to determine the nature of each cluster as either normal operation or failure mode. In order to do so, the correlation binary boards are developed based on BKDE as

described in the Section 3.2.4. For Every cluster, every combination of two measurement possible is studied, where the center of the cluster is assigned an anomaly score based on its probability in the BKDE distribution. The anomaly threshold as described in the algorithm is set at 3, and the results for every cluster of the data is shown in Figure 4. 23.

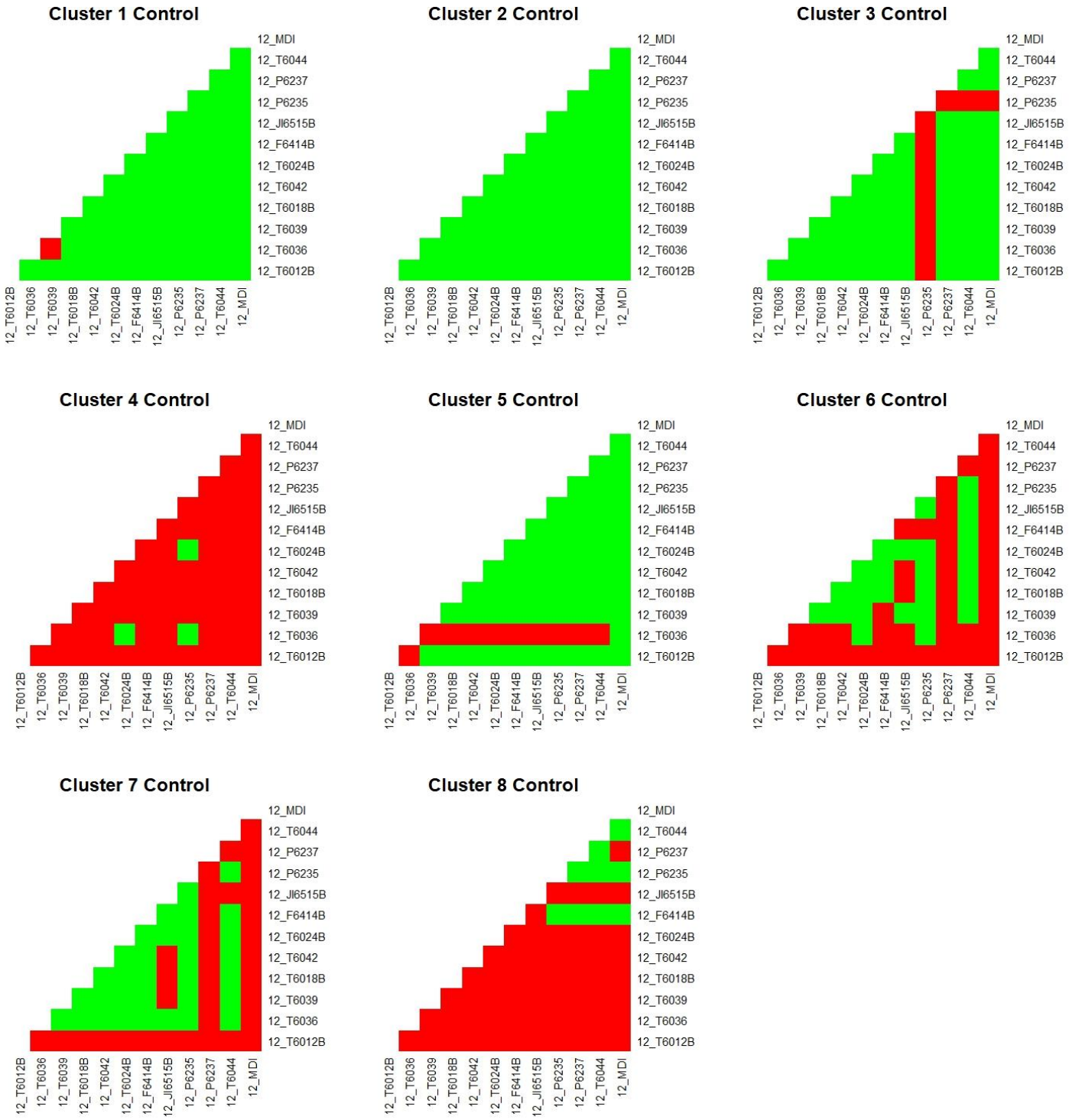

Figure 4. 23 - Cluster control binary boards based on BKDE for Huntsman single compressor data

Although at this point it is again advisable to consult the subject matter expert to decide for each cluster, a more generic approach is described here to categorize the clusters.

Based on observation, it can be deduced that clusters 1 and 2 are normal operation, because there are almost no correlations that have higher anomalies than the threshold. On the contrary, clusters number 4, 6 ,7, and 8 are to be considered failure models, since most of the correlations in these clusters have significantly high values of anomaly score.

The two remaining clusters, 3 and 5, demonstrate one measurement that has high anomaly in correlation with almost every other measurement. The reason for these anomalies can be traced in measurement instruments. Since these measurements are so highly correlated, it is possible that if the correlation of one variable is abnormal with every other measurement, the said variable is not being measured correctly. Normally in a systematic failure, sudden changes in one variable will lead to changes in some of the variables that are closely related to it.

In cluster 5, the variable that has high anomaly in its correlations is an inter-stage temperature measurement. This value is highly correlated with the other inter-stage temperature values of the same compressor, therefore sudden changes in this value should result in changes in the other inter-stage temperature values. In this case, there is no anomaly observed for the other temperature measurements observed, which means that cluster 5 is most probably the result of a measurement error for the said temperature.

On the other hand, cluster 3 shows an anomaly in correlations between every measurement and the input pressure measurement, which is a variable shared by all the compressors in the set-up. This measurement is not an internal property of the compressor; therefore, it is not physically expected to have any specific correlation with the other measurements in this data frame. This can be confirmed by studying the correlations of this measurement with the other variables in Figure 4. 10. Therefore, the measurement error assumption for cluster 3 is considered invalid, as it will be confirmed in the following section.

In any case, clusters 3 and 5 are also considered a failure mode. It should be mentioned again that judging the clusters as normal and failure depends on many different factors, and this method is merely a general guideline. In practice, the data scientist usually does not have to decide on the normal operation zones of a process. However, visualizations with real values of variables can be helpful in decision making process for clusters. Some examples are shown in the following for plots that are helpful for each cluster.

The biplots that will give useful information with regards to nature of the clusters are found by observing all the possible combination of measurements. It is important to note that the separation between every cluster is not visible in every dimension, since some of the clusters will have the same values in some of variables. These plots are specifically chosen out of every possible combination of measurements to highlight the separation of clusters.

For these figures, all the values are descaled for better visualization, meaning that the values are returned to their original values with the original units of measurement. In these plots, cluster 1 and 2 can be seen as good candidates for normal operation zones, due their density compared to the other areas in the plot.

Figure 4. 24 shows a dimension that highlights the distance between clusters 3 and 4 and the rest of the clusters. Both measurements are pressures, so it can be deduced that clusters 3 and 4 signify pressure drops at two different points in the process. As it can be seen data points around nodes of cluster 3 are scattered around space in a physically sensible manner with reasonably variant values, which confirms the cluster to be a physical anomaly and not a measurement error.

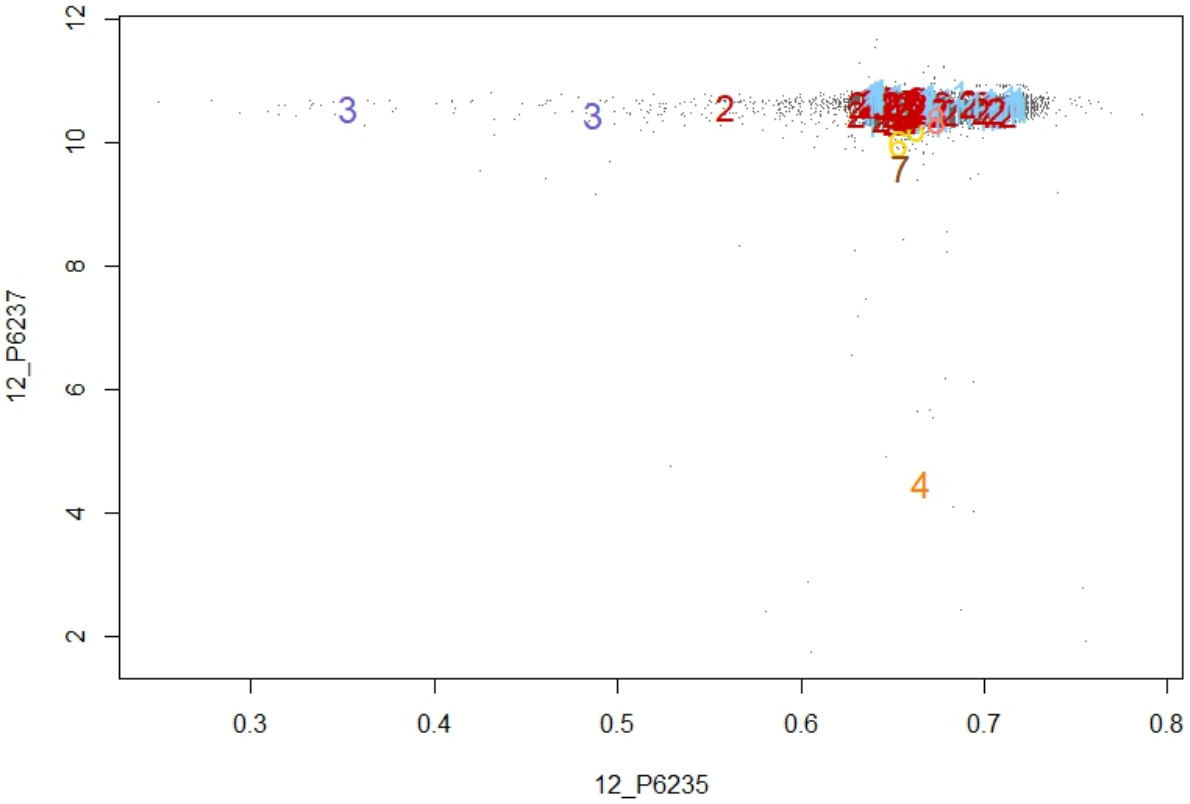

Figure 4. 24 - Position of GSOM nodes and clusters with regards to data points to signify separation of clusters 3 and 4

Figure 4. 25 shows clusters 5 and 8 in their most distant from the rest of data points. Both clusters are related to anomalies in one measurement of an inter-stage temperature. However, there is a major difference between these two clusters that can be studied observing the BKDE correlation charts. Almost every correlation in cluster 8 has high anomaly, making is an extremely abnormal cluster. On the other hand, for cluster 5 the mentioned temperature is the only measurement that shows highly abnormal correlations with almost every other measurement. On this account and the fact that this temperature is for almost every point in cluster 5 measured at the exact same value, which is much higher than every other measurement and physically not sensible, it can be deduced that cluster 5 is in fact caused by a measurement error.

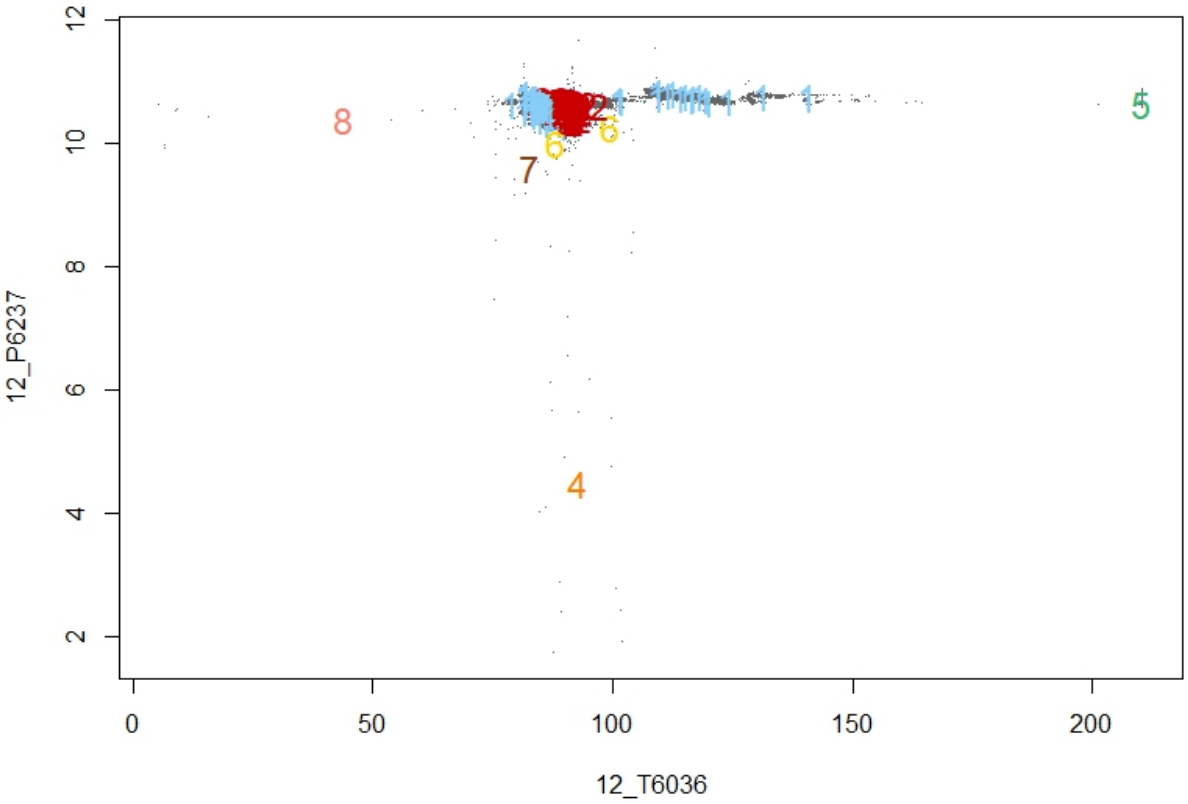

Figure 4. 25 - Position of GSOM nodes and clusters with regards to data points to signify separation of clusters 5 and 8

Anomalies of clusters 6 and 7 are highlighted in Figure 4. 26. Cluster 6 is shown to have higher values in a measurement of inter-stage temperature, and cluster 7 is due to plant running at slightly lower capacity, resulting in lower MDI level. It is also worth noting that clusters 4 and 8 are also observed as outliers in this dimension. This is an expected result, since most of the correlations for these clusters possess high values of anomaly. Nodes of such clusters are

expected to lie at a distance with the normal operation zones in many of the biplots of two chosen variables.

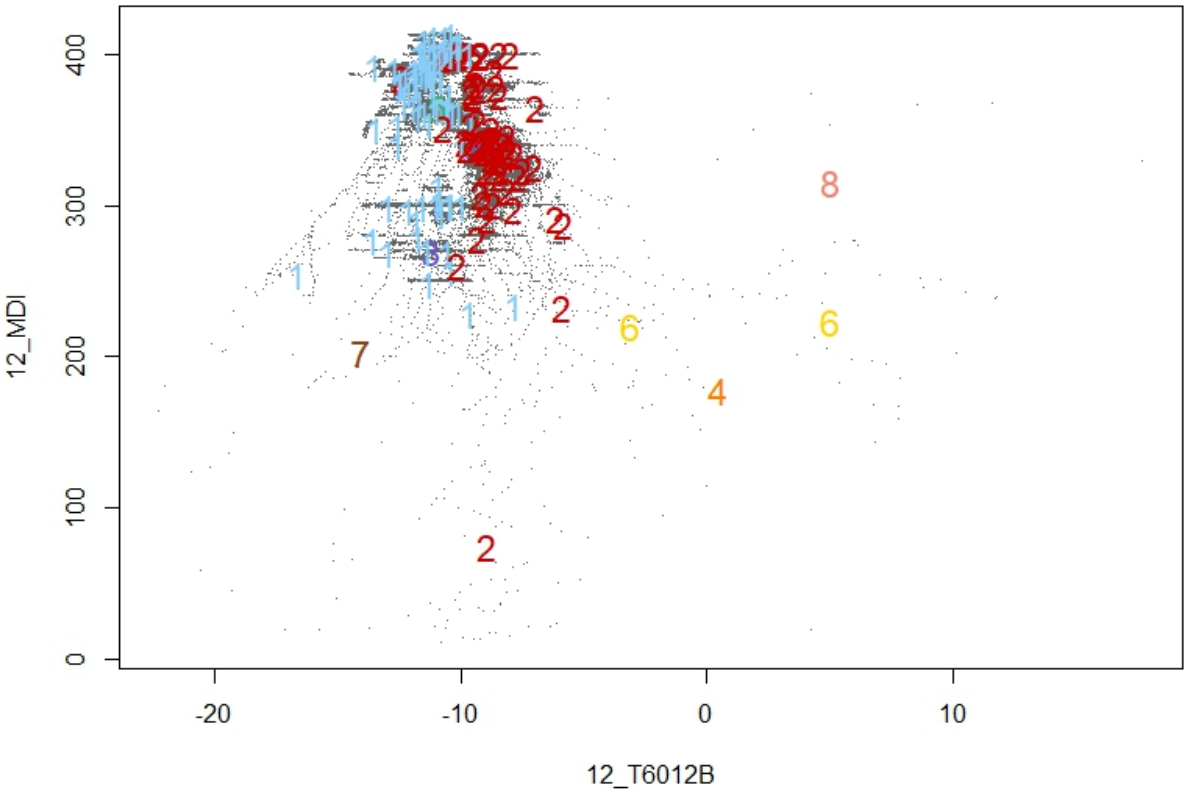

Figure 4. 26 - Position of GSOM nodes and clusters with regards to data points to signify separation of clusters 6 and 7

At this point in the analysis, a monitoring system can already be developed. This system should track the data points in real time and alert the operator whenever the process is operating in a failure mode cluster. However, in order to comprise a monitoring system that has abilities to possibly predict such faults, this method is combined with conventional process data analysis methods based on PCA.

First, it is useful to study the data frame all together in the PCA space and develop the control charts for the process for all the data as one entity with taking the clustering into consideration. Figure shows a biplot of the data for the first two principal components, with the colors signifying the clustering of the points.

It is confirmed in Figure 4. 27 dense area of points around the center is in fact the normal clusters determined by the method, whereas the clusters considered abnormal by the method

are lying mostly far from the center of PCA space. However, some of the abnormal clusters, have blended with normal clusters in this view of the PCA space.

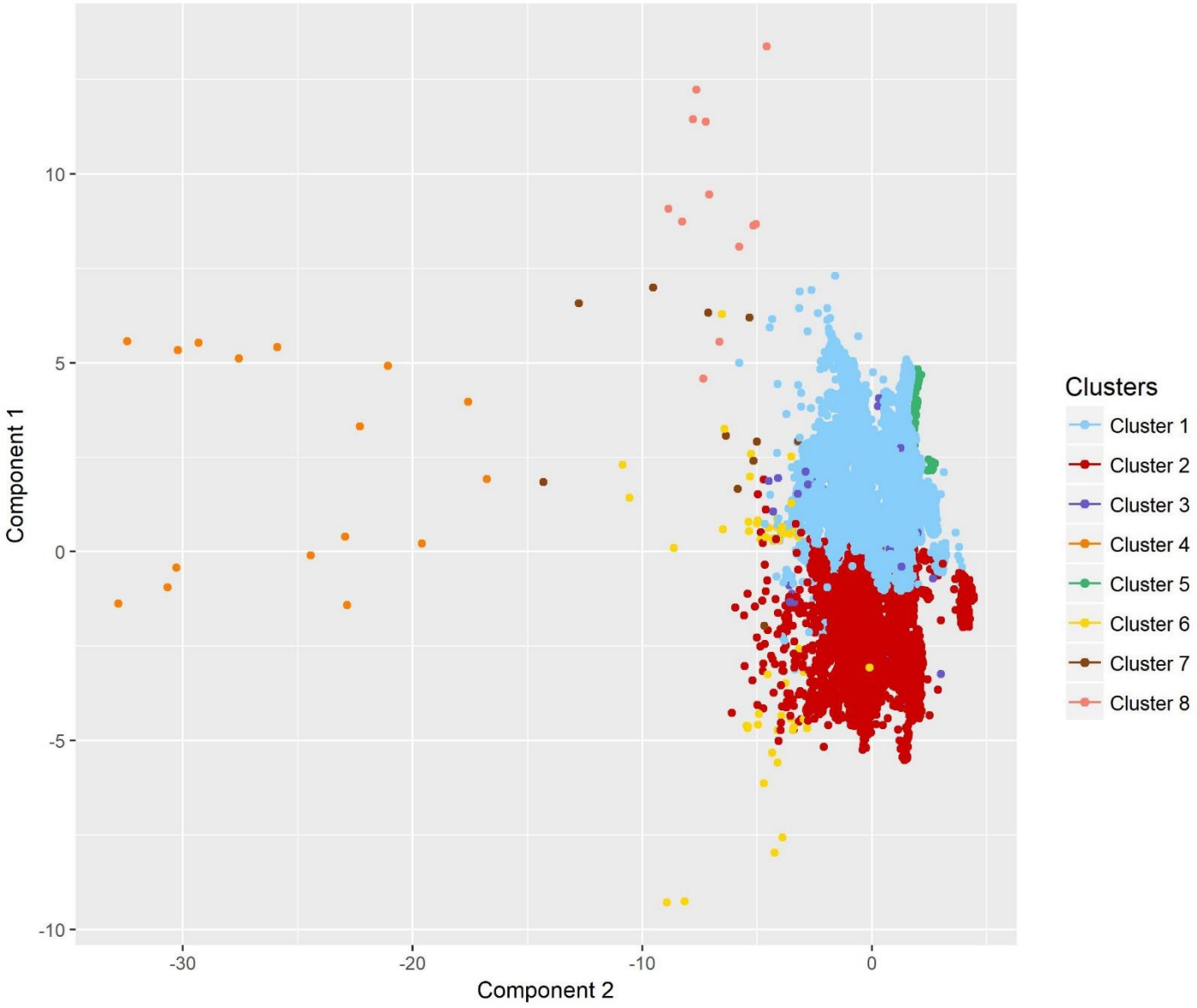

Figure 4. 27 - Clustered Huntsman single compressor data in PCA space with the first two components

Further observation shows that the abnormal clusters that are not visibly separate in the first two principal components view, are possible to detect in other less significant principal components. As an example, Cluster 3 is completely masked by the normal operation clusters when viewed in the first two principal components in Figure 4. 27, but the same cluster lies with a clear separation from normal data when viewed from the fourth principal components

perspective in Figure 4. 28. Same argument can be made for cluster 5, which is revealed when observing the third principal component.

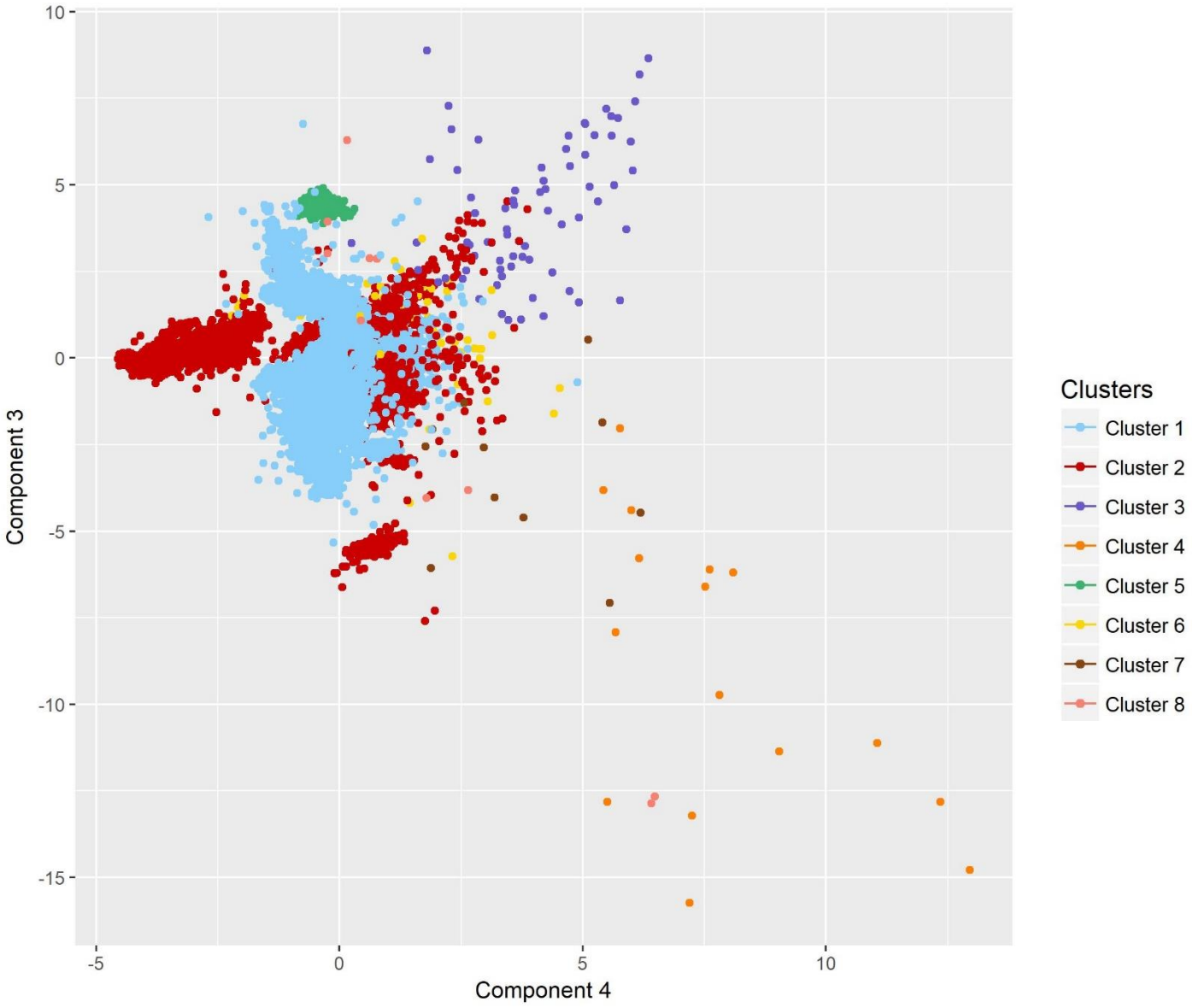

Figure 4. 28 - Clustered Huntsman single compressor data in PCA space with the third and the fourth components

The PCA control charts can also be developed to study the abnormality of clusters when the parameters of the tests are derived from all of data. The number of principal components retained for calculating the parameters of these tests is selected as 4, based on the standard deviation of the components, shown in Figure 4. 29. As divided in this figure, components that have standard deviation values larger than one are retained in for developing the control charts.

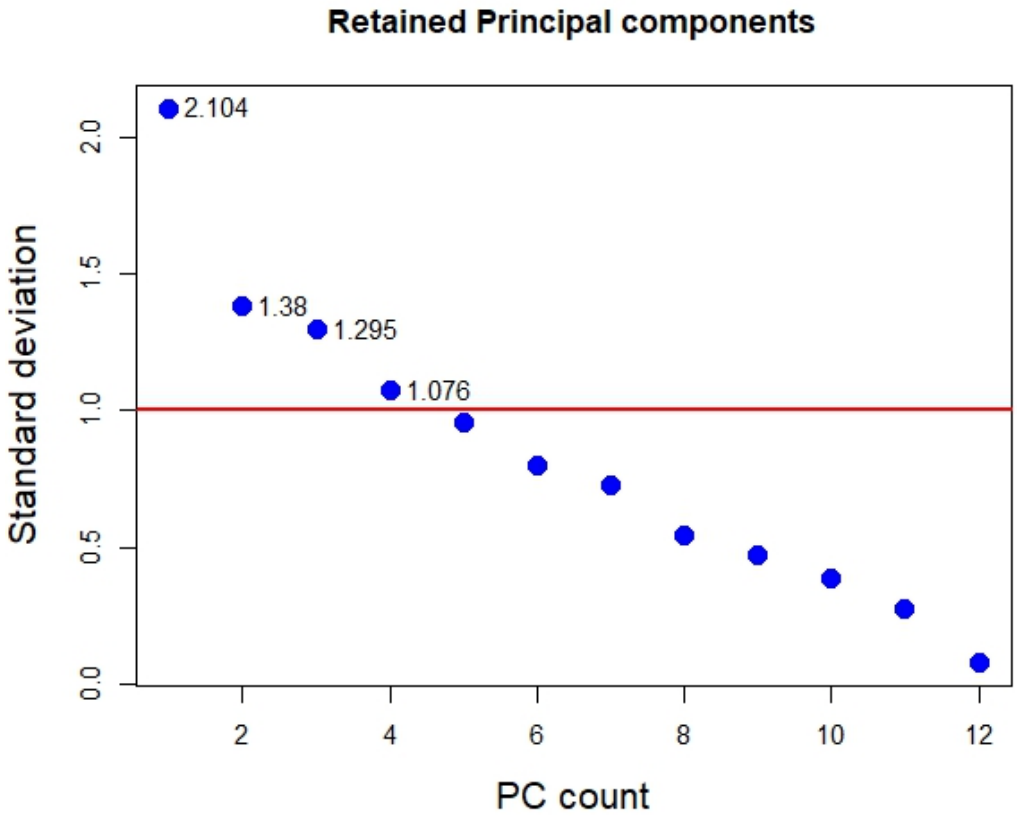

Figure 4. 29 - Decision making on the number of retaining principal components for Huntsman single compressor data

The results of Hotelling's T^2 and Q-statistic tests are shown in Figure 4. 30. It shows the anomaly scores for data points. Here it can be confirmed that the points that belong to clusters 4 and 8 have extremely high anomaly scores in both tests, and points belonging to normal clusters mostly have anomaly scores within the threshold defined by both tests. This figure clearly illustrates the limitation of PCA control charts. Clusters such as 3 and 5 do not score very high in these tests, although it is shown in Figure 4. 24 and Figure 4. 25 that the points belonging to these are extreme outliers.

Additionally, the anomalies determined by PCA control charts are not distinguished based on the state of the process, whereas in GSOM clustering these anomalies are categorized in different clusters. This information is essential for more in-depth analysis in diagnosing the nature, causes, and outcomes of these faults.

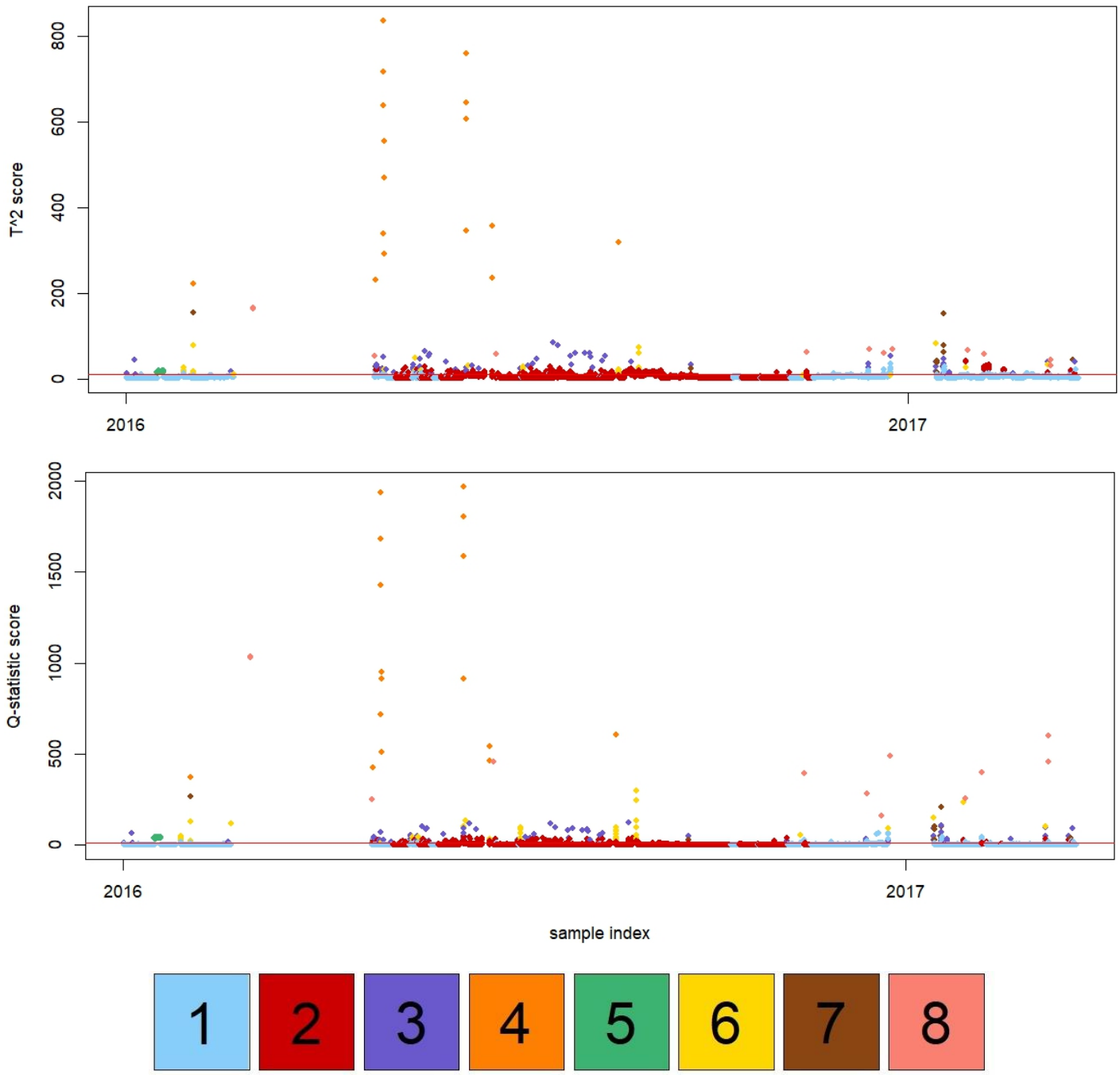

Figure 4. 30 - PCA control charts test results for Huntsman single compressor data. Hotelling's T2 and Q-statistics anomaly scores with different clusters marked with different colors

In order to study the agreement between the two tests, the anomaly scores based on these tests are normalized to a logarithmic scale in a way that the threshold for both tests is transformed to zero, as described in the description of the method in Section 3.2.4. These results are plotted in Figure 4. 31.

It can be seen in Figure 4. 31 that although the two tests have some differences in scoring some points with lower anomaly scores, they match in the spikes, which are the points with high anomaly score.

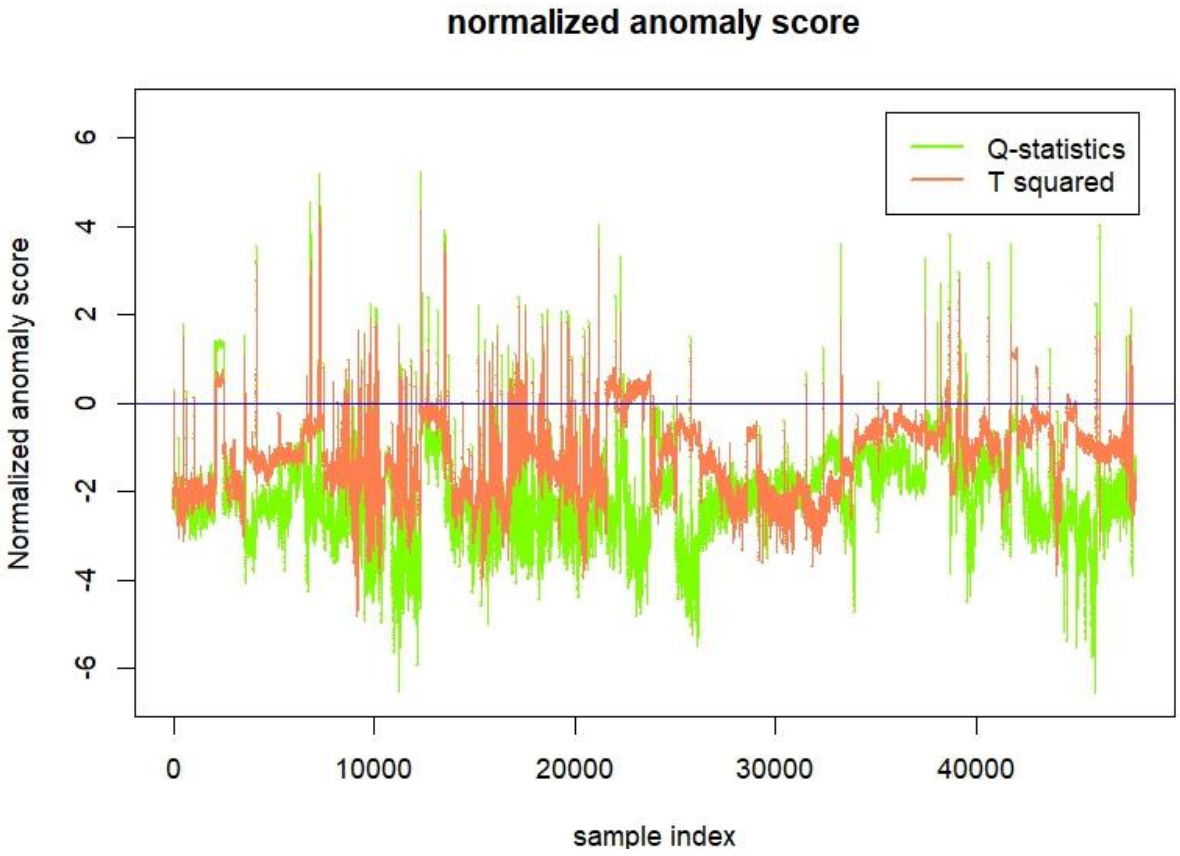

Figure 4. 31 - Normalized PCA control charts anomaly scores for Huntsman single compressor data

4.2.5. Decision making within clusters

The next step according to the method is to develop individual control charts based on PCA for every normal operation cluster. Since The major anomalies are already categorized in the failure clusters, this method will attempt to detect the points inside the normal operation zones that show suboptimal behavior. These points are vital for monitoring purposes, since they are good candidates for the onset of or a move towards a failure.

The Q-statistics control charts for the two normal clusters of the data is shown in Figure 4. 32. Similar plots based on Hotelling's test can also be used, since the two methods were shown to be practically interchangeable.

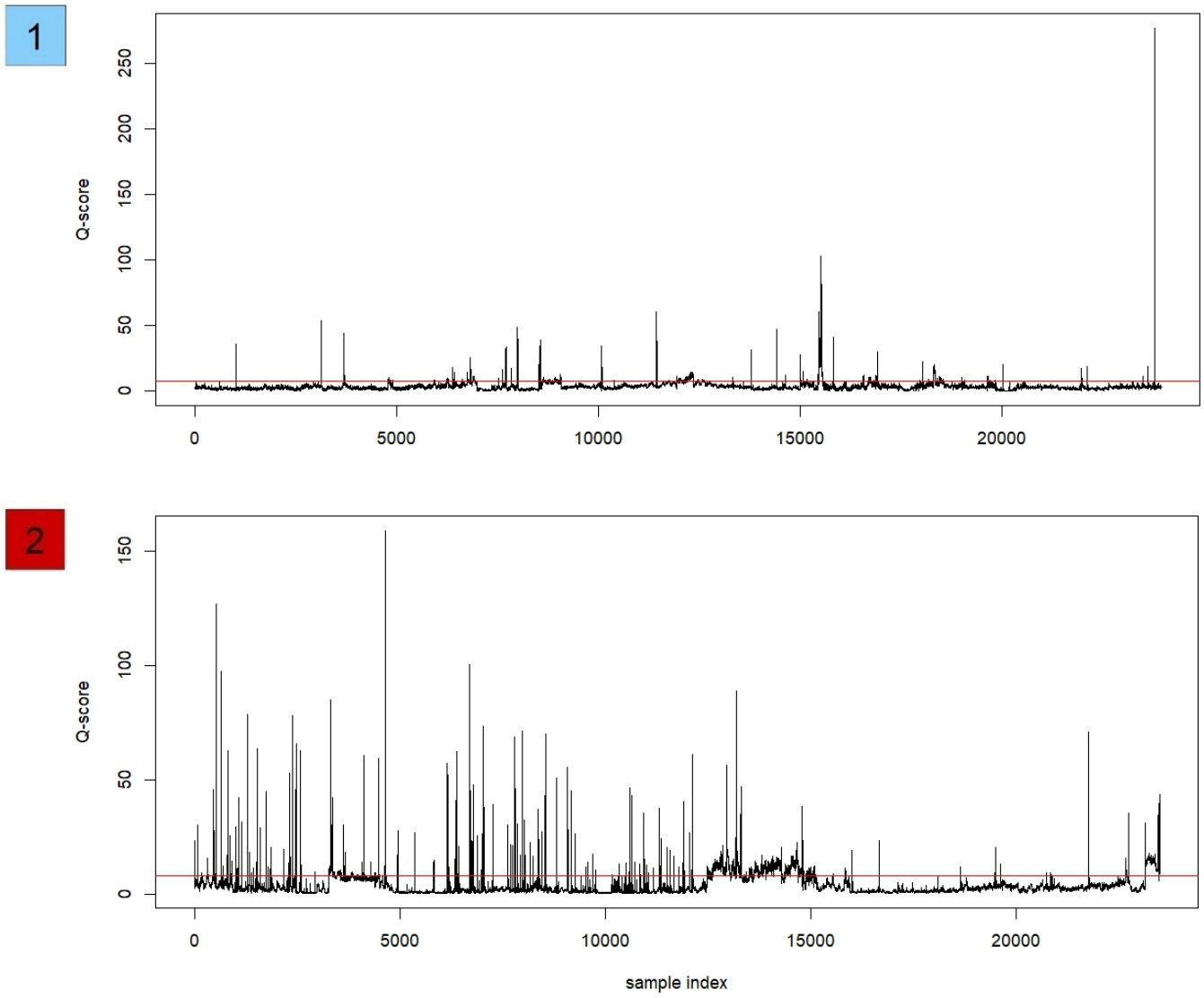

Figure 4. 32 - Q-statistics anomaly scores for individual normal clusters of Huntsman single compressor data

Points that have an anomaly score higher than the Q-statistics threshold, are detected as suboptimal points. The distance between these points and the center other clusters can give important results on the state of the process while monitoring.

A point with high anomaly score in its cluster has is at a distance with the center of the cluster. Taking the physical implication of this point into account, detecting such point in real time means that the process state might be leaving the normal operation zone. If at this period of time, data points generated are approaching the other normal operation zone, it could be interpreted as a shift between normal operation zones. However, if the points are approaching a failure the operator should be notified. A conjecture is also possible to make on what kind of failure is expected to happen based on the cluster that the points are approaching.

Another useful visualization that can be extracted from this method is to form heat maps based on BKDE distributions for the normal operation zones of the process. Such Maps are composed by drawing contours based on BKDE probability calculated only from the data in the normal operation zones. These maps such as Figure 4. 33 are collected and make up a reference to the normal operation of the process in a certain period of time.

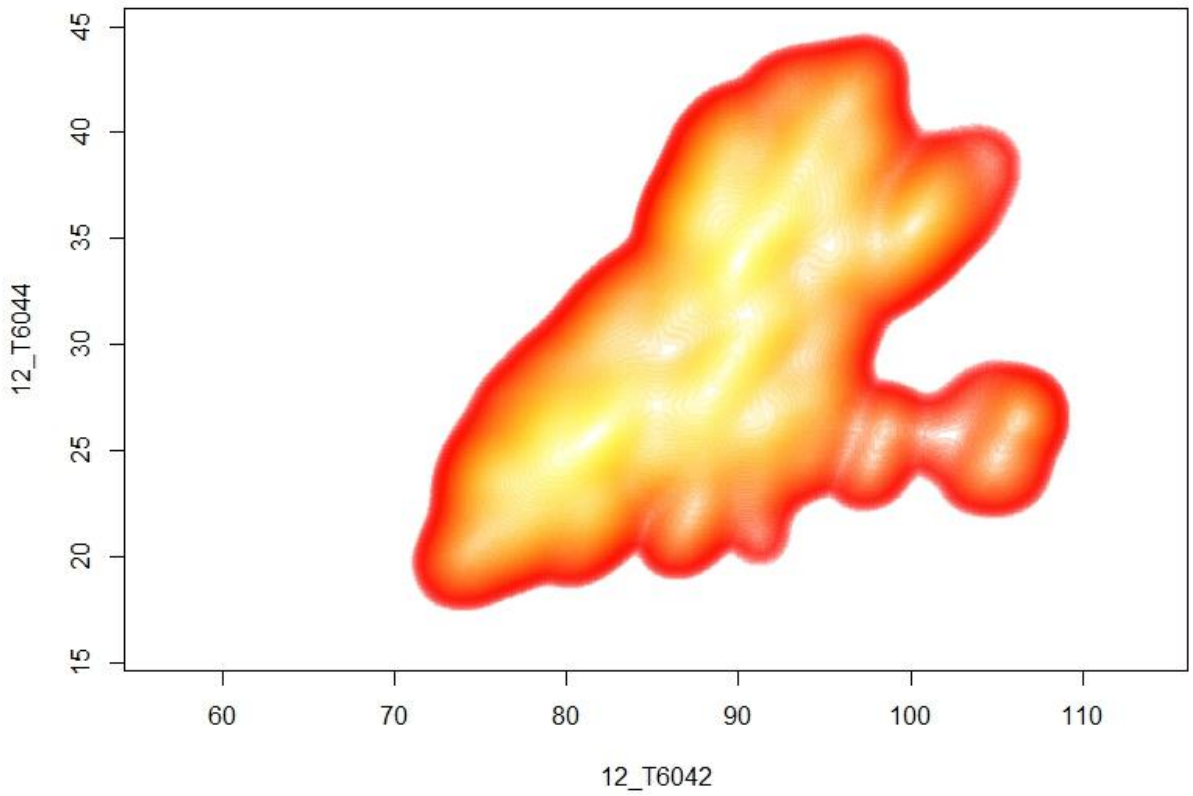

Figure 4. 33 - BKDE based heatmap of normal operation for Huntsman single compressor data

As a verification of the diagnosis based on the fault detection method, the points that are considered noteworthy are marked with an accent color in plots of different measurements against each other. The noteworthy points consist of every point belonging to an abnormal cluster, as well as the points in normal clusters that are considered suboptimal based on the PCA space control charts. Every point is colored according to its GSOM cluster. Additionally, some other points related to known history process are highlighted. These points show the period of time leading to a known failure, as stated by Huntsman process experts. These points are marked by '+' in Figure 4. 34. As it can be seen, many of the points that are specified as leading to a failure, match the accented points based on the fault detection method.

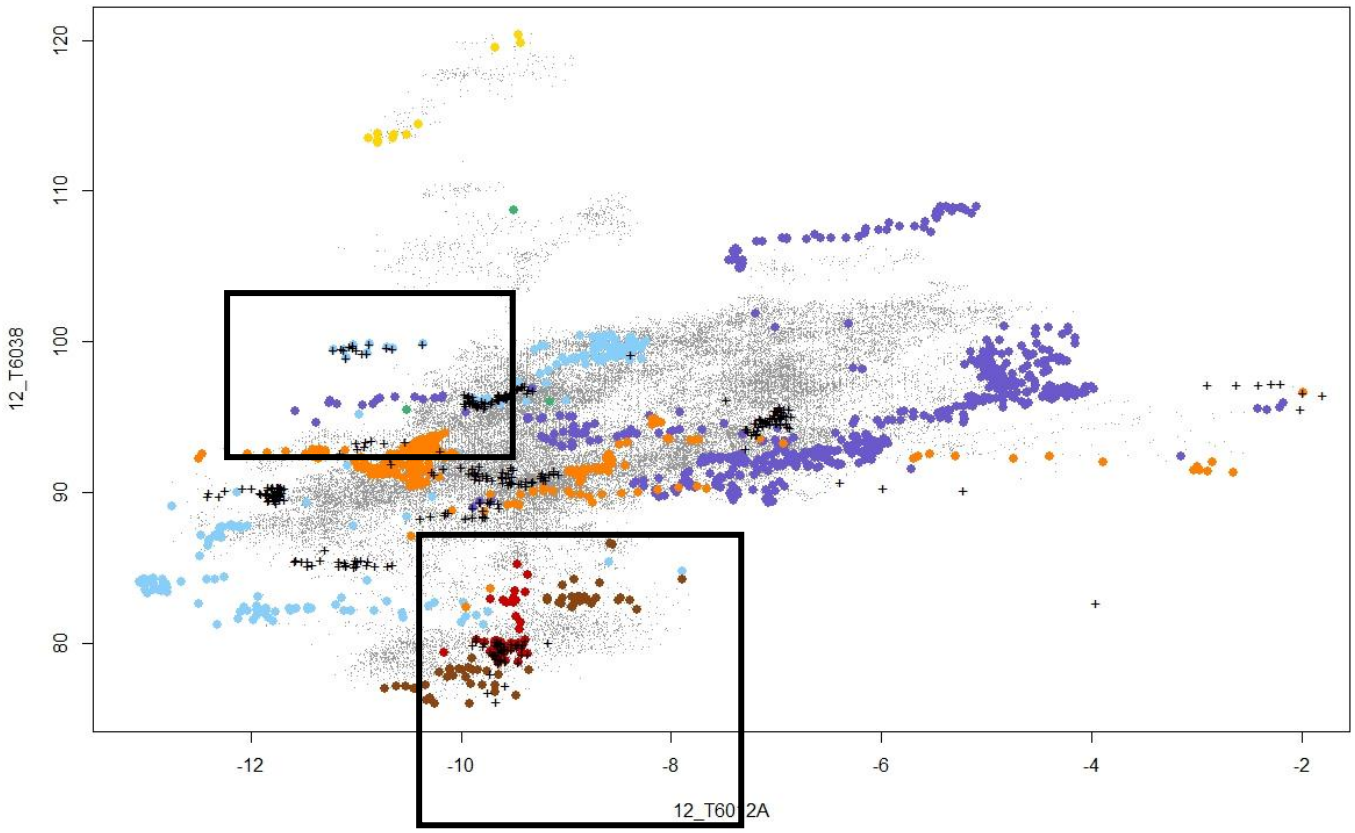

Figure 4. 34 - Comparison between anomalies detected by the method and periods of time leading to failures. Colored points are abnormal points marked by the method, and '+' marks periods leading to failures

A more detailed view of two of these instances is shown below. Figure 4. 35 show a more focused look at the areas of Figure 4. 34. It can be seen that the points determined by the algorithm match exactly the points that are marked as a time period leading to a known failure. Interestingly, both groups belong to a normal cluster, but they are marked due to their high anomaly score in the inside cluster PCA control charts test.

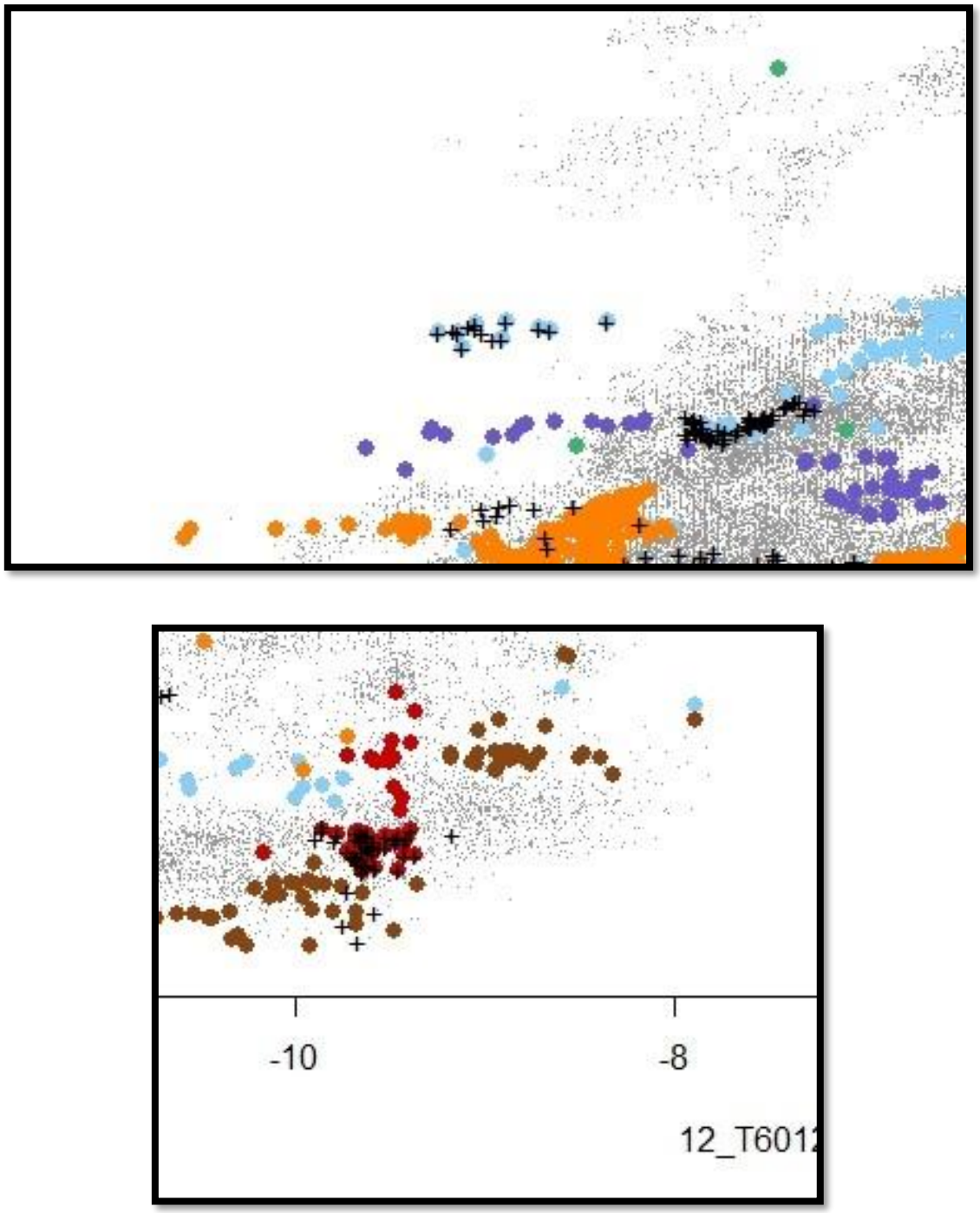

Figure 4. 35 - magnified areas of matching anomaly points inside normal clusters with periods leading to failures

This verifies the importance of monitoring normal operation points for early failure detection. Additionally, it is worth noting that these points are not considered abnormal in PCA control charts developed over all of data, shown in Figure 4. 30.

5. Summary

Big data analytics have gained increasing attention recently due to the advancements in computer technology providing enough computing power for analyzing big datasets. In chemical industry, this attention is focused on a select small group of critical assets that are financially sensitive or are important due to safety reasons. Most of the effort is spent on monitoring these assets, using high-tech devices and model-based approaches to analyze and diagnose these assets. However, less central assets in a process can have as strong effect in improving the process and therefore the profitability of the plant.

This study is dedicated to development of a novel data-driven method in asset fault detection, that is applicable to a wide range of assets. The method is designed to analyze data with no modelling requirements, no assumption on the distribution of data, and no need for pre-defined operation zones or time periods. Additionally, the method is intended to be used by chemical process experts, therefore it requires minimum tuning and parameter definition. The process expert is encouraged to get involved in decision making for the number and nature of clusters based on knowledge on the physical properties of the system as well as the process history.

The method maps multivariate data in a GSOM, and categorizes the data into hypothetical normal operation zones and failure zones using hierarchical clustering. The nature of clusters is estimated by utilizing BKDE distributions and study the anomalies in the correlations between the measurements in every cluster. Observations on BKDE distributions as well as the process expert's opinion validates the clustering and separates the normal operation zone from failure zones. In the end, the normal operation zoned are transformed to PCA space for analysis and calculation of anomaly scores based on Hotelling's T^2 and Q-statistics methods. A guideline for monitoring based on the method is also described in detail.

The method is applied on real process data from a reciprocating compressor, operating at Huntsman chemicals company. The results of analysis for this dataset shows superior results to conventional methods. Decision making for clusters is confirmed by observing plots of real data and the position of GSOM nodes and clusters. Additionally, comparison between abnormal points detected by the method in normal operation zones and points that are known to be leading to a failure from process history shows detection of some of these periods. The method is also evaluated and compared to other methods based on a set of criteria that is defined for evaluation of diagnostic systems.

Acknowledgements

This research was supported by Trendminer research company. I would like to sincerely thank Dr. Stijn Meganck, vice president for research at Trendminer, for providing immaculate insight and expertise that greatly assisted the research, and his considerate supervision on development of the method in every step of the way.

I would like to show my gratitude to Univ.-Prof. Dr.-Ing. Markus Rabe and Dr.-Ing. Dipl.-Inform. Anne Antonia Scheidler for sharing their wisdom and their support during the course of this research.

Special thanks also to Mr. Jasper Rutten and Mr. Chris Waugh, engineers at Huntsman company, for providing data and process information for the case study and for helping me understand the process.

I am also thankful to Mr. Charles Jourquin, research analyst at Trendminer, for his essential assistance with the mathematic fundamentals of this study, as well as comments that greatly improved the report.

Publication bibliography

Ahmed, M.; Gu, F.; Ball, A. D. (2012): Fault Detection of Reciprocating Compressors using a Model from Principles Component Analysis of Vibrations. In *J. Phys.: Conf. Ser.* 364, p. 12133.

Alahakoon, D.; Halgamuge, S. K.; Srinivasan, B. (2000): Dynamic self-organizing maps with controlled growth for knowledge discovery. In *IEEE transactions on neural networks* 11 (3), pp. 601–614.

Andow, P. K. (1980): Fault detection and diagnosis in chemical and petrochemical processes. In *The Chemical Engineering Journal* 20 (1), p. 79.

Bloch, Heinz P. (2006): A practical guide to compressor technology. 2nd ed. Hoboken N.J.: Wiley-Interscience.

Bloch, Heinz P.; Hoefner, John J. (1996): Reciprocating compressors. Operation & maintenance. Houston Tex.: Gulf Pub. Co.

Boustead, I. (2005): Eco-profiles of the European Plastics Industry. Diphenylmethane diisocyanate (MDI). PlasticsEurope. Brussels. Available online at http://www.polyurethanes.org/uploads/documents/eco_midi.pdf, updated on 03/2005, checked on 7/18/2017.

Bro, Rasmus; Smilde, Age K. (2014): Principal component analysis. In *Anal. Methods* 6 (9), pp. 2812–2831.

Cacoullos, Theophilos (1966): Estimation of a multivariate density. In *Annals of the Institute of Statistical Mathematics* 18 (1), pp. 179–189.

Chen, Dezhao; Chen, Yaqiu; Hu, Shangxu (1996): Correlative components analysis for pattern classification. In *Chemometrics and Intelligent Laboratory Systems* 35 (2), pp. 221–229.

Chen, Jinchuan; Chen, Yueguo; Du, Xiaoyong; Li, Cuiping; Lu, Jiaheng; Zhao, Suyun; Zhou, Xuan (2013): Big data challenge. A data management perspective. In *Front. Comput. Sci.* 7 (2), pp. 157–164.

Chen, Xinyi; Yan, Xuefeng (2012): Using improved self-organizing map for fault diagnosis in chemical industry process. In *Chemical Engineering Research and Design* 90 (12), pp. 2262–2277.

Cheng, Yizong (1997): Convergence and Ordering of Kohonen's Batch Map. In *Neural Computation* 9 (8), pp. 1667–1676.

Cline, Daren B. H. (1988): Admissibile Kernel Estimators of a Multivariate Density. In *Ann. Statist.* 16 (4), pp. 1421–1427.

Deheuvels, Paul (1977): Estimation non paramétrique de la densité par histogrammes généralisés. In *Revue de Statistique Appliquée* 25 (3), pp. 5–42.

Delebecq, Etienne; Pascault, Jean-Pierre; Boutevin, Bernard; Ganachaud, François (2013): On the versatility of urethane/urea bonds: reversibility, blocked isocyanate, and non-isocyanate polyurethane. In *Chemical reviews* 113 (1), pp. 80–118.

Dietrich, David; Heller, Barry; Yang, Beibei (Eds.) (2015): Data science & big data analytics. Discovering, analyzing, visualizing and presenting data. EMC Education Services. Indianapolis IN: Wiley.

Dobre, C.; Xhafa, F. (2014): Intelligent services for Big Data science. In *Future Generation Computer Systems* 37, pp. 267–281.

Epanechnikov, V. A. (1969): Non-Parametric Estimation of a Multivariate Probability Density. In *Theory Probab. Appl.* 14 (1), pp. 153–158.

Farzaneh-Gord, Mahmood; Khoshnazar, Hossein (2016): Valve fault detection for single-stage reciprocating compressors. In *Journal of Natural Gas Science and Engineering* 35, pp. 1239–1248.

Forgy, E. (1965): Cluster analysis of multivariate data: efficiency versus interpretability of classifications. In *Biometrics* 21, pp. 768–780.

Fritzke, Bernd (1991): Let It Grow - Self-Organizing Feature Maps With Problem Dependent Cell Structure. In : Artificial Neural Networks: North-Holland, pp. 403–408.

Hair, Joseph F. (2006): Multivariate data analysis. 6th ed. Upper Saddle River N.J.: Pearson Prentice Hall.

Himmelblau, David Mautner (1978): Fault detection and diagnosis in chemical and petrochemical processes. Amsterdam, New York, New York: Elsevier Scientific Pub. Co.; distributors for the US and Canada Elsevier North-Holland (Chemical engineering monographs, v. 8).

Hodges, J. L.; Lehmann, E. L. (1956): The Efficiency of Some Nonparametric Competitors of the t-Test. In *Ann. Math. Statist.* 27 (2), pp. 324–335.

Hotelling, Harold (1931): The Generalization of Student's Ratio. In *Ann. Math. Statist.* 2 (3), pp. 360–378.

Hotelling, Harold (1951): A Generalized T Test and Measure of Multivariate Dispersion. In : Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability. Berkeley, Calif.: University of California Press, pp. 23–41.

Jackson, J. Edward; Mudholkar, Govind S. (1979): Control Procedures for Residuals Associated with Principal Component Analysis. In *Technometrics* 21 (3), p. 341.

Jiang, Qingchao; Yan, Xuefeng; Zhao, Weixiang (2013): Fault Detection and Diagnosis in Chemical Processes Using Sensitive Principal Component Analysis. In *Ind. Eng. Chem. Res.* 52 (4), pp. 1635–1644.

Jolliffe, I. T. (2002): Principal component analysis. 2nd ed. New York: Springer (Springer series in statistics).

Kárný, Miroslav; Warwick, Kevin (1997): Computer Intensive Methods in Control and Signal Processing. The Curse of Dimensionality. N. Boston: Birkhäuser Boston.

Ketelaere, Bart de; Schmitt, Eric: A review of PCA-based statistical process monitoring methods for time-dependent, high-dimensional data.

Kohonen, Teuvo (2001): Self-organizing maps. 3rd ed. Berlin, New York: Springer (Springer series in information sciences, 30).

Kohonen, Teuvo (2013): Essentials of the self-organizing map. In *Neural networks : the official journal of the International Neural Network Society* 37, pp. 52–65.

Kostyukov, V. N.; Naumenko, A. P. (2016): About the Experience in Operation of Reciprocating Compressors Under Control of the Vibration Monitoring System. In *Procedia Engineering* 152, pp. 497–504.

Linde, Y.; Buzo, A.; Gray, R. (1980): An Algorithm for Vector Quantizer Design. In *IEEE Trans. Commun.* 28 (1), pp. 84–95.

Lloyd, S. (1982): Least squares quantization in PCM. In *IEEE Trans. Inform. Theory* 28 (2), pp. 129–137.

Lu, Bo; Stuber, John; Edgar, Thomas F. (2017): Data-driven adaptive multiple model system utilizing growing self-organizing maps. In *Journal of Process Control.*

Lu, Yunsong; Wang, Fuli; Jia, Mingxing; Qi, Yuanchen (2016): Centrifugal compressor fault diagnosis based on qualitative simulation and thermal parameters. In *Mechanical Systems and Signal Processing* 81, pp. 259–273.

Mahalanobis, Prasanta Chandra (1936): On the generalized distance in statistics. In *Proceedings of the National Institute of Sciences (Calcutta)* 2, pp. 49–55.

Mayer-Schönberger, Viktor; Cukier, Kenneth (2013): Big data. A revolution that will transform how we live, work, and think. Boston: Houghton Mifflin Harcourt.

Myler, Harley R. (1998): Fundamentals of engineering programming with C and Fortran. Cambridge, Mass.: Cambridge University Press.

Ohlhorst, Frank (2013): Big data analytics. Turning big data into big money. Hoboken N.J.: Wiley (Wiley & SAS business series).

Oja, Erkki (1992): Principal components, minor components, and linear neural networks. In *Neural Networks* 5 (6), pp. 927–935.

Parzen, Emanuel (1962): On Estimation of a Probability Density Function and Mode. In *Ann. Math. Statist.* 33 (3), pp. 1065–1076.

Pearson, Karl (1901): LIII. On lines and planes of closest fit to systems of points in space. In *Philosophical Magazine Series 6* 2 (11), pp. 559–572.

Penha, Rosani M. L.; Hines, J. (2001): Using Principal Component Analysis Modeling to Monitor Temperature Sensors in a Nuclear Research Reactor.

Potočnik, Primož; Govekar, Edvard (2017): Semi-supervised vibration-based classification and condition monitoring of compressors. In *Mechanical Systems and Signal Processing* 93, pp. 51–65.

Raspoet, G.; Nguyen, M. T. (1998): Experimental and Theoretical Evidence for a Concerted Catalysis by Water Clusters in the Hydrolysis of Isocyanates. In *The journal of organic chemistry* 63 (20), pp. 6867–6877.

Rosenblatt, Murray (1956): Remarks on Some Nonparametric Estimates of a Density Function. In *Ann. Math. Statist.* 27 (3), pp. 832–837.

Rotem, Y.; Wachs, A.; Lewin, D. R. (2000): Ethylene compressor monitoring using model-based PCA. In *AIChE J.* 46 (9), pp. 1825–1836.

Saha, B. (2017): Big Data in Chemical Industry. In *Chemistry International* 39 (3), p. 42.

Sarlin, Peter (2013): Self-organizing time map. An abstraction of temporal multivariate patterns. In *Neurocomputing* 99, pp. 496–508.

Schultheis, S. M.; Lickteig, C. A.; Parchewsky, R. (2007): Reciprocating compressor condition monitoring. In *proceedings of the thirty-sixth turbomachinery symposium* 36, pp. 107–114.

Scott, David W. (2015): Multivariate Density Estimation. Hoboken, NJ: John Wiley & Sons, Inc.

Silverman, B. W. (1982): Kernel Density Estimation Using the Fast Fourier Transform. In *Applied Statistics* 31 (1), p. 93.

Silverman, B. W. (1998): Density estimation for statistics and data analysis. Boca Raton: Chapman & Hall/CRC (Monographs on statistics and applied probability, 26).

Sivarajah, Uthayasankar; Kamal, Muhammad Mustafa; Irani, Zahir; Weerakkody, Vishanth (2017): Critical analysis of Big Data challenges and analytical methods. In *Journal of Business Research* 70, pp. 263–286.

Snedecor, George W.; Cochran, William G. (1989): Statistical methods. 6th ed. Iowa City: Iowa State University Press

Sonnenschein, Mark F. (2015): Polyurethanes. Science, technology, markets, and trends. Hoboken New Jersey: Wiley.

Student (1908): The Probable Error of a Mean. In *Biometrika* 6 (1), p. 1. Venkatasubramanian, Venkat; Rengaswamy, Raghunathan; Kavuri, Surya N.; Yin, Kewen (2003a): A review of process fault detection and diagnosis. Part III: Process history based methods. In *Computers & Chemical Engineering* 27 (3), pp. 327–346.

Venkatasubramanian, Venkat; Rengaswamy, Raghunathan; Yin, Kewen; Kavuri, Surya N. (2003b): A review of process fault detection and diagnosis. Part I: Quantitative model-based methods. In *Computers & Chemical Engineering* 27 (3), pp. 293–311.

Voegtlin, Thomas; Dominey, Peter F. (2001): Recursive Self-Organizing Maps. In : Advances in Self-Organising Maps. London: Springer London, pp. 210–215.

Wand, M. P.; Jones, M. C. (1995): Kernel smoothing. 1st ed. London, New York: Chapman & Hall (Monographs on statistics and applied probability, 60).

Willems, G.; Pison, G.; Rousseeuw, P. J.; van Aelst, S. (2002): A robust Hotelling test. In *Metrika* 55 (1), pp. 125–138.

Willsky, Alan S. (1976): A survey of design methods for failure detection in dynamic systems. In *Automatica* 12 (6), pp. 601–611.

Witten, I. H.; Frank, Eibe; Hall, Mark A. (2011): Data Mining. Practical Machine Learning Tools and Techniques. San Diego, CA, USA: Elsevier Science.

Yi, Xiaomeng; Liu, Fangming; Liu, Jiangchuan; Jin, Hai (2014): Building a network highway for big data. Architecture and challenges. In *IEEE Network* 28 (4), pp. 5–13.

Zhang, Feng; Liu, Min; Gui, Feng; Shen, Weiming; Shami, Abdallah; Ma, Yunlong (2015): A distributed frequent itemset mining algorithm using Spark for Big Data analytics. In *Cluster Comput* 18 (4), pp. 1493–1501.

Appendix

```
##############################################
# R script for asset anomaly detection method
##############################################

# defining the input training dataframe, input timeframe, and known failures dataframe

X <- normal_data2
gtime <- normal_time2
fX <- fail_data

# preprocessing

Xs <- scale(X)
d <- ncol(Xs)
f <- nrow (fX)
Xm <- matrix(ncol=d, nrow=1)
Xdev <- matrix(ncol=d, nrow=1)
for (i in 1:d){
  Xm[,i] <- mean(X[,i])
  Xdev[,i] <- sd(X[,i])
}

# process variables definition

a <- 0.95   #control charts threshold
SF <- 0.9   #GSOM spread factor
AT <- 6     #BKDE anomaly score threshold


# Color palette definition

pretty_palette <- c('lightskyblue', 'red3', 'slateblue3', 'darkorange1',
                    'mediumseagreen', 'gold1', 'chocolate4','salmon','navy')

heatc <- c('green','red')

##################################################
# Bivariate Kernel Density Estimation
##################################################

library(KernSmooth)
library(lattice)
library(fields)
library(rgl)
library(scatterD3)
library(dygraphs)
library(R.basic)

# BKDE distributions for every possible combination of two variables in dataframe
Y <- X
n <- nrow(Y)
LI=list()
Lii <- data.frame(NULL)
k=1
for (i in 2:d){
  for (j in 1:(i-1)){
    h1=sd(Y[,j])/(n^(1/6))
    h2=sd(Y[,i])/(n^(1/6))
    M=cbind(Y[,j],Y[,i])
    L=bkde2D(M,bandwidth=c(h1,h2),gridsize=c(256,256))
    LI[[k]]=L
    t=interp.surface(list(x=L$x1,y=L$x2,z=L$fhat),loc=M)
    tstar=max(L$fhat)
    AST=-log(t/tstar)
    Lii <- rbind(Lii,new=c(i,j,k))
    k=k+1
  }
}
k=1
```

```r
Lii <- as.matrix(Lii)
colnames(Lii) <- c('i','j','k')

################################################
# Growing Self-organizing map
################################################

library(GrowingSOM)
library(KernSmooth)
library(lattice)
library(fields)
library(rgl)
library(scatterD3)
library(dygraphs)

# training the GSOM and plotting the related figures

Gtrain_map <- train.gsom(Xs, spreadFactor=SF)

jpeg(filename = paste0("GSOM_grid.jpg"),
     width = 650, height = 650, units = "px", pointsize = 16,
     quality = 100)
plot(Gtrain_map, type="count")
dev.off()

# Estimate the number of clusters that suits the grid

wss <- (nrow(Gtrain_map$nodes$codes)-1)*sum(apply(Gtrain_map$nodes$codes,2,var))
for (i in 2:15) wss[i] <- sum(kmeans(Gtrain_map$nodes$codes,
                                     centers=i)$withinss)
par(mar=c(5.1,4.1,4.1,2.1))
jpeg(filename = paste0("WCSS.jpg"),
     width = 800, height = 600, units = "px", pointsize = 16,
     quality = 100)
plot(1:15, wss, type="b", xlab="Number of Clusters",
     ylab="Within groups sum of squares", main="Within cluster sum of squares (WCSS)")
dev.off()

gcodesmat <- Gtrain_map$nodes$codes
Gtest_map <- map.gsom(Gtrain_map, Xs)
Gtest_node <- Gtest_map$mapped$bmn

fXs <- matrix(nrow=f,ncol=d)
for (j in 1:f){
  for (i in 1:d){
    fXs[j,i] <- (fX[j,i]-Xm[,i])/Xdev[,i]
  }
}

Gfail_map <- map.gsom(Gtrain_map, fXs)
Gfail_node <- Gfail_map$mapped$bmn

# comparison of the data time-series with different clustering variables

CLUSTER_data <- list()
for (i in 1:9){
  gsom_cluster <- cutree(hclust(dist(gcodesmat)), i)
  Gtest_cluster <- gsom_cluster[Gtest_node]
  Gtest_col <- pretty_palette[Gtest_cluster]
  CLUSTER_data[[i]] <- Gtest_cluster
  jpeg(filename = paste0("MDI_",i,"_clusters.jpg"),
       width = 800, height = 600, units = "px", pointsize = 16,
       quality = 100)
  plot(gtime, X[,d], pch=20 ,col=Gtest_col)
  dev.off()
}

NC <- 8

```

```
# clustering the grid

gsom_cluster <- cutree(hclust(dist(gcodesmat)), NC)
gsom_col <- pretty_palette[gsom_cluster]
Gtest_cluster <- gsom_cluster[Gtest_node]
Gtest_col <- pretty_palette[Gtest_cluster]
Gfail_cluster <- gsom_cluster[Gfail_node]
gcodesreal <- gcodesmat
for (j in 1:d){
  gcodesreal[,j] <- (gcodesmat[,j]*Xdev[,j])+Xm[,j]
}

Nd <-nrow(Gtrain_map$nodes$position)

# Finding the center of each cluster and checking for their anomaly according to BKDE

cl_mean <- matrix(data=0, nrow=NC, ncol=d)
Cl_mat <- as.matrix(gsom_cluster)
for (i in 1:NC){
  cl <- subset(gcodesreal, Cl_mat==i)
  for (j in 1:d){
    cl_mean[i,j] <- mean(cl[,j])
  }
}

Cl_check <- matrix(data=0, nrow=NC,ncol=nrow(Lii))
Cl_val <- matrix(data=0, nrow=NC,ncol=nrow(Lii))

k=1
for (i in 2:d){
  for (j in 1:(i-1)){
    M=cbind(cl_mean[,j],cl_mean[,i])
    L=LI[[k]]
    t=interp.surface(list(x=L$x1,y=L$x2,z=L$fhat),loc=M)
    tstar=max(L$fhat)
    AST=-log(t/tstar)
    Cl_check[,k] <- AST > AT
    Cl_val[,k] <- AST
    k=k+1
  }
}
k=1

Cl_check[is.na(Cl_check)] <- 1
Cl_con <- list()
for (c in 1:NC){
  cll_ch <- matrix(data = NA,ncol=d,nrow=d)
  for (i in 2:d){
    for (j in 1:(i-1)){
      cll_ch[j,i] <- Cl_check[c,k]
      k <- k+1
    }
  }
  k=1
  Cl_con[[c]] <- cll_ch
  colnames(Cl_con[[c]]) <- colnames(X)
  rownames(Cl_con[[c]]) <- colnames(X)
}

# plotting the cluster anomaly control boards based on BKDE

for (i in 1:NC){

  test <- Cl_con[[i]]
  jpeg(filename = paste0("Cluster_",i,"_control.jpg"),
       width = 900, height = 600, units = "px", pointsize = 16,
       quality = 100)
  heatmap(t(test),col=heatc,Rowv = NA,Colv = NA,symm = TRUE,
          main=paste0("Cluster ",i," Control"),margins = c(6, 6))
```

```
  dev.off()
}

# plotting GSOM nodes on top of variable biplot with color coding for clusters

for (i in 2:d){
  for (j in 1:(i-1)){
    jpeg(filename = paste0("GSOM_variables_",j,"_",i,".jpg"),
         width = 1200, height = 800, units = "px", pointsize = 16,
         quality = 100)
    plot(X[,c(j,i)],pch='.',col='gray40')
    points(gcodesreal[,c(j,i)],pch=paste0(gsom_cluster),cex=1.4,col=gsom_col)
    dev.off()
  }
}

###################################
# Principal component analysis
###################################

library(ICSNP)
library(devtools)
library(ggplot2)
library(ggbiplot)

# performing PCA on the training set a whole

m <- nrow(X)
T2 <- matrix(ncol=1, nrow=n)
Q <- matrix(ncol=1, nrow=n)
comp_PCA <- princomp (X, scores=TRUE, cor=TRUE)


jpeg(filename = "PCs.jpg",
     width = 800, height = 800, units = "px", pointsize = 12,
     quality = 100)
plot(comp_PCA)
dev.off()

k <- 0
c <- comp_PCA$sdev[1]

while (c >= 1){
  k <- k+1
  c <- comp_PCA$sdev[k+1]
}
if (k==1){k <- 2}

jpeg(filename = "PCs_sdev.jpg",
     width = 700, height = 600, units = "px", pointsize = 12,
     quality = 100)
plot(comp_PCA$sdev, pch=16,cex=1.6,col="blue", ylab="Standard deviation", xlab="PC
count", cex.lab=1.4)
abline(h=1, col="red", lwd=2)
text(comp_PCA$sdev[1:k], labels=round(comp_PCA$sdev[1:k], digits = 3), cex= 1, pos=4)
dev.off()

# Calculating the parameters for control charts

V <- (comp_PCA$sdev^2)
P <- comp_PCA$loadings[,1:k]
Pt <- t(P)
L <- diag(V[1:k])
L1 <- solve(L)
TR3 <- diag(d)
F <- qf(a, df1=k, df2=m-k)
Ta2 <- (k*((m^2)-1)/(m*(m-k)))*F


```

```
teta <- matrix(data=0,ncol=1, nrow=3)
for (i in 1:3){
  for (j in (k+1):d){
    teta[i] <- teta[i]+V[j]^i
  }
}

h0 <- 1-((2*(teta[1]*teta[3]))/(3*(teta[2]^2)))
za <- qnorm (a, mean=0, sd=1)
Qa <-
teta[1]*(((za*((2*teta[2]*(h0^2))^0.5)/teta[1])+1+((teta[2]*h0*(1-h0))/(teta[2]^2)))^(1/h
0))


for (j in 1:n){
  Bt <- Xs[j,]
  B <- t(Bt)

  #T2 Hotelling statistic
  Y <- Pt %*% Bt
  Yt <- t(Y)
  TR <- L1 %*% Y
  T2[j] <- Yt %*% TR
  #Q-statistic
  TR2 <- P %*% Pt
  TR4 <- TR3 - TR2
  TR5 <- TR4 %*% Bt
  Q[j,] <- B %*% TR5
}

# plotting the control charts for all of the data

jpeg(filename = "Hotelling.jpg",
     width = 1200, height = 600, units = "px", pointsize = 18,
     quality = 100)
plot.ts(T2, main="Hotelling test",pch=20,cex=0.8,xlab="sample index",ylab="T^2 score")
abline(h=Ta2, col="red")
dev.off()

jpeg(filename = "Q-statistics.jpg",
     width = 1200, height = 600, units = "px", pointsize = 18,
     quality = 100)
plot.ts(Q, main="Q-statistics",pch=20,cex=0.8, xlab="sample index", ylab="Q-score")
abline(h=Qa, col="red")
dev.off()

nQ <- log(Q/Qa)
nT2<- log(T2/Ta2)

Ylm <- max(abs(nQ),abs(nT2))

jpeg(filename = "anomaly comparison.jpg",
     width = 800, height = 600, units = "px", pointsize = 18,
     quality = 100)
plot(nQ,type="o", main="normalized anomaly score",pch='.',col='chartreuse',
     xlab="sample index", ylab="Normalized anomaly score",ylim=c(-Ylm,Ylm))
points(nT2,type="o", main="Hotelling test",col='coral',pch='.',xlab="sample index",
       ylab="T^2 score")
legend((n-12500),Ylm, c("Q-statistics","T squared"),
       lty=c(1,1),lwd=c(2.5,2.5),col=c('chartreuse','coral'))
abline(h=0,col='blue')
dev.off()

# control charts plot with color coding clusters

jpeg(filename = paste0("Qstat_",NC,"_clusters.jpg"),
     width = 1200, height = 600, units = "px", pointsize = 18,
     quality = 100)
plot(gtime$date, Q, main="Q-statistic test",pch=20, col=Gtest_col,
```

```
343         cex=0.8,xlab="sample index",ylab="Q-statistic score")
344   abline(h=Qa, col="red")
345   dev.off()
346
347   jpeg(filename = paste0("Hotelling_",NC,"_clusters.jpg"),
348         width = 1200, height = 600, units = "px", pointsize = 18,
349         quality = 100)
350   plot(gtime$date, T2, main="Hotelling test",pch=20, col=Gtest_col,
351         cex=0.8,xlab="sample index",ylab="T^2 score")
352   abline(h=Ta2, col="red")
353   dev.off()
354
355   # plotting PC biplot with color coding for clusters
356
357   for (i in 2:k){
358     for (j in 1:(i-1)){
359
        qplot(comp_PCA$scores[,c(i)],comp_PCA$scores[,c(j)],xlab=paste("Component",i),ylab=pa
        ste("Component",j),col = I(Gtest_col) ,asp=1)
360       ggsave(paste0("_components_",i,"_",j,".jpg"), device = "jpeg")
361     }
362   }
363
364   # Distinguishing the clusters as normal operation and abnormal
365
366   PCAs <- list()
367   Anom_cluster <- list()
368   Anom <- list()
369   Anomt <- list()
370   QAnom <- list()
371   QAnomt <- list()
372   Tas2 <- matrix(data=0, nrow=NC, ncol=1)
373   Qas2 <- matrix(data=0, nrow=NC, ncol=1)
374   NNC <- c(1,2)          # normal cluster numbers
375   ANC <- C(3,4,5,6,7,8) # anomaly cluster numbers
376
377   # Developing control charts for normal cluster
378
379   for (i in 1:length(NNC)){
380
381     clusp <- subset(X, Gtest_cluster == NNC[i])
382     clust <- subset(gtime, Gtest_cluster == NNC[i])
383     clusps <- scale(clusp)
384     PCAs[[i]] <- princomp (clusp, scores=TRUE, cor=FALSE)
385
386     m <- nrow(clusp)
387     T2 <- matrix(ncol=1, nrow=m)
388     Q <- matrix(ncol=1, nrow=m)
389
390     jpeg(filename = paste0("PCs_cluster_",i,".jpg"),
391           width = 800, height = 800, units = "px", pointsize = 12,
392           quality = 100)
393     plot(PCAs[[i]])
394     dev.off()
395
396     k <- 0
397     c <- PCAs[[i]]$sdev[1]
398
399     while (c >= 1){
400         k <- k+1
401         c <- PCAs[[i]]$sdev[k+1]
402     }
403     if (k==1){k <- 2}
404
405     jpeg(filename = paste0("PCs_sdev_cluste_",i,".jpg"),
406           width = 700, height = 600, units = "px", pointsize = 12,
407           quality = 100)
408     plot(PCAs[[i]]$sdev, pch=16,cex=1.6,col="blue", ylab="Standard deviation", xlab="PC
      count", cex.lab=1.4)
```

```
409     abline(h=1, col="red", lwd=2)
410     text(PCAs[[i]]$sdev[1:k], labels=round(PCAs[[i]]$sdev[1:k], digits = 3), cex= 1, pos=4)
411     dev.off()
412
413     V <- (PCAs[[i]]$sdev^2)
414     P <- PCAs[[i]]$loadings[,1:k]
415     Pt <- t(P)
416     L <- diag(V[1:k])
417     L1 <- solve(L)
418     TR3 <- diag(d)
419     F <- qf(a, df1=k, df2=m-k)
420     Tas2[i,] <- (k*((m^2)-1)/(m*(m-k)))*F
421
422     teta <- matrix(data=0,ncol=1, nrow=3)
423     for (t in 1:3){
424       for (j in (k+1):d){
425           teta[t] <- teta[t]+V[j]^i
426       }
427     }
428
429     h0 <- 1-((2*(teta[1]*teta[3]))/(3*(teta[2]^2)))
430     za <- qnorm (a, mean=0, sd=1)
431     Qas2[i,] <-
        teta[1]*(((za*((2*teta[2]*(h0^2))^0.5)/teta[1])+1+((teta[2]*h0*(1-h0))/(teta[2]^2)))^(1
        /h0))
432
433     for (j in 1:m){
434       Bt <- clusps[j,]
435       B <- t(Bt)
436
437       #T2
438       Y <- Pt %*% Bt
439       Yt <- t(Y)
440       TR <- L1 %*% Y
441       T2[j] <- Yt %*% TR
442       #Q
443       TR2 <- P %*% Pt
444       TR4 <- TR3 - TR2
445       TR5 <- TR4 %*% Bt
446       Q[j,] <- B %*% TR5
447       }
448
449     jpeg(filename = paste0("Hotelling_cluster_",i,".jpg"),
450          width = 1200, height = 600, units = "px", pointsize = 18,
451          quality = 100)
452     plot.ts(T2, main="Hotelling test",pch=20,cex=0.8,xlab="sample index",ylab="T^2 score")
453     abline(h=Tas2[i], col="red")
454     dev.off()
455
456     jpeg(filename = paste0("Q-statistics_cluster_",i,".jpg"),
457          width = 1200, height = 600, units = "px", pointsize = 18,
458          quality = 100)
459     plot.ts(Q, main="Q-statistics",pch=20,cex=0.8, xlab="sample index", ylab="Q-score")
460     abline(h=Qas2[i,], col="red")
461     dev.off()
462
463     Anom[[NNC[i]]] <- subset(clusp, T2 > Tas2[i])
464     Anomt[[NNC[i]]] <- subset(clust, T2 > Tas2[i])
465
466     QAnom[[NNC[i]]] <- subset(clusp, Q > Qas2[i])
467     QAnomt[[NNC[i]]] <- subset(clust, Q > Qas2[i])
468   }
469
470   # Adding the abnormal clusters to the anomaly values from normal clusters
471
472   for (i in 1:(NC-(length(NNC)+1))){
473     clusp <- subset(X, Gtest_cluster == ANC[i])
474     clust <- subset(gtime, Gtest_cluster == ANC[i])
475     clusps <- scale(clusp)
```

```
476
477     Anom[[ANC[i]]] <- clusp
478     Anomt[[ANC[i]]] <- clust
479
480     QAnom[[ANC[i]]] <- clusp
481     QAnomt[[ANC[i]]] <- clust
482
483   }
484
485   # generating biplots with the anomaly points color coded according to cluster
486
487   for (i in 2:d){
488     for (j in 1:(i-1)){
489       jpeg(filename = paste0("cluster_T2_anomalies_",j,"_",i,".jpg"),
490            width = 1200, height = 800, units = "px", pointsize = 16,
491            quality = 100)
492       plot(X[,c(j,i)],pch='.',col='gray60')
493       for (z in 1:NC){
494         points(Anom[[z]][,c(j,i)],pch=20,cex=1.6,col=pretty_palette[z])
495       }
496       points(fX[,c(j,i)],pch='+',cex=1,col='black')
497       dev.off()
498     }
499   }
500
501   # Heat-map biplots for BKDE distributions
502
503   k=1
504   for (i in 2:d){
505     for (j in 1:(i-1)){
506       L=LI[[k]]
507       tstar=max(L$fhat)
508       jpeg(filename = paste0("Kernel_map_test_",i,"_",j,".jpg"),
509            width = 1200, height = 800, units = "px", pointsize = 16,
510            quality = 100)
511       contour(x=L$x1,y=L$x2,z=-log((L$fhat)/tstar),
512               levels=seq(0,6,0.1),labcex = 1.8,lwd = 8,
513               drawlabels = FALSE,
514               vfont = c("sans serif", "bold"),
515               col=rgb(1,1-(seq(0,6,0.1)/6),0,alpha=0.2),
516               xlab=colnames(X[j]),ylab=colnames(X[i]))
517       points(X[,j],X[,i],pch='.',cex=0.5,col='black')
518       dev.off()
519       k=k+1
520     }
521   }
522
523
524   # Heat-maps with anomaly plot combination based on T2
525   k=1
526
527   for (i in 2:d){
528     for (j in 1:(i-1)){
529
530       L=LI[[k]]
531       tstar=max(L$fhat)
532       jpeg(filename = paste0("Kernel_map_all_",i,"_",j,".jpg"),
533            width = 1200, height = 800, units = "px", pointsize = 16,
534            quality = 100)
535       contour(x=L$x1,y=L$x2,z=-log((L$fhat)/tstar),
536               levels=seq(0,6,0.1),labcex = 1.8,lwd = 8,
537               drawlabels = FALSE,
538               vfont = c("sans serif", "bold"),
539               col=rgb(1,1-(seq(0,6,0.1)/6),0,alpha=0.1),
540               xlab=colnames(X[j]),ylab=colnames(X[i]))
541       points(X[,j],X[,i],pch='.',cex=0.5,col='gray60')
542       for (z in 1:NC){
543         points(Anom[[z]][,c(j,i)],pch=20,cex=1.6,col=pretty_palette[z])
544       }
```

```
    points(fX[,c(j,i)],pch='+',cex=1,col='black')
    dev.off()
    k=k+1
  }
}


# Heat-maps with anomaly plot combination based on Q statistics
k=1

for (i in 2:d){
  for (j in 1:(i-1)){

    L=LI[[k]]
    tstar=max(L$fhat)
    jpeg(filename = paste0("Kernel_map_all_",i,"_",j,".jpg"),
         width = 1200, height = 800, units = "px", pointsize = 16,
         quality = 100)
    contour(x=L$x1,y=L$x2,z=-log((L$fhat)/tstar),
            levels=seq(0,6,0.1),labcex = 1.8,lwd = 8,
            drawlabels = FALSE,
            vfont = c("sans serif", "bold"),
            col=rgb(1,1-(seq(0,6,0.1)/6),0,alpha=0.1),
            xlab=colnames(X[j]),ylab=colnames(X[i]))
    points(X[,j],X[,i],pch='.',cex=0.5,col='gray60')
    for (z in 1:NC){
      points(Anom[[z]][,c(j,i)],pch=20,cex=1.6,col=pretty_palette[z])
    }
    points(fX[,c(j,i)],pch='+',cex=1,col='black')
    dev.off()
    k=k+1
  }
}
k=1
```

Declaration in Lieu of Oath

_____ _____

Last name, First name Student ID:

I herebey declare in lieu of oath that I have composed the enclosed bachelor´s/master´s* thesis

entitled _____

_____

_____

entirely on my own and without any inadmissible help from outside. I have not used any outside sources without declaration in the text. Any concepts or quotations applicable to these sources are clearly attributed to them.

This master´s thesis has not been submitted in the same or substantially similar version, not even in part, to any other authority for grading.

_____ _____

Place, date Signature

*Please delete where inapplicable

Instruction:

Anyone who intentionally violates a rule in a university´s examination regulations on cheating in examinations is committing a regulatory offence. This offence may be penalized with a fine of up to 50,000 euros. The administrative authority responsible for the prosecution and punishment of an offence is the Head of Administration (Kanzler) of TU Dortmund University. Furthermore, in case of a repeated or any other serious attempt to deceive, the examinee may be exmatriculated (§ 63 paragraph 5 Higher Education Act).

A willfully false declaration in lieu of oath may be punished with imprisonment of up to 3 years or a fine.

TU Dortmund University may use intellectual property verification tools (e.g. "turnitin") to check for regulatory offences in examination procedures.

I have noted the instructions given above:

_____ _____

Place, date signature