MÁSTER UNIVERSITARIO EN INGENIERÍA INDUSTRIAL

TRABAJO FIN DE MÁSTER

UNSUPERVISED LEARNING FOR CONDITION MONITORING

Autor: Álvaro Gómez Asensio

Director: Prof. Dr.-Ing. Michael F. Zäh

Madrid

Julio de 2019

I, hereby, declare that I am the only author of the project report with title:

Unsupervised Learning for Condition Monitoring

which has been submitted to ICAI School of Engineering of Comillas Pontifical University in the academic year 2018/19. This project is original, has not been submitted before for any other purpose and has not been copied from any other source either fully or partially. All information sources used have been rightly acknowledged.

Fdo.: Álvaro Gómez Asensio Date: 18/07/2019

I authorize the submission of this project

PROJECT SUPERVISOR

Fdo.: Prof. Dr.-Ing. Michael F. Zäh Date: 01 / 07 / 2019

AUTHORIZATION FOR DIGITALIZATION, STORAGE AND DISSEMINATION IN THE NETWORK OF END-OF-DEGREE PROJECTS, MASTER PROJECTS, DISSERTATIONS OR BACHILLERATO REPORTS

1. Declaration of authorship and accreditation thereof.

The author Mr. /Ms. Álvaro Gómez Asensio

HEREBY DECLARES that he/she owns the intellectual property rights regarding the piece of work: Unsupervised Learning for Condition Monitoring

that this is an original piece of work, and that he/she holds the status of author, in the sense granted by the Intellectual Property Law.

2. Subject matter and purpose of this assignment.

With the aim of disseminating the aforementioned piece of work as widely as possible using the University's Institutional Repository the author hereby **GRANTS** Comillas Pontifical University, on a royalty-free and non-exclusive basis, for the maximum legal term and with universal scope, the digitization, archiving, reproduction, distribution and public communication rights, including the right to make it electronically available, as described in the Intellectual Property Law. Transformation rights are assigned solely for the purposes described in a) of the following section.

3. Transfer and access terms

Without prejudice to the ownership of the work, which remains with its author, the transfer of rights covered by this license enables:

a) Transform it in order to adapt it to any technology suitable for sharing it online, as well as including metadata to register the piece of work and include "watermarks" or any other security or protection system.

b) Reproduce it in any digital medium in order to be included on an electronic database, including the right to reproduce and store the work on servers for the purposes of guaranteeing its security, maintaining it and preserving its format.

c) Communicate it, by default, by means of an institutional open archive. which has open and cost-free online access.

d) Any other way of access (restricted, embargoed, closed) shall be explicitly requested and requires that good cause be demonstrated.

e) Assign these pieces of work a Creative Commons license by default.

f) Assign these pieces of work a HANDLE (*persistent* URL). by default.

4. Copyright.

The author, as the owner of a piece of work, has the right to:

a) Have his/her name clearly identified by the University as the author

b) Communicate and publish the work in the version assigned and in other subsequent versions using any medium.

c) Request that the work be withdrawn from the repository for just cause.

d) Receive reliable communication of any claims third parties may make in relation to the work and, in particular, any claims relating to its intellectual property rights.

5. Duties of the author.

The author agrees to:

a) Guarantee that the commitment undertaken by means of this official document does not infringe any third party rights, regardless of whether they relate to industrial or intellectual property or any other type.

b) Guarantee that the content of the work does not infringe any third party honor, privacy or image rights.
c) Take responsibility for all claims and liability, including compensation for any damages, which may be brought against the University by third parties who believe that their rights and interests have been infringed by the assignment.
d) Take responsibility in the event that the institutions are found guilty of a rights infringement regarding the work subject to assignment.

6. Institutional Repository purposes and functioning.

The work shall be made available to the users so that they may use it in a fair and respectful way with regards to the copyright, according to the allowances given in the relevant legislation, and for study or research purposes, or any other legal use. With this aim in mind, the University undertakes the following duties and reserves the following powers:

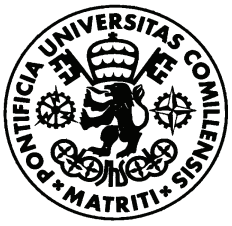
a) The University shall inform the archive users of the permitted uses; however, it shall not guarantee or take any responsibility for any other subsequent ways the work may be used by users, which are non-compliant with the legislation in force. Any subsequent use, beyond private copying, shall require the source to be cited and authorship to be recognized, as well as the guarantee not to use it to gain commercial profit or carry out any derivative works.
b) The University shall not review the content of the works, which shall at all times fall under the exclusive responsibility of the author and it shall not be obligated to take part in lawsuits on behalf of the author in the event of any infringement of intellectual property rights deriving from storing and archiving the works. The author hereby waives any claim against the University due to any way the users may use the works that is not in keeping with the legislation in force.
c) The University shall adopt the necessary measures to safeguard the work in the future.
d) The University reserves the right to withdraw the work, after notifying the author, in sufficiently justified cases, or in the event of third party claims.

Madrid, on ....15.... of .....July....., ..2019

HEREBY ACCEPTS

Signed..Alvaro....Gomez....Asensio

Reasons for requesting the restricted, closed or embargoed access to the work in the Institution's Repository



PONTIFICIAL UNIVERSITY OF COMILLAS
SUPERIOR TECHNICAL SCHOOL OF ENGINEERING (ICAI)
MASTER IN INDUSTRIAL ENGINEERING

MASTER THESIS

UNSUPERVISED LEARNING FOR CONDITION MONITORING

AUTHOR: Álvaro Gómez Asensio
DIRECTOR: Prof. Dr.-Ing. Michael F. Zäh

MADRID, July 2019

Resumen

0.1. Introducción

Las sociedades actuales dependen de sistemas estructurales y mecánicos, muchos de los cuales se acercan al final de su vida útil. Reemplazar estos sistemas puede ser costoso de reparar si ha ocurrido un fallo catastrófico o si se deben realizar inspecciones frecuentes. La monitorización del estado puede ayudar a reducir el coste de la inspección y el tiempo de inactividad de las máquinas, aumentando la productividad de los equipos de fabricación [4]. Una forma de hacer monitoreo de la condición es con enfoques basados en datos como el aprendizaje automático. En el aprendizaje automático, dependiendo de los datos que se posean, el aprendizaje puede ser supervisado o no. Aquí sólo vamos a estudiar el caso del aprendizaje no supervisado, donde los datos no tienen etiquetas. El aprendizaje no supervisado obtiene información de la estructura de los datos en lugar de las etiquetas [8]. Hasta ahora, ha habido muchos estudios y proyectos para aplicar el aprendizaje no supervisado a la monitorización de la condición pero ninguno ha utilizado métodos como UMAP para reducir las dimensiones antes de agrupar con HDBSCAN ([7],[1],[2]).

0.2. Objetivos

Esta tesis va a estudiar los efectos de nuevos métodos recientemente descubiertos en el monitoreo de condiciones con aprendizaje no supervisado. Estos incluyen el manifold learning y la agrupación basada en la densidad y la jerarquia como formas de agrupación de aprendizaje no supervisado.

Los objetivos de este proyecto son:

- Encontrar fallos de las máquinas en los datos mediante la aplicación de las técnicas más avanzadas de aprendizaje no supervisado.
- Investigar el manifold learning para la reducción de dimensiones y varias nuevas técnicas de agrupación como HDBSCAN con el fin de diferenciar entre los estados de fallo y las condiciones normales.
- Extraer conclusiones sobre la capacidad de los métodos aplicados para identificar las diferentes condiciones y fallos de la máquina en el conjunto de datos.

0.3. Estado del arte

0.3.1. Aprendizaje no supervisado

Las técnicas de aprendizaje sin supervisión son técnicas de aprendizaje automático utilizadas para encontrar patrones en los datos. Los datos dados al algoritmo no supervisado no están

etiquetados, lo que significa que sólo se dan las variables de entrada sin las variables de salida correspondientes. El aprendizaje sin supervisión se realiza a menudo como parte de un análisis exploratorio de datos. Tiende a ser más subjetivo y no hay un objetivo simple para el análisis, por ejemplo, la predicción de una respuesta. Además, puede ser difícil evaluar los resultados obtenidos de un aprendizaje no supervisado, ya que no existe un método universalmente aceptado para evaluar los resultados [3].

0.3.2. Reducción de dimensiones

0.3.2.1. Analisis de Componentes Principales (PCA)

El análisis de componentes principales se refiere al proceso mediante el cual se calculan los componentes principales y el uso de estos componentes en la comprensión de los datos. Esto se consigue definiendo un nuevo espacio de coordenadas (formado por componentes principales) para reexpresar el original. Una de las formas en que el análisis de componentes principales puede ser útil es reducir la dimensionalidad de un conjunto de datos compuesto por un gran número de variables interrelacionadas, manteniendo al mismo tiempo la mayor varianza posible.

0.3.2.2. Uniform Manifold Approximation and Projection (UMAP)

El manifold learning se concibe como una generalización de las transformaciones lineales (como el PCA) sensibles a la estructura no lineal de los datos. Aunque existen variantes supervisadas, la mayoría de los usos del manifold learning no están supervisados: aprende la estructura de alta dimensión de los datos a partir de los datos mismos, sin el uso de clasificaciones predeterminadas [5]. UMAP tiene una serie de hiperparámetros que pueden cambiar los resultados obtenidos con este método.

0.3.2.3. t-distributed Stochastic Neighbor Embedding (t-SNE)

t-SNE es muy similar a UMAP en el sentido de que ambos son métodos de reducción de dimensiones no lineales que pretenden terminar con 2 o 3 dimensiones finales, manteniendo al mismo tiempo la mayor parte posible de la estructura. t-SNE tiene como objetivo principal preservar la mayor parte posible de los vecindarios locales, mientras que UMAP trata de preservar también la estructura global.

0.3.3. Clusterización

0.3.3.1. K-Means

K-means divide el conjunto de datos en k clusters definidos por la media de cada cluster llamado centroide. Es un proceso iterativo donde el algoritmo trata de minimizar la inercia del cluster o error cuadrático medio.

0.3.3.2. Hierarchical Density Based Spatial Clustering of Applications with Noise (HDBSCAN)

HDBSCAN es un algoritmo de clusterización jerárquica que tiene en cuenta la densidad de los clusters. Tiene muchas ventajas sobre los métodos tradicionales de clustering, como ser más rápido que otros métodos basados en densidad como DBSCAN, usando parámetros intuitivos como el número mínimo de puntos de un cluster (sin requerir la selección del número de clusters).

0.3.4. Medición del rendimiento

0.3.4.1. Puntuación de precisión

La función de puntuación de precisión calcula la precisión, ya sea la fracción o el número total de predicciones correctas con las que un algoritmo ha predicho una salida. Si todo el conjunto de etiquetas previstas para un conjunto de datos coincide perfectamente con el verdadero conjunto de etiquetas, la puntuación de precisión es 1,0; de lo contrario, es 0,0 [6].

0.3.4.2. Análisis de silueta

El análisis de siluetas se refiere al método de interpretación y validación de la coherencia dentro de grupos de datos. El análisis de siluetas se utiliza para estudiar la distancia de separación entre los conglomerados resultantes. La puntuación de la silueta tiene un rango de[-1, 1] donde los coeficientes cerca de 1 indican que la muestra está lejos de los conglomerados vecinos. Un valor de 0 indica que la muestra está en el límite de decisión entre dos conglomerados vecinos y -1 indica que esas muestras podrían haber sido asignadas al conglomerado equivocado.

0.4. Resultados

Se utilizaron UMAP y HDBSCAN para encontrar fallos en máquinas de dos conjuntos de datos y los nuevos métodos se compararon con métodos más tradicionales de reducción de dimensiones y agrupamiento. Los dos conjuntos de datos utilizados fueron una simulación de un banco de pruebas hidráulico en el que las etiquetas se conocían pero no se utilizaban y un conjunto de datos de un escenario real (conjunto de datos de fresado) en el que las etiquetas no se conocían.

0.4.1. Resultados del método con el conjunto de datos simulado

Cuando se buscan daños en todos los componentes al mismo tiempo, el método muestra sus debilidades. UMAP y t-SNE pudieron separar en clusters algunos de los estados de los componentes. Por lo tanto, somos capaces de conocer el estado de algunos de los componentes, pero no de los otros. Comparando los resultados de t-SNE y UMAP para la reducción de dimensiones (ambos con HDBSCAN para clustering), las métricas objetivas mostraron que ambos tienen un rendimiento muy similar (ver Tabla 1).

HDBSCAN con:	Acc. Score	Ad. Rand Score	Mean silh. score
UMAP	0.834	0.707	0.741
t-SNE	0.821	0.707	0.649
K-means con:			
UMAP	0.836	0.709	0.754
t-SNE	0.834	0.707	0.665

Tabla 1. Comparación de resultados UMAP vs t-SNE

Respecto a la velocidad con la que se realizaron ambas reducciones, UMAP fue significativamente más rápido como muestra la Tabla 2, incluso con la paralelización de los cálculos (en 6 núcleos de una CPU).

Al comparar los métodos de agrupamiento, los datos muestran que la diferencia de rendimiento entre k-means y HDBSCAN es muy pequeña. El tiempo de cálculo de ambos

Método de reducción de dimensiones	Tiempo (s)
UMAP	25.5
t-SNE	150.1

Tabla 2. Velocidad de métodos de reducción de dimensiones

métodos de clustering es también muy similar con las implementaciones elegidas, HDBSCAN era más exigente con el ordenador pero no fue un factor decisivo con este conjunto de datos. La única diferencia significativa entre ambos métodos fue la facilidad de uso de ambos métodos. Mientras que k-means necesitaba el número de clusters esperado a priori, HDBSCAN necesitaba el tamaño mínimo de los clusters. Cuando se aplicaba el agrupamiento al aprendizaje no supervisado, era más fácil saber como de grande deberían ser los cluster que cuántos modos o tipos de fallo tenía que encontrar el algoritmo. Por lo tanto, HDBSCAN es un mejor método para la monitorización del estado sin supervisión en este caso.

0.4.2. Resultados del conjunto de datos del caso real

Para este conjunto de datos, el mejor método de reducción de dimensiones para separar los puntos en conglomerados para que estos últimos calculen las etiquetas con un método de conglomerados, fue UMAP. En este caso (contrariamente a lo que ocurrió en el conjunto de datos hidráulicos) UMAP y t-SNE dan resultados muy diferentes. Mientras que UMAP produce (en una de las incrustaciones) clusters muy densos y bien separados, t-SNE produce clusters que son menos densos y que son más difíciles de agrupar con un método de agrupación. En términos de velocidad, UMAP también fue más rápido que t-SNE (incluso con las optimizaciones de t-SNE y el procesamiento multinúcleo). La comparación entre ambas transformaciones de reducción de dimensiones se muestra en la Tabla 3.

Método de reudción de dimensiones	Tiempo (min)
UMAP	88
t-SNE	219

Tabla 3. Comparación de tiempo de calculo de los diferentes métodos de reducción de dimensiones

La tabla 4 muestra las diferentes métricas de rendimiento de los diferentes clústeres con los conjuntos de datos. En este caso, el método más fiable y fácil de usar debe ser el recomendado. Al agrupar la integración de este conjunto de datos, seleccionar el número de clústeres no es fácil y es más fácil seleccionar un número mínimo de puntos dentro de un clúster.

Métodos	Mean silhouette score
K-means con PCA	0.570
K-means con UMAP	0.604
K-means con t-SNE	0.368
HDBSCAN con PCA	0.401
HDBSCAN con UMAP	0.548
HDBSCAN con t-SNE	0.324

Tabla 4. Comparación de rendimiento entre los distintos métodos de cluterización.

Con este método es muy difícil decir si un cluster representa un fallo o no. Mientras que en el conjunto de datos del sistema hidráulico teníamos las etiquetas que nos decían si un punto tenía un fallo o no, aquí no hay suficiente información que indique si un punto es un fallo o no. Esto significa que este método no es suficiente para detectar y reaccionar ante fallos en las máquinas en una situación real de una producción.

0.5. Conclusiones

Tanto en la hidráulica como en los conjuntos de datos reales, UMAP fue el método superior de reducción de dimensiones. UMAP en comparación con PCA fue capaz de agrupar mejor los puntos para mejorar el rendimiento de la agrupación en clusters, también fue capaz de encontrar más clústeres en ambos conjuntos de datos, explicando mejor la estructura de todo el conjunto de datos a diferencia de PCA, donde se perdió mucha información en sus muchos componentes. UMAP y t-SNE crearon incrustaciones similares, pero UMAP tuvo tiempos de cálculo significativamente menores.

Al comparar los métodos de agrupamiento, HDBSCAN y k-means fueron muy similares en rendimiento. Ambos métodos ofrecieron resultados similares cuando se eligieron los parámetros correctos. La principal diferencia entre ambos métodos de clustering es que para que k-means funcione, el usuario tiene que conocer el número de clusters de antemano y esto no siempre es posible (especialmente en el caso de aprendizaje no supervisado). Incluso cuando un usuario sabe cómo funciona su máquina, puede no saber cuántos modos de fallo o actividades puede haber realizado la máquina durante la duración del conjunto de datos (lo que podría dar al usuario una pista sobre la cantidad de clústeres a seleccionar en k-means). Por lo tanto, en este caso HDBSCAN es el método superior para hacer clustering en el aprendizaje no supervisado.

Por un lado, el método propuesto mostró un buen desempeño con el conjunto de datos de la hidráulica porque los datos fueron adquiridos correctamente y sólo incluía características que afectaban a los componentes cuyo estado queríamos estudiar. Por otro lado, al final no pudimos decir con certeza, en base al cluster en el que se etiquetó un punto en qué estado se encontraban cada uno de los cuatro componentes. Con el conjunto de datos del caso real, el rendimiento no fue tan bueno como con el sistema hidráulico, ya que la calidad de los datos no era tan buena. Aunque el método era capaz de extraer información que antes no estaba disponible para el usuario, el estado de la máquina no estaba claro cuando se agrupó el conjunto de datos.

En conclusión, este trabajo ha demostrado que el método propuesto es mejor que otros métodos comúnmente utilizados para la agrupación no supervisada para la monitorización del estado con menores tiempos de cálculo y mayor calidad de agrupación. Sin embargo, la capacidad del método para discernir entre los estados de una máquina depende en gran medida de la calidad de los datos.

Bibliografía

[1] M. M. Alamdari y col. «A spectral-based clustering for structural health monitoring of the Sydney Harbour Bridge». En: Mechanical Systems and Signal Processing 87 (mar. de 2017), págs. 384-400. URL: http://www.sciencedirect.com/science/article/pii/S0888327016304538.

[2] N. Dervilis. «A machine learning approach to Structural Health Monitoring with a view towards wind turbines». Tesis doct. University of Shefield, 2013.

[3] G. James y col. An Introduction to Statistical Learning: With Applications in R. Springer texts in statistics 103. Springer Publishing Company, Incorporated, 2014. 426 págs. ISBN: 978-1-4614-7137-0.

[4] N. L. D. Khoa y col. «Structural Health Monitoring Using Machine Learning Techniques and Domain Knowledge Based Features». En: Human and Machine Learning: Visible, Explainable, Trustworthy and Transparent. Ed. por J. Zhou y F. Chen. Cham: Springer International Publishing, 2018, págs. 409-435. ISBN: 978-3-319-90403-0. URL: https://doi.org/10.1007/978-3-319-90403-0_20.

[5] F. Pedregosa y col. Manifold learning. 2011. URL: https://scikit-learn.org/stable/modules/manifold.html.

[6] F. Pedregosa y col. Model evaluation: quantifying the quality of predictions. 2011. URL: https://scikit-learn.org/stable/modules/model_evaluation.html.

[7] G. Singh y col. «Machine Learning-Based Framework for Multi-Class Diagnosis of Neurodegenerative Diseases: A Study on Parkinson's Disease». En: IFAC-PapersOnLine 49.7 (2016). 11th IFAC Symposium on Dynamics and Control of Process SystemsIncluding Biosystems DYCOPS-CAB 2016, págs. 990-995. URL: http://www.sciencedirect.com/science/article/pii/S2405896316305389.

[8] K. Smarsly y col. «Machine learning techniques for structural health monitoring». En: jul. de 2016.

Abstract

0.1 Introduction

Today's societies rely on structural and mechanical systems, many of which are nearing the end of their designed life. Replacing these systems can be expensive to repair if a catastrophic failure has occurred or if frequent inspections have to be done. Condition monitoring can help reduce the cost of inspection and down-time of machines , increasing the productivity of the manufacturing equipment [4]. One way of doing condition monitoring is with data based approaches like machine learning. In machine learning, depending on the amount of data possessed, the learning can be supervised or unsupervised. Here we are just going to study the case of unsupervised learning, where the data points don't have labels. Unsupervised learning gets information from the structure of the data rather than from the labels [8]. Until now, there have been many studies and projects for applying unsupervised learning and supervised learning to condition monitoring. None have used methods like Uniform Manifold Approximation and Projection for reducing dimensions before clustering with Hierarchical Density Based Spatial Clustering of Application with Noise ([7],[1],[2]).

0.2 Objectives

This thesis is going to study the effects of new approaches recently discovered in CM with unsupervised learning. These include manifold learning and hierarchical density based clustering as ways of unsupervised learning clustering.

The objectives of this project are:

- Find machine failures in raw data by applying state of the art techniques of unsupervised machine learning.
- Investigate manifold learning for dimension reduction and various new cluster techniques like HDBSCAN in order to differentiate between the states of damage and normal condition.
- Draw conclusions on the ability of the applied methods to identify different machine conditions and failures in the data set.

0.3 State of the Science and Technology

0.3.1 Unsupervised Learning

Unsupervised learning techniques are machine learning techniques used to find patterns in the data. The data given to the unsupervised algorithm is not labelled, meaning that only the input

variables are given without corresponding output variables. Unsupervised learning is often performed as part of an exploratory data analysis. It tends to be more subjective and there is no simple goal for the analysis, for example, prediction of a response. Furthermore, it can be hard to assess the results obtained from unsupervised learning, as there is no universally accepted method for assessing the results [3].

0.3.2 Dimension reduction

0.3.2.1 Principal Component Analysis (PCA)

Principal component analysis refers to the process by which principal components are computed and the use of these components in the understanding of the data. This is achieved by defining a new coordinate space (made up of principal components) to re-express the original one. One of the ways in which principal component analysis can be useful is to reduce the dimensionality of a dataset made of a large number of interrelated variables, while also maintaining as much variance as possible.

0.3.2.2 Uniform Manifold Approximation and Projection (UMAP)

Manifold learning is thought of as a generalization of linear transformations (like PCA) sensitive to non-linear structure in data. Though supervised variants exist, most manifold learning uses are unsupervised: it learns the high dimensional structure of the data from the data itself, without the use of predetermined classifications [5]. UMAP has a number of hyperparameters that can change the results achieved with this method.

0.3.2.3 t-distributed Stochastic Neighbor Embedding (t-SNE)

t-SNE is very similar to UMAP in that they are both non-linear dimension reduction methods that aim to end up with 2 or 3 final dimensions while also maintaining as much of the structure as possible. t-SNE main objective is to preserve as much of the local neighborhoods as possible, while UMAP tries to preserve global structure as well.

0.3.3 Clustering

0.3.3.1 K-Means

K-means divides the data set into k clusters defined by the mean of each cluster called centroid. It is an iterative process where the algorithm tries to minimize the inertia of the cluster or the sum of squared criterion.

0.3.3.2 Hierarchical Density Based Spatial Clustering of Applications with Noise (HDBSCAN)

HDBSCAN is a hierarchical clustering algortithm that takes into account the density of the clusters. It has many advantages over traditional clustering methods, like being faster than other density based methods like DBSCAN, using intuitive parameters like the minimum number of points of a cluster (not requiring the selection of the number of clusters).

0.3.4 Performance metrics

0.3.4.1 Accuracy Score

The accuracy score function computes the precision, either the fraction or the total number of correct predictions with which an algorithm has predicted an output. If the whole set of predicted labels for a data set perfectly match with the true set of labels, then the accuracy score is 1.0; otherwise it is 0.0 [6]

0.3.4.2 Silhouette Score

Silhouette analysis refers to the method of interpretation and validation of consistency within clusters of data. Silhouette analysis is used to study the separation distance between the resulting clusters. The silhouette score has a range of [-1, 1] where coefficients near 1 indicate that the sample is far away from the neighboring clusters. A value of 0 indicates that the sample is on the decision boundary between two neighboring clusters and -1 indicate that those samples might have been assigned to the wrong cluster.

0.4 Results

UMAP and HDBSCAN were used to find failures in machines from two data sets and the methods were compared with more traditional dimension reduction and clustering methods. The two data sets used were a simulation of a hydraulic test rig where the labels were known but not used and a data from a real scenario (milling data set) where the labels were not known.

0.4.1 Results from the hydraulic data set

When looking for damage in all components at the same time, the method shows its weaknesses. UMAP and t-SNE were able to separate in clusters some of the states of the components. Therefore, we may be able to know at any given point the state of some of the components but not the others. Comparing the results of t-SNE and UMAP for the dimension reduction (both with HDBSCAN for clustering), the objective metrics showed that both have very similar performance (see Table 1).

HDBSCAN with:	Acc. Score	Ad. Rand Score	Mean silh. score
UMAP	0.834	0.707	0.741
t-SNE	0.821	0.707	0.649
K-means with:			
UMAP	0.836	0.709	0.754
t-SNE	0.834	0.707	0.665

Table 1. Result comparison UMAP vs t-SNE

Regarding the speed with which both embeddings were made, UMAP was significantly faster as Table 2 shows, even with the parallelization of the computations (in 6 cores of a CPU).

When comparing clustering methods, the data shows that the difference in performance between k-means and HDBSCAN is very small. For example, when comparing UMAP clustered by HDBSCAN vs k-means there is only a difference of 0.002 on the adjusted Rand score. The computing time of both clustering methods is also very similar with the chosen implementations, HDBSCAN was more computer demanding but it was not a deciding factor with this data set.

Dimension reduction method	Compute time (s)
UMAP	25.5
t-SNE	150.1

Table 2. Speed of dimensionality reduction methods

The only meaningful difference between both methods was the ease of use of both methods. While k-means needed the number of expected clusters a priori, HDBSCAN needed the minimum size of the clusters. When applying clustering to unsupervised learning, it was easier to know how big the failure cluster should be than how many failure modes or clusters it had to find. Therefore, HDBSCAN is a better method for unsupervised condition monitoring.

0.4.2 Results from the milling data set

For this data set, the best dimension reduction method for separating the points into clusters for latter computing the labels with one clustering method, was UMAP. In this case (contrary to what happened on the hydraulic data set) UMAP and t-SNE give very different results. While UMAP produces (in one of the embeddings) very dense and well separated clusters, t-SNE produces clusters that are less dense and that are more difficult to cluster with a clustering method. In terms of speed, UMAP was also faster than t-SNE (even with the t-SNE optimizations and the multicore processing). The comparison between both dimension reduction transformations is shown in Table 3.

Dimension reduction method	Compute time (min)
UMAP	88
t-SNE	219

Table 3. Comparison of the compute time of the different dimension reduction methods

Table 4 shows the different performance metrics of the different clusterings with the embeddings. UMAP has the best result in both K-means and HDBSCAN clustering when compared to the rest of the dimension reduction methods. K-means with UMAP gets a better result than HDBSCAN with UMAP but only by 0.05 points. In this case, the method that is the most reliable and most easy to use should be the one that is recommended. When clustering the embedding of this data set selecting the number of clusters is not easy and it is much easier to select a minimum number of points inside a cluster.

Methods	Mean silhouette score
K-means with PCA	0.570
K-means with UMAP	0.604
K-means with t-SNE	0.368
HDBSCAN with PCA	0.401
HDBSCAN with UMAP	0.548
HDBSCAN with t-SNE	0.324

Table 4. Comparison of performance between the different clustering methods

With this method it is very difficult to say whether a cluster represents a failure or not. While on the data set of the hydraulic system we had the labels that told us whether a point had a failure or not, here there is not enough information that would indicate if a point is a failure or not. This

means that this method is not enough to detect and react to failures in machines on a real life situation of a production.

0.5 Conclusions

This thesis has proved that unsupervised learning is a useful method to extract information from data of which we knew nothing or little to nothing. It can also help detect failures in the system when the quality of the data is sufficient. However, it can not be seen as the only mechanism to detect failures in data with no labels. Even with high quality data, the performance of the method is not high enough to rely solely on it.

In both the hydraulics and the real data sets, UMAP was the superior dimensionality reduction method. UMAP compared to PCA was able to better group together points in order to improve the clustering performance, it was also able to find more clusters in both data sets, better explaining the structure of the whole data set unlike PCA where much information was lost in its many components. UMAP and t-SNE created similar embeddings but UMAP had significantly lower computing times. This affects the user's ability to apply this method with less powerful hardware and makes reaching a decision in the maintenance much quicker. In addition, as the data set increases its size, UMAP is a better option as t-SNE will stop being a feasible solution due to the big increase in computing time relative to the size of the data set.

When comparing clustering methods, HDBSCAN and k-means were very similar in performance. Both methods offered similar results when the correct parameters were chosen. The main difference between both clustering methods is that in order to make k-means work, the user has to know the number of clusters beforehand and this is not always possible (specially in the unsupervised learning case). Even when a user knows how its machine works, it may not know how many failure modes or activities the machine might have done during the length of the data set (which could give the user a clue about the amount of clusters to select in k-means). HDBSCAN is also a great tool to get more information about the data set it is working with. Condensed tree dendograms and the ability to include new data points into the clustering (after having fitted the model) are additional resources the user might use that were not available with k-means. Therefore, in this case HDBSCAN is the superior method for doing clustering in unsupervised learning.

In one hand, the proposed method showed good performance with the hydraulics data set because the data was correctly acquired and only included features that affected the components whose state we wanted to study. On the other hand, in the end we could not say with certainty, based on the cluster a point was labeled into, which state each of the 4 components was in. With the real-world data set, the performance was not as good as with the hydraulics as the data quality was not as good. Even though the method was able to extract information not available before to the user, the state of the machine was not clear when clustering the data set.

In conclusion, this thesis has proved that the proposed method is better than other commonly used methods for unsupervised clustering for condition monitoring with lower computing times and higher clustering quality. However, the method's ability to discern between the states of a machine is highly dependant on the data quality.

Bibliography

[1] M. M. Alamdari et al. “A spectral-based clustering for structural health monitoring of the Sydney Harbour Bridge”. In: Mechanical Systems and Signal Processing 87 (Mar. 2017), pp. 384–400. URL: http://www.sciencedirect.com/science/article/pii/S0888327016304538.

[2] N. Dervilis. “A machine learning approach to Structural Health Monitoring with a view towards wind turbines”. PhD thesis. University of Shefield, 2013.

[3] G. James et al. An Introduction to Statistical Learning: With Applications in R. Springer texts in statistics 103. Springer Publishing Company, Incorporated, 2014. 426 pp. ISBN: 978-1-4614-7137-0.

[4] N. L. D. Khoa et al. “Structural Health Monitoring Using Machine Learning Techniques and Domain Knowledge Based Features”. In: Human and Machine Learning: Visible, Explainable, Trustworthy and Transparent. Ed. by J. Zhou and F. Chen. Cham: Springer International Publishing, 2018, pp. 409–435. ISBN: 978-3-319-90403-0. URL: https://doi.org/10.1007/978-3-319-90403-0_20.

[5] F. Pedregosa et al. Manifold learning. 2011. URL: https://scikit-learn.org/stable/modules/manifold.html.

[6] F. Pedregosa et al. Model evaluation: quantifying the quality of predictions. 2011. URL: https://scikit-learn.org/stable/modules/model_evaluation.html.

[7] G. Singh et al. “Machine Learning-Based Framework for Multi-Class Diagnosis of Neurodegenerative Diseases: A Study on Parkinson’s Disease”. In: IFAC-PapersOnLine 49.7 (2016). 11th IFAC Symposium on Dynamics and Control of Process SystemsIncluding Biosystems DYCOPS-CAB 2016, pp. 990–995. URL: http://www.sciencedirect.com/science/article/pii/S2405896316305389.

[8] K. Smarsly et al. “Machine learning techniques for structural health monitoring”. In: July 2016.

DOCUMENTO I

MEMORIA

Table of Contents

List of Figures

List of Tables

Acronyms

DBSCAN	Density Based Spatial Clustering of Applications with Noise
HDBSCAN	Hierarchical Density Based Spatial Clustering of Applications with Noise
PCA	Principal Component Analysis
PC	Principal Components
CM	Condition Monitoring
t-SNE	t-distributed Stochastic Neighborhood Embedding
UMAP	Uniform Manifold Approximation and Projection
ARI	Adjusted Rand Index
SC	Silhouette Analysis
AS	Accuracy Score
SSE	Sum of Squared Errors
GMM	Gaussian Mixture Models
SVDD	Support Vector Data Description

Symbols

KL	Kullback-Leiber
w_h	High dimension weight
w_l	Low dimension weight
$s(x_i, x_j)$	Similarity

PART I

REPORT

Chapter 1

Introduction

Today's societies rely on mechanical and structural systems, many of which are nearing the end of their designed life. Replacing these systems with new ones is not always economically viable, therefore the logical solution is to maintain and repair them when damage is detected and prevent failures. Most structural and mechanical maintenance systems are time-based, i.e. an inspection is carried out after a predefined amount of time [18]. In contrast, Condition Monitoring (CM) is a condition-based maintenance system, i.e. an inspection is carried out after a certain condition is detected. Condition Monitoring is the process of implementing a damage detection and characterization strategy for manufacturing equipment. A sensing system monitors the system's response and will notify the operator of damage. CM methods can be divided into physics-based (f.e. by using finite element analysis) and data-based. This thesis focuses on the data driven approaches of CM, where the sensor data allows learning patterns without any knowledge of the physical characteristics of the machine [43].

Data-driven methods offer new solutions for condition monitoring of system failures. This has made data analytics within diagnostic technologies a high-priority research topic. As many industries continuously work to improve their performance (by delivering more reliable products with a higher availability), operational pressures expect to reduce the time required for diagnostics. Here, there is value of having many data collection sources that can be used to provide information if a failure occurs during operation. However, data sources are often scarce. In this context, new approaches are required to make better maintenance decisions. In the normal environment, these problems require advanced capabilities to monitor operations, record and share expert knowledge. To address this issue, diagnostic systems based on conventional techniques are being replaced by AI-based ones like machine learning which can increase the efficiency of the monitoring technology [2].

Machine learning is a field of artificial intelligence that uses statistical techniques to give computer systems the ability to learn from data, without being programmed. In CM there are 2 ways to approach machine learning: supervised and unsupervised. In supervised machine learning, the labels for the classes are known (failure data is known), while in the unsupervised learning case failure information has to be taken out from the intrinsic relationships within the data.

Often, failure data is either rare or not recorded properly, which limits the use of supervised machine learning techniques in CM. This can be addressed with the help of unsupervised learning techniques [45]. In this thesis we are going to focus on clustering. Clustering is the task of grouping together a set of objects where some objects are more similar to each other than those in other groups. This can help in CM by differentiating between data points where there was damage and where the system was working correctly. This method assumes that collected data shows a difference between the states of damage and no-damage. There are many types

of clustering, but the most common is K-means, which has many limitations. Recently, the HDBSCAN method was developed to overcome many of these limitations.

Before clustering, it is recommended to reduce the number of features or dimensions of the data set. This is usually done with a principal component analysis. This method is very effective on linear data sets, but many data sets are non-linear and need a new way of reducing dimensions. UMAP is a non-linear method of dimension reduction that can produce 2-dimensional representations of the data set taking into account both the local and the global distances between the points in the data set. By combining these two methods, we can create a unsupervised learning method for the condition monitoring that can yield accurate and reliable results.

1.1 Motivation

Modern condition monitoring systems exist to offer basic monitoring. Even though there have been advancements in the field, there have also been technological limitations for real-time decision-making. Modern systems are reliable and safe but they are becoming more expensive to operate and maintain. There will always be a financial trade-off while carrying out these activities, and this will be reflected in the resources needed to support maintenance. In recent years, globalization and technological innovation have changed the way a system is maintained. Techniques such as condition monitoring offer maintenance staff early indications of both impending and actual failures. No matter how well a maintenance system is designed, there are always deficiencies due to trade-offs in design that can lead to a reduction in the quality of service. This is because there are always inherent weaknesses in systems which only become evident once the system is in operation [17].

The Society of Automotive Engineers states there are many economic motivations for the use of CM systems such as reduction in inspection time and cost, repair planning improvement, economic life extension and optimization of inspection intervals [6]. Additionally, there is a lower probability of catastrophic failure and subsequent litigation. Manufacturers are constantly seeking to improve the productivity, cost and quality of their manufacturing. For example, the reduction of downtime is important for industrial equipment and machinery to drive up productivity and overall operational equipment efficiency. Production efficiency and productivity can be improved by maximizing the time that machines are up and running through predictive maintenance and CM. Another example is how we could predict the condition of a machine based on the defects it creates on its outputs (in this case the CM directionality is reversed) [40]

A key aspect of implementing CM is to include next-generation digital core technologies like machine learning and advanced analytics. These tools must be flexible, scalable and easily embeddable into legacy IT infrastructure. Achieving near zero downtime, while at the same time integrating more sophisticated health monitoring systems into existing designs is a big challenge with high cost and long design periods. Advanced analytics enable organizations to derive new insights, such as predicting outcomes or proposing new actions. The machine learning system is the brain of a condition monitoring platform, its models are designed to detect anomalous behaviour during production. Training data helps develop specific models for specific equipment types. Timely detections and prescriptions are accomplished by determining whether a data point falls outside the normal behaviour bounds [7]. Traditionally, systems had been developed as model-based systems which could consider different fault conditions. These can simulate a wide range of operating conditions but their implementation within a single application often lead to complexities difficult to maintain and manage. In 1991, Merrill and Lorenzo [20] studied the need of AI for diagnostic tools. In this study, they imagined that expert-based decision making

in real-time will eventually be real once technological advancements can improve computing times and implementation architectures.

The rise in interest in developing AI solutions is a consequence of improving processor performance and falling cost [26]. Any diagnostic analysis requires a rigorous procedure to obtain reliable system models. There is a dichotomy in the use of AI for condition monitoring. Improved capability is achieved through added complexity, which reduces the system's reliability. However, AI based diagnostics have many benefits when compared to conventional approaches.

In the past few decades, many AI approaches were integrated into condition monitoring systems, but emphasis was placed on mathematical rigour. [19]. As research advanced, many other techniques were incorporated, replacing conventional techniques. However, these mathematical models are heavily dependent on the quality of the captured data. This led to the rise of new systems based on other methods, like fuzzy logic and Bayesian networks, to overcome the limitations of heuristic approaches in the old models [3].

Capability to train a new model is key. CM works best when accurate information about the state of the system is available. Otherwise, if this information is not available or it is not possible to get, the only other solution is to infer this information from the data set. Inferring information from the data set is called unsupervised learning. Some examples of unsupervised learning methods are: clustering, anomaly detection and neural networks. These algorithms (especially neural networks) need computing power and time. Computing time is directly proportional to the number of features, so reducing them improves the computing time. The number of dimensions of a data set is equal to the number of features it has. One important dimension reduction technique is Principal Component Analysis. PCA can give a better way to visualize a high dimensional data set, by agglomerating the variance of many variables into a few and therefore being able to represent the data in two or three dimensions. PCA can be unhelpful due to the linear nature of its algorithm. One way to solve this problem is through the use of non-linear dimension reduction like manifold learning. Manifold learning is able to adapt to different kinds of data better than a linear approach like PCA [25]. Traditionally, clustering has been done with simple algorithms that make strong assumptions. For example, the K-means method assumes clusters are spherical and similar in size (number of points inside) without imposing any minimum density of points inside the cluster. Hierarchical Density Based Spatial Clustering of Applications with Noise is a way of overcoming these limitations and creating a powerful tool for finding clusters in the data.

Unsupervised learning has been done before. A list of examples where unsupervised learning was applied is as follows:

- Diagnosing neuro-degenerative diseases [42].
- Sensing damage in large structures like bridges [1].
- Sensing damage in wind turbines [8].

1.2 Objectives

This thesis is going to study the effects of new approaches recently discovered in CM with unsupervised learning. These include manifold learning and hierarchical density based clustering as ways of unsupervised learning clustering.

The objectives of this project are:

- Find machine failures in raw data by applying state of the art techniques of unsupervised machine learning.

- Investigate manifold learning for dimension reduction and various new cluster techniques like HDBSCAN in order to differentiate between the states of damage and normal condition.
- Draw conclusions on the ability of the applied methods to identify different machine conditions and failures in the data set.

Chapter 2

State of the Science and Technology

2.1 Unsupervised machine learning

Machine learning is a method of data analysis that automates analytical model building. It is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention [39]. Because of new computing technologies, machine learning today is not like machine learning of the past. It was born from pattern recognition and the theory that computers can learn without being programmed to perform specific tasks. The iterative aspect of machine learning is important because as models are exposed to new data, they are able to independently adapt. They learn from previous computations to produce reliable, repeatable decisions and results.

Two of the most widely adopted machine learning methods are supervised learning and unsupervised learning. Supervised learning algorithms are trained using labeled examples (an input where the desired output is known). For example, a piece of equipment could have data points labeled either F (failed) or R (runs). The learning algorithm receives a set of inputs along with the corresponding correct outputs and the algorithm learns by comparing its actual output with correct outputs to find errors. It then modifies the model accordingly. Through methods like classification, regression, prediction and gradient boosting, supervised learning uses patterns to predict the values of the label on additional unlabeled data.

There is a lack of both benchmarking and large databases in fault diagnosis. The collaboration of industry and researchers is essential to create large, publicly available databases of machines working in real scenarios. The access to real data in the industrial environment for research purposes benefits both sides. Public databases provide common data to comparatively evaluate fault diagnosis techniques and CM systems [13].

Unsupervised learning is used against data that has no historical labels. Unsupervised learning techniques are machine learning techniques used to find patterns in the data. The data given to the unsupervised algorithm is not labelled, meaning that only the input variables are given without corresponding output variables. Unsupervised learning is often performed as part of an exploratory data analysis. It tends to be more subjective and there is no simple goal for the analysis, for example, prediction of a response. Furthermore, it can be hard to assess the results obtained from unsupervised learning, as there is no universally accepted method for assessing the results [14]. Techniques for unsupervised learning are becoming more important in a number of areas. Tasks like cancer research, search engine optimization or identifying groups of buyers can be performed with unsupervised learning techniques.

Some applications of unsupervised machine learning are [37]:

- Clustering: automatically split a dataset into groups according to differences in similarity.

- Anomaly detection: automatically discover unusual data points in a dataset. This can be useful in spotting fraudulent transactions, discovering faulty pieces of hardware or identifying an outlier data point caused by a human error during data entry.
- Association mining: identify sets of items that frequently occur together in your dataset. Often used in retail for basket analysis allowing the analyst to discover goods frequently purchased together and develop more effective marketing and merchandising strategies.
- Latent variable models: used for data preprocessing, such as reducing the number of features in a dataset (dimensionality reduction) or decomposing the dataset into multiple components.

The use of a common framework of performance metrics is also necessary for this task. Although researchers and industry have defined performance metrics, their application in scientific research is still poor. In case of unavailable fault data, unsupervised classification methods are a good alternative. Further research is needed to model data from normal conditions, taking into account different conditions, noisy samples, and environmental changes. The improvement of unsupervised modeling, using GMMs or SVDD, and the application of the universal background model in fault diagnosis, where much data from normal conditions is available, are promising research lines.

2.2 Preprocessing

Sometimes, the data used by machine learning algorithms is not optimized for them [41]. In order to optimize the performance of machine learning algorithms, the data has to be preprocessed.

2.2.1 Imputation

Imputation is the process of substituting the missing values of our data set for new values like the mean or 0. This is useful when the data has some missing values due to errors in the machine or the sensor. This might cause problems on the machine learning algorithm, so removing them is crucial. There are various ways to remove missing values for whatever value the user wants automatically. For example, the user can choose to fill the gaps with the mean or the value of the previous position in the array (this method is called forward-fill). [4]

2.2.2 Scaling

Having the same scale for every feature is important for the machine learning algorithms as they will normally assign more importance to the feature with the most variance or scale, even though really the difference between the features may be due to something as arbitrary as the units they are represented in [31]. In this project only two scaling mechanisms will be discussed:

- Standardization: subtracts the feature's mean to every value and then divides each value by the feature's variance. This way the resulting feature will have a mean of 0 and a variance of 1. This is useful when there are features with very different scales, but no real meaning.
- Standardization with the median: When the data set has many outlier, its mean can shift towards the outliers. This method uses the feature's median and its inter-quantile ranges (IQR) instead of its mean and variance in order to reduce the effect of outliers in the rest of the data.

2.2.3 Dealing with categorical features

Attributes aren't always represented by numbers or continuous. For example, when we have the city of birth as a feature, its values are strings containing the name of the city. These features are called categorical features and some algorithms like PCA need every value to be numerical in order to calculate the euclidean distance between them. An example of these kind of attributes is shown in table 1

City	Income	Gender	Age
New York City	112	M	28
Paris	98	F	34
Madrid	64	M	42

Table 1. An example data set with categorical features.

One way to deal with categorical features is to change the strings with numbers, but we do not want the algorithm to assume that, for example, 2 is bigger than 1. A way to solve this problem is to create new variables that add one column per unique value of the categorical feature. The result of this change is shown in Table 2

New York City	Paris	Madrid	Income	Male	Female	Age
1	0	0	112	1	0	28
0	1	0	98	0	1	34
0	0	1	64	1	0	42

Table 2. An example data set with converted categorical features.

The result of encoding categorical features into binary numerical features means that the distance metric used to calculate the distance between the points has to change.

2.2.3.1 Distance metric for binary attributes

If the binary attribute is symmetric (both of its states are equally valuable) a matching coefficient assesses the dissimilarity between the two values:

$$d(x_i, x_j) = \frac{r + s}{q + r + s + t} \tag{1}$$

where x_i and x_j are both objects, q is the number of attributes that equal 1 for both objects, t is the number of attributes that equal 0 for both objects and s and r are the number of attributes that are unequal for both objects [38].

2.2.3.2 Distance metric for mixed-type attributes

In the cases where the instances are characterized by attributes of mixed-type (binary, categorical and continuous), one may calculate the distance by combining the distance between the binary and the continuous features. For instance, when calculating the distance between instances i and j, one may calculate the difference between nominal and binary attributes as 0 or 1 (match or mismatch, respectively), and the difference between continuous attributes as the difference between their normalized values. The square of each such difference will be added to the total

distance [38]. The dissimilarity $d(x_i, x_j)$ between two instances, containing p attributes of mixed types, is defined as:

$$d(x_i, x_j) = \frac{\sum_{n=1}^{p} \delta_{ij}^{(n)} d_{ij}^{(n)}}{\sum_{n=1}^{p} \delta_{ij}^{(n)}} \tag{2}$$

where the indicator $\delta(n) = 0$ if one of the values is missing. The contribution of attribute n to the distance between the two objects $d^{(n)}(x_i, x_j)$ is computed according to its type:

- If the attribute is binary, $d^{(n)}(x_i, x_j) = 0$ if $x_{in} = x_{jn}$, otherwise $d^{(n)}(x_i, x_j) = 1$.
- If the attribute is continuous, $d_{ij}^{(n)} = \frac{|x_{in} - x_{jn}|}{max_h x_{hn} - min_h x_{hn}}$, where h runs over all non-missing points for attribute n.
- If the attribute is ordinal, the standardized values of the attribute are computed first and then $z_{i,n}$ is treated as continuous-values.

This distance is sometimes called the Gower distance because of the research done by J. C. Gower [11].

2.3 Dimension reduction

Dimension reduction is important in data science for visualization and a potential preprocessing step for machine learning [25]. Dimension reduction seeks to produce a low dimensional representation of a high dimensional data set that preserves its relevant structure. Dimension reduction methods tend to fall into two categories: those that seek to preserve the distance structure within the data and those that favor the preservation of local distances over global distance. Methods such as Principal Component Analysis fall into the former category while manifold learning falls into the latter category.

Methods can also be categorized into linear and non-linear. Linear methods are the most common like PCA. These tend to focus in keeping the low-dimensional representation of dissimilar points far apart. For high-dimensional data sets that lie on or near a low-dimensional, non-linear manifold it is usually important to keep the low-dimensional representations of very similar data points close together, which is typically not possible with a linear dimension reduction method.

2.3.1 Linear: Principal Component Analysis (PCA)

Principal component analysis refers to the process by which principal components are computed and the use of these components in the understanding of the data. This is achieved by defining a new coordinate space (made up of principal components) to re-express the original one. The main objective of PCA is to distinguish which dynamics are more relevant in the system and which are redundant [34]. One of the ways in which principal component analysis can be useful is to reduce the dimensionality of a dataset made of a large number of interrelated variables, while also maintaining as much variance as possible. This is achieved by transforming to a new set of variables, the principal components, which are uncorrelated by principle and are ordered by the amount of variation they express. With PCA, the user can decide how much variation one is willing to maintain in exchange for the reduction in dimensions. [16]

Calculating the principal components is simple. Consider, the covariance matrix Σ is known, the kth PC is given by $z_k = \alpha_k^{'} x$ where α_k is an eigenvector of Σ corresponding to its kth largest eigenvalue λ_k and x is the original dataset matrix. If α_k has unit length, then $var(z_k) = \lambda_k$, where

$var(z_k)$ denotes the variance of z_k. Meaning, the eigenvector of the first principal component is the direction with the most variance. Then, the second is perpendicular to the first, with the second most variance explained by it, and so on. This can be easily represented on a 2D-plane, like on Figure 1.

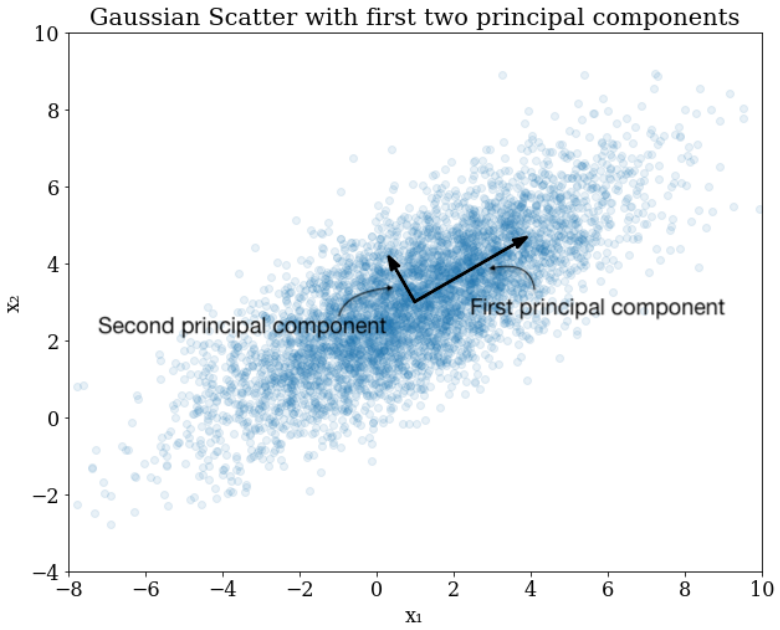


Figure 1. Example of the first two principal components on a 2D plane

Under the assumption that $x = s + n$ with s being the information bearing signal and n being a noise signal, PCA will minimize the information loss associated with the dimensionality reduction when s and n are gaussian-distributed.

Two-dimensional plots are useful for detecting patterns in the data. If the data doesn't lie close to a two dimensional space, the representation won't represent adequately the data. On the contrary, if the data is close to a 2-dimensional space, then most of the variation in the data will be represented by the first two Principal Components and a plot will give a realistic representation of what the data looks like, unless important parts of the data structure are in the direction of low variance PCs. It should also be noted that the range of structures that may be revealed by PCs is limited by the fact that the PCs are uncorrelated. Some types of outlier patterns or non-linear relationships may not appear in the PCs [15].

2.3.2 Non-linear: Uniform Manifold Approximation and Projection (UMAP)

Methods like PCA create a linear projection of the data with no input from the user (there is no need to select parameters depending on the data set it is transforming). This means that for every data set transformed, the PCA representation will always be its best linear projection. This method can be powerful, but often misses important non-linear structure in the data.

Manifold learning is thought of as a generalization of linear transformations (like PCA) sensitive to non-linear structure in data. Though supervised variants exist, most manifold learning uses are unsupervised: it learns the high dimensional structure of the data from the data itself, without the use of predetermined classifications [29]. In this thesis two specific implementations of manifold learning are described, Uniform Manifold Approximation and Projection [25] and t-distributed Stochastic Neighborhood Embedding. The theory for UMAP

is largely based on manifold theory and topological data analysis. This thesis's objective is not to explain in detail how the method works as it requires complex mathematics but if the reader is interested it is recommended to read McInnes' paper on the matter [25]. UMAP is a k-nearest neighbor graph (k-NNG) based method. A nearest neighbor graph is a directed graph in which two vertices are connected by an edge if the distance between the points is among the k-th smallest distances from the point to other objects. As with other k-neighbor graph based algorithms, UMAP is described in two phases. In the first phase, a weighted k-neighbour graph is constructed. In the second phase a low dimensional layout of this graph is created and optimized. The variable to optimize when making the low dimensional representation is the cross entropy. The cross entropy is:

$$\sum_{e \in E} w_h(e) log\left(\frac{w_h(e)}{w_l(e)}\right) + (1 - w_h(e)) log\left(\frac{1 - w_h(e)}{1 - w_l(e)}\right) \tag{3}$$

where w_h is the weight associated to the high dimension case, w_l is the weight associated to the low dimensional case and E is the set of all possible distances between the points (also called 1-simplices) where $e \in E$. The first element of equation 3 provides an attractive force between points when there is a large weight associated to the high dimension case. This term is minimized when w_l is as large as possible. The second term of equation 3 provides a repulsive force between the ends of e whenever w_h is small. This is minimized when w_l is small. The UMAP method takes a few shortcuts to make it less computationally intensive. When constructing the low dimensional representation, only the cross entropy for the nearest neighbor of each point is computed. This is done by using the Nearest Neighbor Descent algorithm of *Dong et al*. When optimizing the low dimensional representation we can also take some shortcuts, for example, using the stochastic gradient descent [33].

UMAP has a number of hyperparameters that can change the results achieved with this method.

- n, the number of neighbors to consider when approximating the local metric. It is used when forming the k-NN graph.
- d, the target embedding dimension. Usually chosen to be 2 for better visualization, but it can also be any other number as sometimes representing the data is not the main objective. Sometimes for clustering it can be a good idea.
- min-dist, the desired separation between close points in the embedding space. It provides the minimum distance apart that points are allowed to be in the low dimensional representation. This can be useful if you are interested in clustering, or in finer topological structure.
- n-epochs, the number of training iterations when optimizing the low dimensional representation.

The effects of changing d and n-epochs are very clear, but the effect of changing n and min-dist has to be discussed in more detail. The number of neighbors n represents a trade-off between fine grained and large scale manifold features. In contrast, min-dist determines how closely points are packed together in the low dimensional representation. Low values of min-dist will result in potentially densely packed regions, but will more faithfully represent the manifold structure. The min-dist parameter is also an aesthetic parameter, governing the appearance of the embedding, and thus more important when using UMAP for visualization.

An example of how changing the parameters n and min-dist can affect the embedding are shown in Figure 2. The different colors for the points in each of the plots represent a label and

show how the parameters affect the grouping of points. When the points belonging to one label (or color) are closer together it is beneficial because the clustering method will cluster more easily these points together.

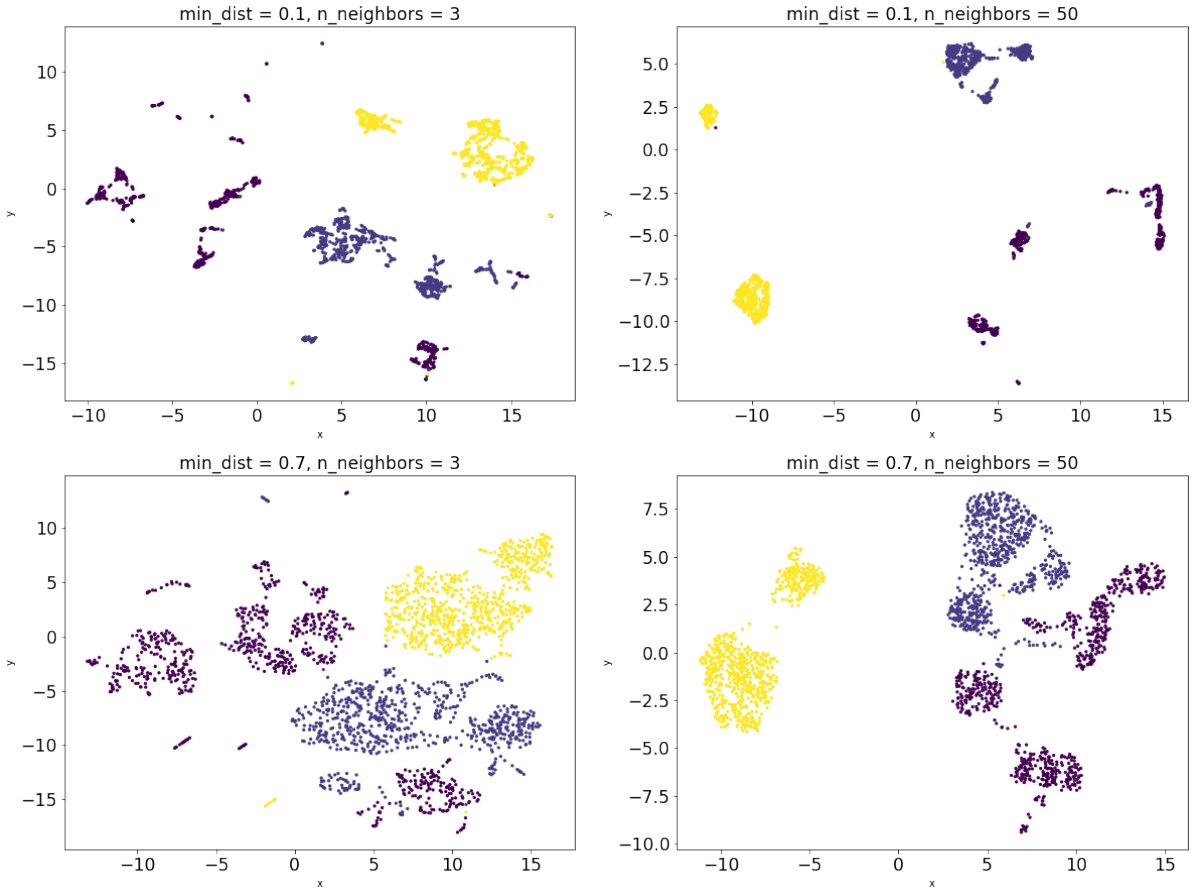


Figure 2. Variation of the UMAP hyperparameters neighbors and min dist

Another important parameter of the algorithm is the distance or similarity metric. There are many distance and similarity metrics to choose from: euclidean, Manhattan, cosine, correlation and many more. For this thesis we focus on four:

- Euclidean and Manhattan: The distance between two p-dimensional instance x_i and x_j is calculated using the Minkowski metric:

$$d(x_i, x_j) = (|x_{i1} - x_{j1}|^g + |x_{i2} - x_{j2}|^g + \cdots + |x_{ip} - x_{jp}|^g)^{1/g} \tag{4}$$

  Euclidean distance uses $g = 2$ and Manhattan distance uses $g = 1$.

- Cosine: When the angle between two vectors is a meaningful measure of their similarity, then the normalized inner-product can be an appropriate measure of their similarity.

$$s(x_i, x_j) = \frac{x_i^T \cdot x_j}{||x_i|| \cdot ||x_j||} \tag{5}$$

- Correlation: The correlation measure is defined as:

$$s(x_i, x_j) = \frac{(x_i - \overline{x}_i)^T \cdot (x_j - \overline{x}_j)}{||x_i - \overline{x}_i|| \cdot ||x_j - \overline{x}_j||} \tag{6}$$

where $\overline{x}_i$ denotes the average value of x over all dimensions.

There is also the possibility of computing the distance metric beforehand and then continuing from there. This is useful when a specific implementation of the algorithm in a software package doesn't support the distance function you are looking for. This is the case with the distance metric for mixed attributes and the UMAP implementation in Python.

2.3.3 Non-linear: t-distributed Stochastic Neighbor Embedding (t-SNE)

t-SNE is very similar to UMAP in that they are both non-linear dimension reduction methods that aim to end up with 2 or 3 final dimensions while also maintaining as much of the structure as possible. t-SNE main objective is to preserve as much of the local neighborhoods as possible, while UMAP tries to preserve global structure as well.

Stochastic Neighbor Embedding starts by converting the distance between the points into conditional probabilities that represent similarities. The similarity of data point x_j to data point x_i is the conditional probability, $p_{j|i}$, that x_i would pick x_j as its neighbor if neighbors were picked in proportion to their probability density under a gaussian centered at x_i. The mathematical expression of this condition is given by:

$$p_{j|i} = \frac{exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_{k \neq i} exp(-||x_i - x_k||^2/2\sigma_i^2)} \tag{7}$$

where σ_i is the variance of the gaussian that is centered on data point x_i. x_k is every other point in the dataset that is not x_i. These conditional probabilities are symmetrized to obtain the joint probability p_{ij}, calculated by:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2} \tag{8}$$

The joint probability for the low dimensional points (y_i and y_j), which we denote by q_{ij} is calculated as follows:

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq l}(1 + ||y_k - y_l||^2)^{-1}} \tag{9}$$

This calculation of the joint probability uses the student-t distribution with a single degree of freedom. t-SNE aims to find a low dimensional representation of the data that minimizes the mismatch between p_{ij} and q_{ij}. In order to do this, t-SNE minimizes the Kullback-Leibler (KL) divergence using a gradient descent method. The cost function C is given by:

$$C = KL(P||Q) = \sum_i \sum_j p_{j|i} log \frac{p_{ij}}{q_{ij}} \tag{10}$$

in which P represents the joint probability distribution over all other data points x_i and Q represents the Student-t based joint probability distribution over all other low dimensional points y_i.

One of the parameters the user has to define in t-SNE is the perplexity. This starts with the selection of a variance σ_i. In dense regions a smaller value of σ_i is more appropriate than in sparser regions. In order to find the perfect σ_i t-SNE performs a binary search for the value of σ_i that produces a P with a fixed perplexity. The perplexity is defined as:

$$Perp(P_i) = 2^{H(P_i)} \tag{11}$$

where $H(P_i)$ is the Shannon entropy of P_i given by:

$$H(P_i) = -\sum_j p_{j|i} log_2(p_{j|i}) \tag{12}$$

The perplexity can be interpreted as a smooth measure of the effective number of neighbors. It can also be interpreted as a continuous analogue to the k nearest neighbors, to which t-SNE will attempt to preserve distances. Its typical values are between 5 and 50 [21]. The last step is to minimize the cost function with the gradient descent method. This can be done by:

$$\frac{\delta C}{\delta y_i} = 4 \sum_{j \neq i} (p_{ij} - q_{ij})(y_i - y_j)(1 + ||y_i - y_j||^2)^{-1} \tag{13}$$

This is the interpretation of the gradient descent method done by the algorithm made by . The terms $(1 + ||y_i - y_j||^2)^{-1}$ are added to the normal t-SNE formulas as a way to reduce the crowding problem. The crowding problem is present when the area of the two-dimensional data set that is available to accommodate moderately distant data points is smaller than the area available to accommodate nearby data points [32].

Even though t-SNE is a reliable method to reduce the number of dimensions, it can be slow to compute. Therefore, there are a couple of ways to reduce the computing time, for example when calculating the p_{ij} and q_{ij}. This requires computing all the pair-wise interactions with a time complexity of $O(N^2)$. Moreover, q_{ij} must be computed every single iteration of the optimization while p_{ij} should only be computed once, as the input space stays the same. The reason behind this is the normalization constant of q_{ij} which we are going to define as Z.

$$Z = \sum_{k \neq l} (1 + ||y_k - y_l||^2)^{-1} \tag{14}$$

The rewritten gradient update rule is:

$$\frac{\delta C}{\delta y_i} = 4 \left(\sum_{j \neq i} p_{ij} q_{ij} Z (y_i - y_j) - \sum_{j \neq i} q_{ij}^2 Z (y_i - y_j) \right) \tag{15}$$

There are other improvements to be made to the computation of the p_{ij} and q_{ij}. p_{ij} (attractive forces) can be computed faster by reducing the number of nearest neighbors to 3, then computing it approximately. The repulsive forces or q_{ij} can be calculated more easily with the Barnes-Hut t-SNE [44].

The process to get the t-SNE embedding consists of the following steps:

1. Calculate the affinities between data points
2. Generate the initial coordinates for the embedding
3. Construct the Embedding object
4. Optimize the embedding
   (a) Early exaggeration phase
   (b) Regular optimization
5. Transform the test points

This process allows new points to be added after the fact to the embedding without having to recalculate the embedding. This is important in real life scenarios where one might train an

embedding and want to know where new points (maybe new data from the same sensors) fall in the embedding and which cluster they belong to.

t-SNE hyperparameters should be carefully selected in order to achieve the desired data set. These are:

- Perplexity: Perhaps the most important parameters in t-SNE. It can be thought of as the balance between preserving the local and the global structure of the data.
- Exaggeration: This is used during the early exaggeration phase. It increases the attractive forces between the points and allows points to move around more freely, finding their nearest neighbors more easily.
- Iterations: number of times the algorithm runs before stopping.
- Learning rate: Controls the step size in the gradient descent optimization procedure. Typically ranges from 100 to 1000, depending on the size of the dataset.
- Momentum: With the momentum, the gradient descent keeps a sum of exponentially decaying weights from previous iterations, speeding up convergence. In the early exaggeration phase it defaults to 0.5 and then it is increased on the next phase, to speed up convergence.

Deciding on a good perplexity value depends mostly on the size of the data set. Most implementations default perplexity at 30, this is enough for small sized data sets (100), it will uncover the global structure since each point will preserve distances to a big portion of the data set. The opposite can also happen. With a value of 30 and a big dataset (500k points) it may not preserve the global structure. Computing time is affected linearly by the choice of perplexity, higher values of perplexity will take longer to compute. Selecting an exaggeration parameter is also crucial. The most typical value for the exaggeration is 12 during the early exaggeration phase, but higher values, when in combination with different learning rates, have been shown to work well.

An example of how perplexity and exaggeration affect the embedding are shown in Figure 3.

2.4 Clustering

Clustering is an unsupervised learning method for pattern classification and a vital step for exploratory data analysis [48]. Clustering is the task of grouping a set of objects in a way that objects in the same group or cluster are more similar to each other than to the ones in a different cluster. There are many different clustering methods. Each method focuses on a different aspect of what defines a cluster such as, the density of the points in the cluster, the distance between the different cluster members or the statistical distribution of the cluster.

While clustering has many uses, this thesis focuses on clustering for the purpose of exploratory data analysis. Exploratory data analysis is the process of looking for patterns in a data set, primarily with the goal of generating new hypotheses or research questions about the data set in question.

2.4.1 Types of clustering

The main reason for having many clustering methods is the fact that the notion of cluster is not precisely defined. Consequently, many clustering methods have been developed, each of which

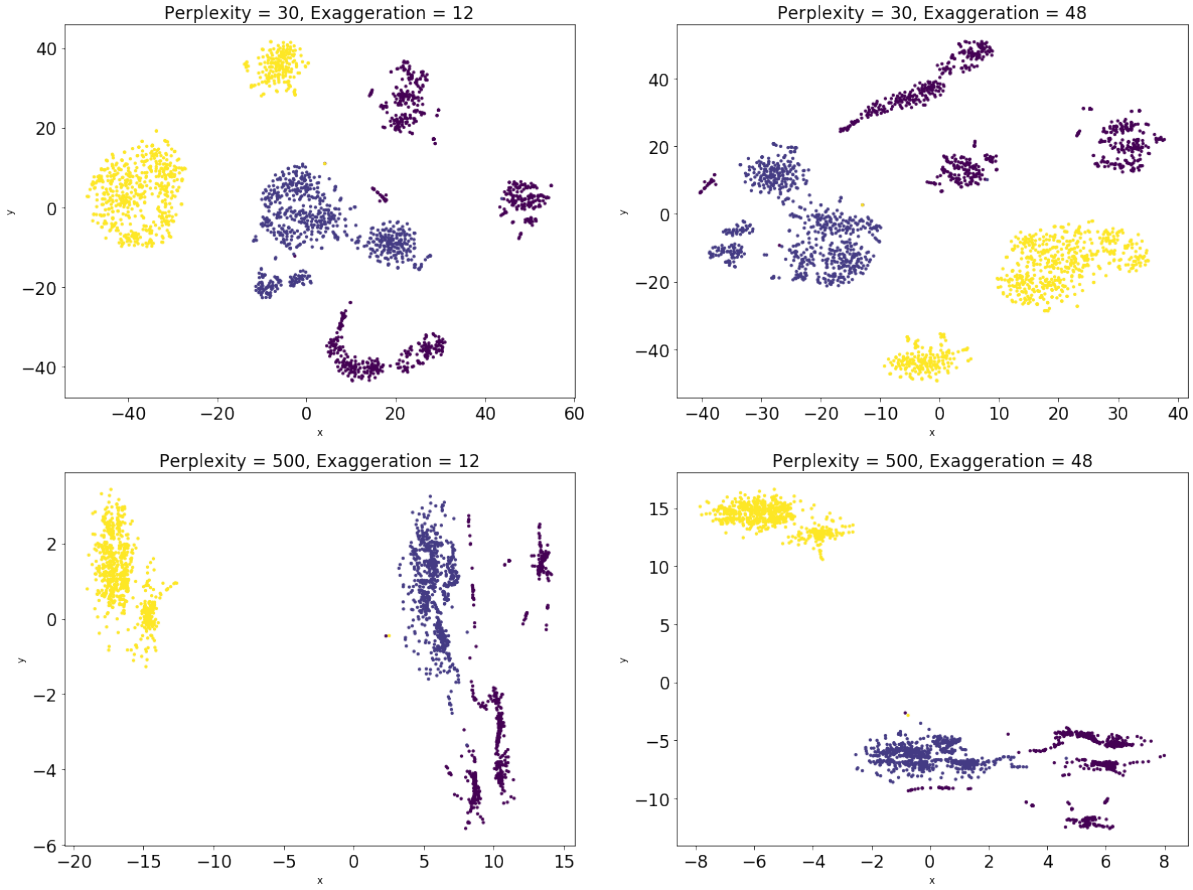


Figure 3. Variation of the t-SNE hyperparameters exaggeration and perplexity

uses a different induction principle. There are many types of clustering (according to Han and Kamber (2001):

- Hierarchical clustering
- Partitioning clustering
- Model-based
- Density-based
- Grid-based

In this work, hierarchical, partitioning and density-based will be covered in detail. For more information on the other types of clustering, read Lior Rokach's book "Data Mining and Knowledge Discovery", specifically the chapter Clustering Methods. [38]

2.4.1.1 Hierarchical clustering

These methods construct clusters by recursively partitioning instances in either a top down or a bottom-up fashion. The result is a dendogram, representing the nested grouping of objects and similarity levels. The resulting clusters are obtained by cutting by the desired similarity level. These methods can be sub-divided into further categories as follows:

- Agglomerative hierarchical clustering: Each object represents initially a cluster of its own. Then they are successively merged until the desired structure is achieved.

- Divisive hierarchical clustering: All objects initially belong to one cluster. Then they are divided into sub-clusters until the desired structured is achieved.

Hierarchical clustering is also employed for predicting multiple continuous target variables from high-dimensional and heterogeneous data sets, and it can also be taken together with autoregressive models for identifying nonlinear systems from empirical data and detecting faults in single-input and single-output systems. It can also be employed in signal processing for analyzing low signal-to-noise ratio in acoustic emission signals for monitoring crack growth [48].

2.4.1.2 Partitioning clustering

Partitioning methods relocate instances by moving them from one cluster to another, starting from an initial partitioning. They require that the number of resulting clusters is preset by the user. A relocation method iteratively relocates points between the k clusters.

2.4.1.3 Density-based clustering

Density-based methods locate regions or neighborhoods of high density that are separated from one another by regions of low density. The key idea is that the density of the neighborhood has to exceed some threshold. In order to apply this, the neighborhood of a given radius has to contain at least a minimum number of points. The shape of this threshold depends on the distance function. Density-based methods assume that the points that belong to each cluster are drawn from a specific probability distribution (Banfield and Raftery, 1993). An example of a density-based algorithm is DBSCAN (Density-Based Spatial Clustering of Applications with Noise). It discovers clusters of any shape and is efficient for large datasets. The method searches for clusters by looking at the neighborhood of each point in the dataset and checks if it contains more than the minimum number of points [9].

The clustering methods we discuss are: k-means clustering (2.4.2) and hierarchical density-based (2.4.3).

2.4.2 K-means clustering

K-means divides the data set into k clusters defined by the mean of each cluster called centroid. It is an iterative process where the algorithm tries to minimize the inertia of the cluster or the sum of squared criterion (Equation 16):

$$\sum_{i=0}^{n} \min_{\mu_j \in C} (||x_i - \mu_j||^2) \tag{16}$$

The iterative process consists of the following steps:

1. Choose the initial centroids.
2. Loop of the next two steps:
   (a) Assign each point to its nearest centroid
   (b) Create new centroids by taking the mean value of all the samples assigned to each previous centroid.
3. Compute the difference between the previous and new centroids.

4. If the difference is lower than a threshold stop the loop.

Given enough time, K-Means will converge, but maybe only to a local minimum. This is highly dependent on the initialization of centroids. The computation is done several times, with different initializations of the centroid. One way to address the uncertainty is to use the k-means++ initialization scheme. It initializes the centroids to be distant from each other, leading to the best solution [28].

In order to select the correct number of clusters there is a method called the *elbow method*. This method employs the sum of squared errors and plots it against the number of clusters. There is a point where the error converges. This is the optimal point as increasing the number of clusters further will only introduce more computing time and complexity to the algorithm but no improvement in performance. This however doesn't always work, especially when the clusters are not well separated.

An example of the graph to select the number of cluster with the elbow method can be seen in Figure 4. Here there are two points that could be seen as elbow points, k=2 and k=4. The result of doing K-means with each number of clusters is represented above in Figure 5

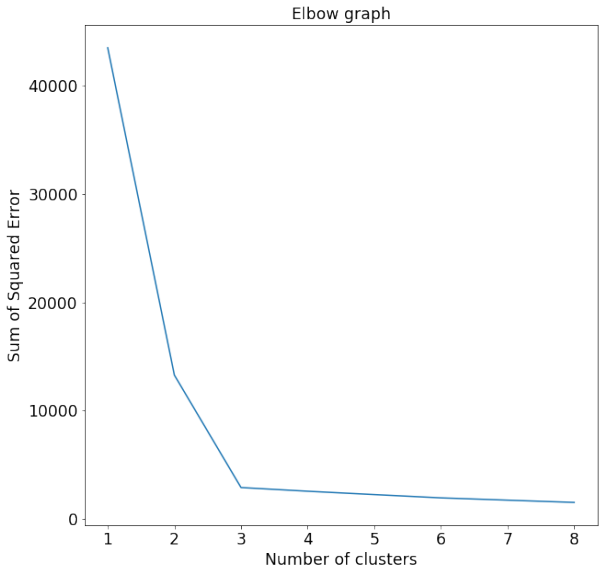


Figure 4. SSE graph

In the example both results can be convincing but one may think that selecting 2 for k might leave one of the clusters with too much inertia or sum of squared errors.

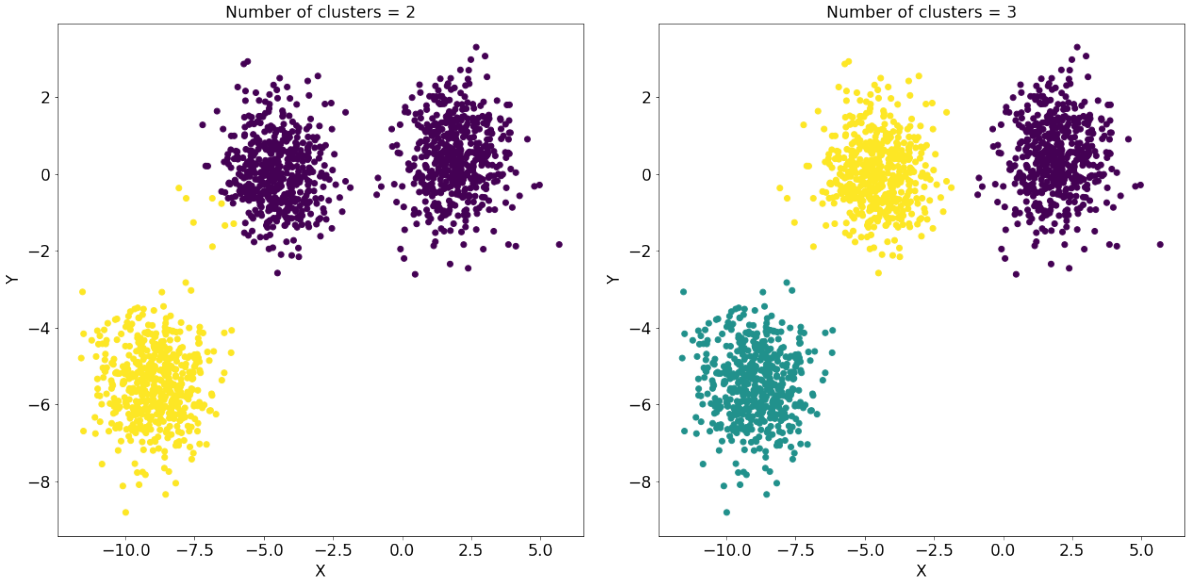
Other available methods to select the number of clusters use a different parameter called the silhouette score. This method is explained in Section 2.5.

K-means main advantages are:

- The math is simple, so the computation time is very small compared to other clustering algorithms.
- When using kmeans++ initialization, it is a deterministic algorithm.

K-means also has some drawbacks:

- The inertia criterion makes the assumption that clusters are convex and isotropic, therefore it responds poorly to elongated clusters or manifolds with irregular shapes.



(a) K-means with K=2 **(b)** K-means with K=3

Figure 5. Results by selected number of clusters

- Inertia is not a normalized metric: Lower is better. In high-dimensional spaces, Euclidean distances tend to be inflated (curse of dimensionality). One way to alleviate this problem is running a PCA dimension reduction in the preprocessing phase; this can also speed up the calculations. [28]

- K-means works poorly with clusters with different variances (See Figure 6).

- The user has to input the number of desired clusters a priori. This information is not always available to the user.

Examples of cases where k-means does a poor job can be seen in Figure 6

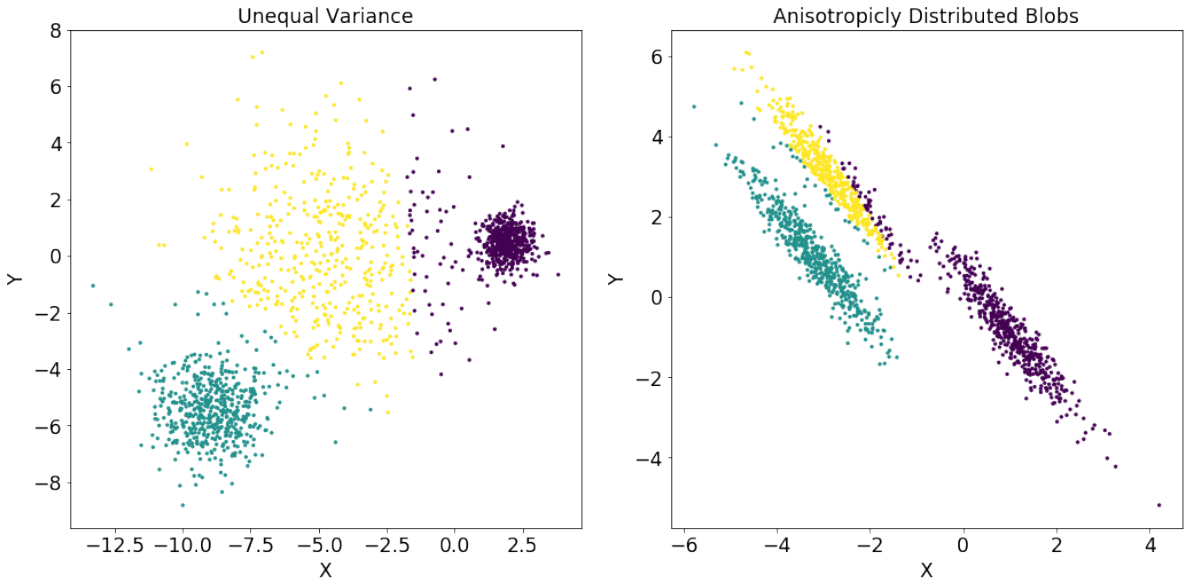


Figure 6. Examples where K-means behaves poorly

2.4.3 Hierarchical Density Based Spatial Clustering of Applications with Noise (HDBSCAN)

Many traditional clustering algorithms are poorly suited to exploratory data analysis tasks. In particular, most clustering algorithms suffer from the problems of difficult parameter selection, insufficient robustness to noise in the data, and distributional assumptions about the clusters themselves [5].

HDBSCAN takes a series of steps in order to get the clusters:

1. Transform the space according to the density
2. Build the minimum spanning tree of the distance weighted graph
3. Construct a cluster hierarchy of connected components.
4. Condense the cluster hierarchy based on minimum cluster size
5. Extract the stable clusters from the condensed tree

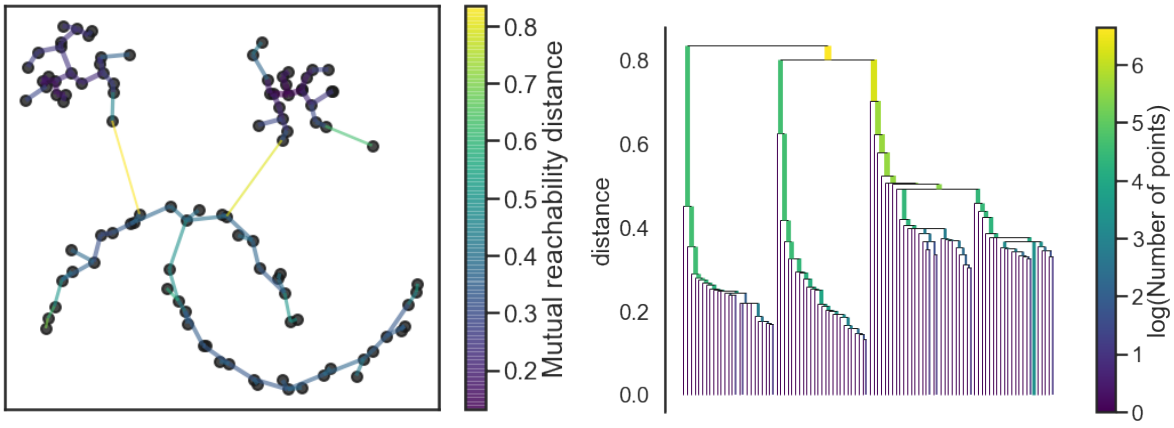
The method starts by computing the core distance of each point, this is the distance to its k-nearest neighbor defined by the parameter "minimum samples". It then transforms the space. First, an initial estimation of density is computed to distinguish the higher density areas from the lower density ones in order to make the algorithm more robust to noise or outliers in the data. This is done by computing the distance to the kth-nearest neighbor. We start by defining the core distance defined by parameter k for a point x and denote as $core_k(x)$. Then, we define a new distance metric between points called the mutual reachability distance:

$$d_{mreach-k}(a, b) = max(core_k(a), core_k(b), d(a, b)) \tag{17}$$

where $d(a, b)$ is the original metric distance between a and b (euclidean, for example). Under this metric, dense points remain the same distance from each other while sparser points are pushed away to be at least their core distance away from any other point. This is dependent on the choice of k (minimum number of points). Then, we find the clusters on the dense data. Conceptually, we consider the data as a weighted graph with the data points as vertices and an edge between any two points with weight equal to the mutual reachability distance of those points. We drop any edges with weight above a threshold (starting from high and steadily lowering it) and then disconnect the graph into connected components. At one point, we get the hierarchy of connected components at varying threshold levels. Next, we build the minimum spanning tree via Prim's algorithm [35], one edge at a time, adding the lowest weight edge that connects the current tree to a vertex not yet in the tree. You can see an example in Figure 7.

Given this minimum spanning tree the next step is to convert it into a hierarchy of connected components and view it as a dendogram. We then sort the edges of the tree by distance and then iterate through, creating merged cluster for each edge. Next, we join the edges together by the union-find data structure [10].

The next step is to select clusters by drawing a horizontal line by choosing a distance. This would mean that every cluster would have the same density. The next step tries to solve this. Once we have a value of minimum cluster size which HDBSCAN takes as a parameter, we can now go through the hierarchy and at each split ask if one of the new clusters created by the split has fewer points than the minimum cluster size. In the case that we have less points than the minimum we declare it to be 'points falling out of a cluster' and have the larger cluster retain the identity of the parent, marking which points fell and at what distance value that happened. If, on the other hand, the split is into two clusters each and at least as large as the minimum cluster size



(a) Minimum span tree **(b)** Single linkage tree

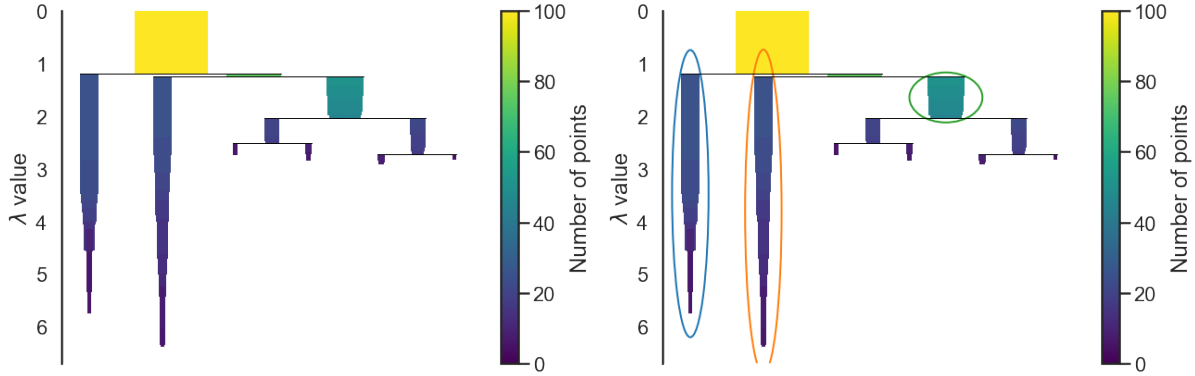
Figure 7. Example trees produced by HDBSCAN

then we consider that a true cluster split and let that split persist in the tree. After doing this we end up with something like in Figure 8.

Now, intuitively, we choose the clusters that persist and have longer lifetime (survive longer with different levels of alpha). To make a flat clustering we need to add a requirement that, if you select a cluster, then you can not select any cluster that is descendant of it. Instead of the distance we use its inverse $\lambda = \frac{1}{distance}$. For a given cluster we define values λ_{birth} and λ_{death} to be the value when the cluster split off and becomes its own cluster and the value when it split into smaller clusters. λ_p is the value at which each point p falls out of the cluster in its lifetime. At first, we declare all leaf nodes to be selected clusters, then go up the tree. If the sum of the stabilities of child clusters is greater than the stability of the cluster, then we set the cluster stability to be the sum of the child stabilities. If on the other hand, the cluster stability is greater than the sum of its children then we declare the cluster to be a selected cluster and unselect of all its children. The stability is computed as:

$$stability = \sum_{p \in cluster} (\lambda_p - \lambda_{birth}) \tag{18}$$

An example of this is shown in Figure 8



(a) Condensed tree **(b)** Condensed tree with selected clusters

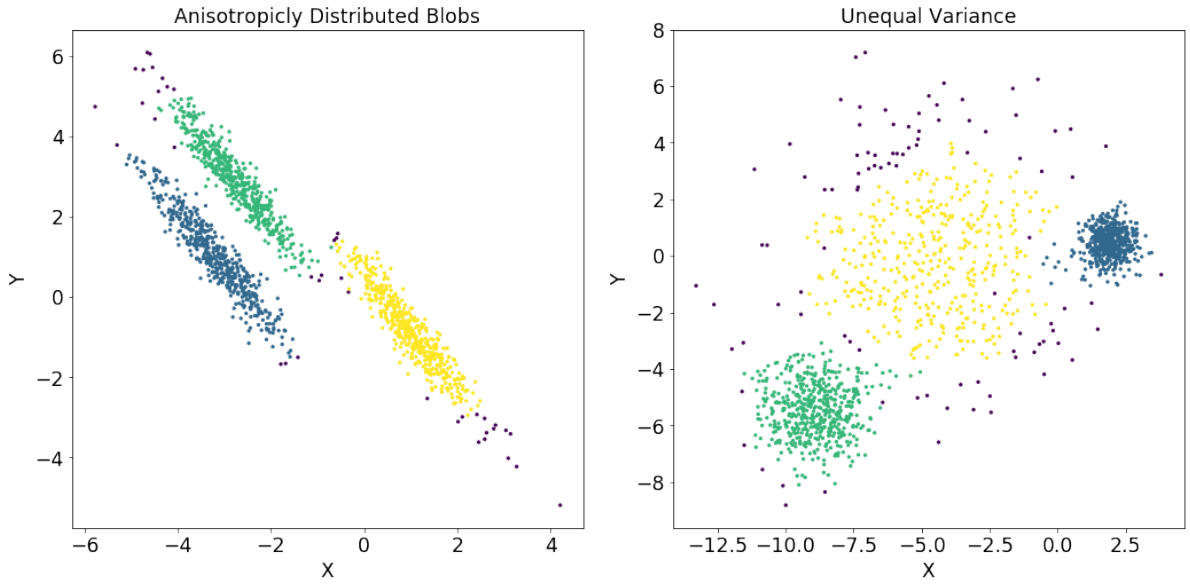
Figure 8. Example condensed trees produced by HDBSCAN

It can be informative to look at the hierarchy represented in the condensed tree and potentially make use of the extra information contained. This way one can see the dependencies clearly and where these clusters form and decide a way to improve the clustering. Additionally, each point has its own λ_p representing the probability of this point to be in the cluster the algorithm has given it. An example of the data given by HDBSCAN in dataframe form is shown in Table 3. It represents where the yellow cluster at the top is divided into the light blue at the left and the green at the right, and then the green is divided into the orange and the darker blue and so on [22].

parent	child	lambda value	child size
100	101	1.198	25
100	102	1.198	75
102	104	1.248	50
104	105	2.040	21

Table 3. Example of data given by HDBSCAN

There is an accelerated version of HDBSCAN that Leland McInnes investigated [24]. It is the same method in principle as HDBSCAN but it doesn't suffer from its computation time problems. An example of HDBSCAN in action is shown in Figure 9. Comparing Figure 9 and Figure 6 from the K-means section, it is visible that HDBSCAN was able to better separate the clusters, while also giving more information like noise in the data, for example on the right plot of Figure 9. This is due to the fact that K-means works at its best when the clusters are spherical and with even size and HDBSCAN relies on the density of the data to distinguish between clusters. On the plot on the left, the clusters have an anisotropic distribution and therefore, the clusters found by K-means have areas of low density inside, which is not the way the human eye would separate the clusters. The plot on the right has a different, but similar problem, even though the clusters have an spherical shape, K-means is mixing together clusters with different densities, while HDBSCAN does a much better job.



(a) Anisotropic distribution **(b)** Unequal variance

Figure 9. Sample data clustered by HDBSCAN

2.4.3.1 Predicting clusters for new points

It is often useful to train a clustering model on a large amount of data and then query the model repeatedly on small amounts of new data. This is hard for HDBSCAN to do, as it might change the model and create a new cluster with the new points of data received or alter the clusters already created. But, there is a way of maintaining the model fixed and then assign the new data points to one of the clusters. This is the same as determining where in the condensed tree the new data point would fall, assuming the condensed tree is not changed.

2.4.3.2 HDBSCAN advantages and disadvantages

The main advantages of HDBSCAN are:

- HDBSCAN runs faster than other density based methods like DBSCAN. This is very important in data sets with a size of millions of data points.
- The method gets better results in situations where the various clusters in the data set have different densities. Other density based clustering methods like DBSCAN will see various clusters where really there is only one because the density of said cluster is lower than the cluster with the highest density.
- Uses the minimum number of points in the cluster as a parameter. This is more intuitive than the the parameters used in other density-based clustering methods like DBSCAN.
- Doesn't require the selection of the number of clusters, either explicitly nor implicitly through parameters.
- Possibility of visualizing the results with a condensed tree dendogram.
- While K-means is limited to euclidean distances (it can only work with numerical data), HDBSCAN can work with binary, categorical, strings and mixed type by using different distance metrics like Jaccard, Levenshstein or Gower distance [27].

HDBSCAN's main disadvantages are:

- HDBSCAN is less scalable than K-means because it needs more computations. K-means computational complexity $O(n \cdot k \cdot d \cdot i)$ is much lower than HDBSCAN $O(n^2 \cdot d)$ where k is the number of clusters, i is the number of iterations needed until convergence, d is the number of dimensions and n is the number of datapoints to be clustered.
- Memory consumption grows in a quadratic way with respect to the number of datapoints as opposed to linear in K-means [27].

2.5 Measuring clustering performance

As the unsupervised learning methods do not contain the output label data, comparing the results with these is no easy feat. There are two general ways of comparing clustering methods: first, considering how easy they are to use and second, to evaluate how well they perform. The first type usually only considers computing time and storage requirements. Yet, how well a clustering method performs is still the biggest concern. While both criteria are important, deciding between a good and slow method and a quick and bad method requires quantification [36].

The term to define clustering performance is the *cluster validity*. It provides a way of validating the quality of clustering algorithms and the means of discovering the a natural structure of data sets. Furthermore, its measurements are used to compare the results of different clustering algorithms, as well as the same clustering algorithm with different number of clusters. Cluster validity indices are classified into internal and external indices, the former are usually based on information intrinsic to the data while the latter are based on prior knowledge about the data. Since external indices are based mainly on prior information of the data, the indices are used for choosing the best clustering method for a specific dataset. On the contrary, internal indices can be used to choose the best clustering algorithm as well as the optimal number of clusters, without he need for additional information [47].

2.5.1 Silhouette score (SC)

Silhouette analysis refers to the method of interpretation and validation of consistency within clusters of data. Silhouette analysis is used to study the separation distance between the resulting clusters. The silhouette score has a range of [-1, 1] where coefficients near 1 indicate that the sample is far away from the neighboring clusters. A value of 0 indicates that the sample is on the decision boundary between two neighboring clusters and -1 indicate that those samples might have been assigned to the wrong cluster.

This coefficient is calculated with the mean intra-cluster distance a and the mean nearest-cluster distance b. a is the distance to the points that are in the same cluster and b is the distance between the sample and the nearest cluster that the sample is not a part of.

$$silhouettescore = \frac{b - a}{max(a, b)} \tag{19}$$

The silhouette score can also be used to select the number of clusters in algorithms that require that number a priori, like K-means. This is done with the silhouette analysis. Silhouette analysis is a way of measuring how close each point in a cluster is to the points in its neighboring clusters. The steps in the silhouette analysis are:

1. Calculate the silhouette score of each point.
2. Sort the points of each cluster from highest to lowest and plot them.
3. Calculate the mean of every silhouette score.

In order to measure the performance of a method there are a couple points to consider:

- The mean value should be as close to 1 as possible.
- The plot of each cluster should be above the mean value as much as possible. Any cluster below the mean is not desirable.
- The width of the clusters should be as uniform as possible.

If the plot of your silhouette analysis achieves all of the above, then it means that the number of clusters selected is adequate. A comparison between two plots, one that satisfies the above requirements and one that doesn't is shown in Figure 10

In Figure 10 it is clearly seen that every cluster has at least one point inside the mean silhouette score while on Figure 11 clusters 4 and 2 are completely under the mean silhouette score. Moreover, the mean silhouette score of the right plot is lower than the one on the left. Therefore, it can be concluded that K=4 produces better clustering than with K=5.

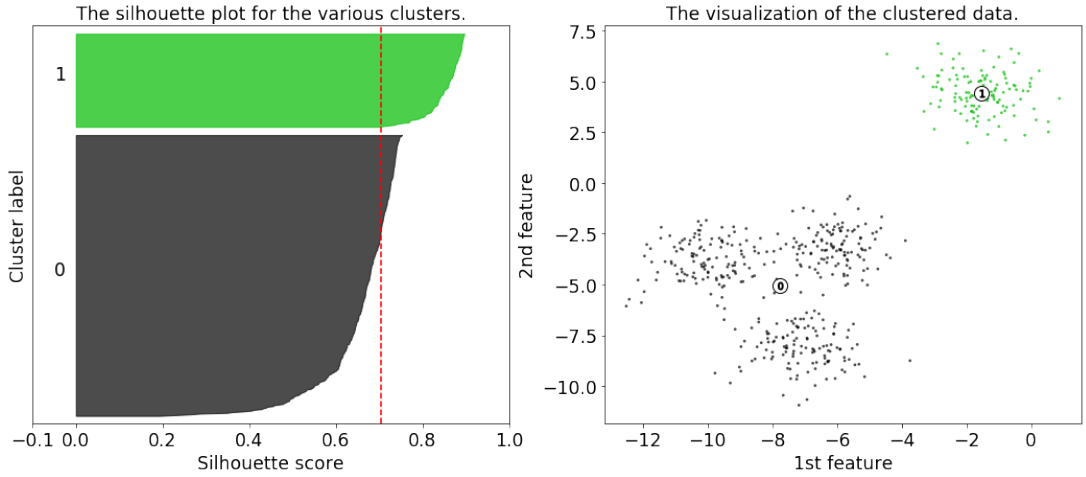


Figure 10. An exemplary silhouette analysis of a K-means clustering with k=2

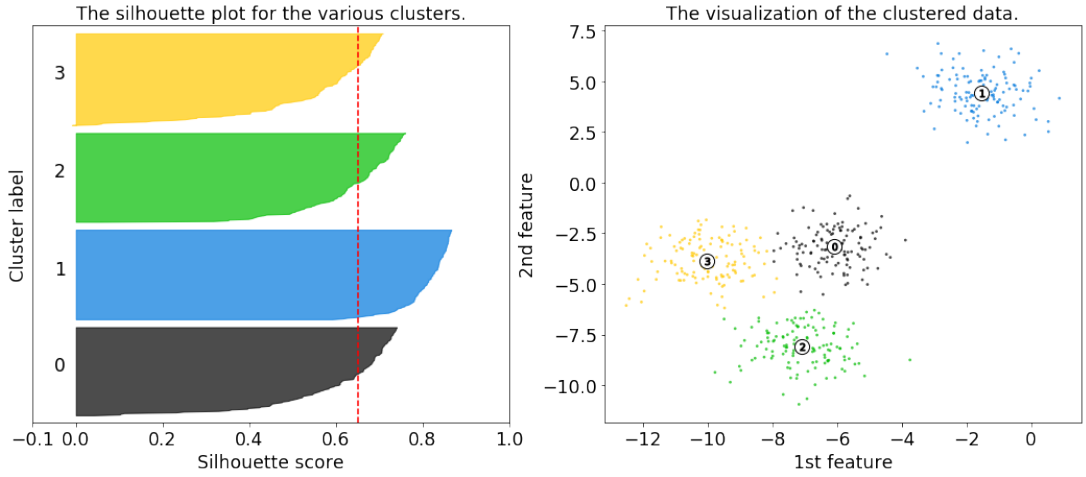


Figure 11. An exemplary silhouette analysis of a K-means clustering with k=4

2.5.2 Accuracy score (AS)

The accuracy score function computes the precision, either the fraction or the total number of correct predictions with which an algorithm has predicted an output. If the whole set of predicted labels for a data set perfectly match with the true set of labels, then the accuracy score is 1.0; otherwise it is 0.0 [30]. If y_{predi} is the predicted value of the ith sample and y_{truei} is the corresponding true value, then the fraction of correct predictions over $n_{samples}$ is defined as:

$$accuracy(y_{true}, y_{pred}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} 1(y_{predi} = y_{truei}) \tag{20}$$

2.5.3 Adjusted Rand Index (ARI)

Given the knowledge of the ground truth class assignments and our clustering algorithm assignments of the same samples, the adjusted Rand index is a function that measures the similarity of the two assignments, ignoring permutations and with chance normalization. It is the corrected-for-chance version of the Rand Index. Such correction for chance establishes a baseline by using the expected similarity of all pair-wise comparisons between clusterings

specified by a random model. Though the Rand Index can only vary from 0 to 1, the Adjusted Rand Index can yield negative values [28].

There are some advantages to using this method compared to the accuracy score [28]:

- Random (uniform) label assignments have a ARI score close to 0.0 for any value of n clusters and n samples (which is not the case for raw Rand index or the V-measure for instance).

- Bounded range [-1, 1]: negative values are bad (independent labelings), similar clusterings have a positive ARI, 1.0 is the perfect match score.

- No assumption is made on the cluster structure. It can be used to compare clustering algorithms such as k-means which assumes isotropic blob shapes with results of spectral clustering algorithms which can find cluster with folded shapes.

But at the same time there are some disadvantages to using this metric [28]:

- Contrary to inertia, ARI requires knowledge of the ground truth classes while almost never being available in practice or requires manual assignment by human annotators (as in the supervised learning setting).

The ARI is computed in this way: The contingency table is the summary of the overlap between two subsets $X = X_1, X_2, ..., X_r$ and $Y = Y_1, Y_2, ...Y_s$ where each entry n_{ij} denotes the number of object common between X_i and $Y_j : n_{ij} = |X_i \cap Y_j|$. An example of contingency matrix composed of X and Y (the contingency matrix is very similar to the confusion matrix) is:

	Y_1	Y_2	Y_s	Row sums
X_1	n_{11}	n_{12}	n_{12}	a_1
X_2	n_{21}	n_{22}	n_{2s}	a_2
X_r	n_{r1}	n_{r2}	n_{rs}	a_r
Column sums	b_1	b_2	b_s	n

Table 4. Contingency matrix

$$ARI = \frac{\sum_{ij}\binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n}{2}} \tag{21}$$

where n_{ij}, $a_i$and b_j are the values of the contingency table.

2.5.4 Confusion matrix

Confusion matrices are used to compare how a clustering or classification method has predicted the labels, how many false negatives and how many false positives an algorithms has (how many times some label should have been selected and it wasn't and how many times one label was selected even though it shouldn't have) [30]. An example of a confusion matrix is shown in Figure 12.

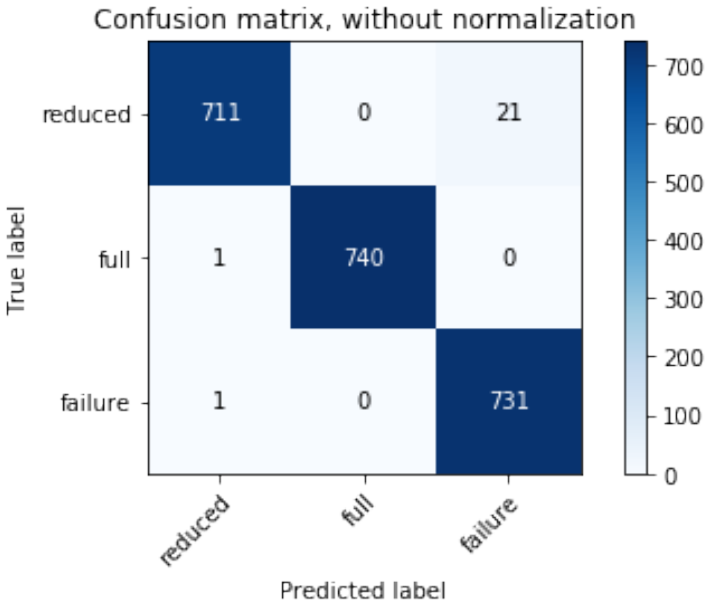


Figure 12. Example confusion matrix

The name confusion matrix comes from the fact that it makes it easier to see if the system is confusing two classes, for example, mislabelling one as another.

2.6 Condition Monitoring

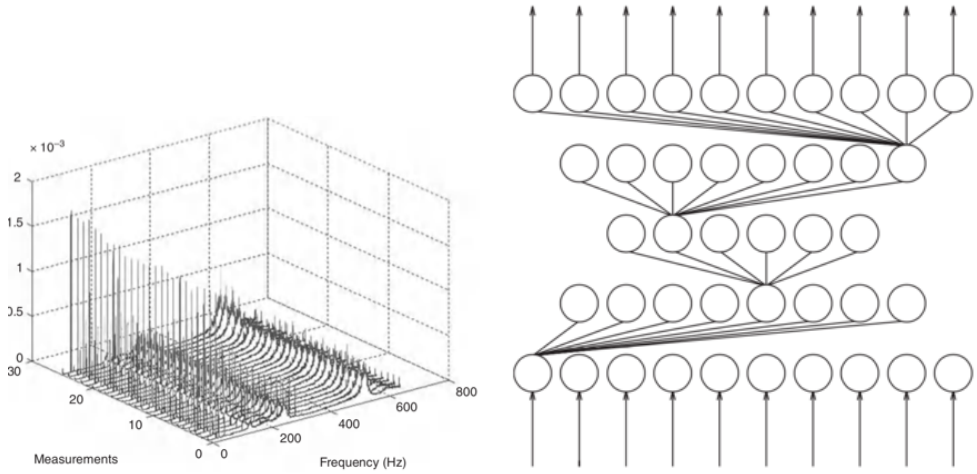
CM refers to the process of implementing damage detection and characterization of engineering structures. Damage is defined as changes to the material or geometric properties of a structural system. The CM process involves the observation of a system over time using periodically sampled dynamic response measurements from an array of sensors. The state of the machine must remain in the domain specified in the design, although this can be altered by normal aging due to usage, by the action of the environment and/or by accidental events.

CM methodologies can be divided into two main categories: physical model-based and data model-based approaches. A typical engineering approach in CM adopts a physic-based model of the structure, usually based on finite element analysis. The differences between measured data and the data generated by the model are used to identify any damage. However, a numerical model is not always available in practice and does not adapt well to changes in environmental and operational conditions. This challenge motivates the use of a data model-based approach which establishes a model by learning from measured data and then makes a comparison between the data model and new measured responses to detect damage. This approach normally uses techniques in machine learning [46].

CM has developed from a crude qualitative procedure like holding a screwdriver on the machine housing and listening for changes in the acoustic signal to very sophisticated techniques that use high precision sensors. Even though we now have modern data acquisition systems, qualitative interpretation of vibration signatures both in the frequency and time domain still are the primary data interrogation procedures [45]. The approach usually taken has been to consider the detection of damage qualitatively on a fault-by-fault basis by examining vibration signatures measured on the equipment housing for the presence and growth of peaks at certain frequencies like multiples of shaft speed. An example of this is shown in Figure 13.

CM can be passive or active. Lets imagine a structure equipped with sensors, this structure interacts with its surroundings in such a way that its state and its physical parameters are evolving.

When the manufacturer is only monitoring the evolution of the system thanks to the embedded sensors, then this is called passive monitoring. If the manufacturer has equipped the structure with both sensors and actuators, they can generate perturbations in the structure and then use the sensors to monitor the response of the structure. This is called active monitoring.



(a) Multiples of shaft speed **(b)** Neural network

Figure 13. Sensor data showing multiples of shaft speed and a representation of a neural network.[45]

Although CM has progressed a lot, detecting damage at a very early stage is still an open question, as problems exist in real engineering for CM, especially in industrial applications. [48].

Recently, more general approaches to damage detection in machinery, like machine learning, have been developed. These approaches use analytic methods to assess the presence of damage on a statistical basis. When damage is detected one can now program maintenance operations ahead of the complete failure of the machine and excessive machine downtime. Adapting and staying ahead of machine failure is called Predictive Maintenance. In recent years, predictive maintenance has risen in the Internet of Things data industry for two reasons:

- Modern machinery comes with embedded chips to control and take readings from the machine. This data hasn't always been stored by the owners, but it is available for them to start recording.

- New machines have embedded sensors and storage mechanisms to implement predictive maintenance in a cost efficient way.

Predictions can be made by taking the most recent data points from a machine and running a machine learning algorithm against them. In the case of predictive maintenance, this enables the customization of the maintenance algorithm to each machine, a departure from the traditional method where the maintenance was based on schedules or usage thresholds such as the number of products made or the number of miles driven in a car. This can only be done when the data includes the state of the machine and one can label the data as healthy and not healthy. When this is the case, the machine learning problem is supervised. But there are times when this data is unavailable or it is not possible to obtain. Then it is called unsupervised machine learning.

CM with machine learning has been successfully applied to a number of situations with varying degrees of success.

- K-means with spectral features: Sydney Harbour Bridge [1]: This paper focuses on a subset of the components of the Sydney's Harbour Bridge. They extracted spectrum-driven features from measured accelerations from the jack arches. They then used this features as an input to a k-means clustering algorithm. Finally, they proposed a selection mechanism to process the clustering result and identified the jacks with structural damage or another system issue. They also discovered some of the issues of this approach like the use of real-time data streams, how to distribute the computation among the embedded nodes on a bridge to minimize network communication and computing delay or how to increase the resilience of the approach when multiple nodes have system issues. They are currently working on this CM system on the Sydney Harbour Bridge on the study of the variabilities induced by daily and seasonal local environmental changes. They have continued their research in the following topics:
  - The estimation of the variability induced by daily and seasonal changes in environmental conditions
  - The dynamic deployment of algorithms to on-site computing nodes to automatically filter out environmental effects and known device issues.
  - The application of a CM system to bridges in rural and remote areas
- Multilayer Perceptron (MLP) for remaining useful life estimation: Herzog et al. (2009) applied an MLP neural network for estimating the residual life of a machine and its components. A multilayer perceptron is a class of neural network that uses a supervised and non-linear way of training and has many layers (input, hidden and output). They trained and tested numerous neural network variations with data from two different data sets. The first data set represented the renewal case where the failed unit was repaired and restored to its initial condition. Data was collected in the laboratory by fatigue loading with a hydraulic actuator a series of similar test pieces. The average prediction error of the various neural networks being compared varied from 431 to 841 seconds on this dataset, where test pieces had a characteristic life of 8971 seconds. The second data set was gathered from a collection of pumps used to circulate a water and magnetite solution within a plant. The data was obtained from a repaired system affected by reliability degradation. When optimized, the multilayer perceptron neural networks gave a sums-of squares error within 11.1% of each other for the renewal data set. The small number of inputs and poorly mapped input space on the second data set indicated that much larger errors were verified on some of the test data. The prospect of using neural networks for residual life prediction and the advantage of integrating condition-based data into the model was demonstrated in both examples.
- Bayesian networks: Kim et al. (2011) successfully applied a Bayesian framework to identify faults in a partially observable system subject to random failures. The deterioration of a system was modeled with a hidden 3-state continuous time homogeneous Markov process with states 0 and 1 not being observable, signifying good and warning situations respectively, and only State 2 being observable. The model's parameters were identified using an Expectation Maximization procedure and a cost-optimal Bayesian fault prediction scheme was presented. The technique was validated using real data from a spectrometric analysis of oil samples from the transmission units of heavy hauler trucks.
- Gaussian Mixture Models: Boutros and Liang (2011) applied the discrete hidden Markov model for the detection and diagnosis of bearing and cutting tool faults. Their method was tested and validated using two situations, tool fracture, and bearing faults. In the

first situation, the model correctly detected the state of the tool and, in the second case; the model classified the severity of the fault seeded into two different engine bearings. The result obtained for fault severity classification was above 95%. In addition to the fault severity, a location index was developed to determine the fault location and gave an average success rate of 96%.

- Deep learning: Linxia Liao (2016) applied a novel deep learning structure to predict asset failure probability using sequential data by stacking deep feature learning and survival analysis. The learning process learns the feature representation and prediction task together using stochastic gradient descent. It provides an end-to-end prediction model using sequential data. The proposed model is validated using data collected from a fleet of mining trucks. The results show that the model can predict the failure probability with acceptable result in a leave p-out test.

These studies have successfully applied various machine learning methods to the problem of condition monitoring for the identification of failures but they each had their problems. We think the proposed unsupervised learning method of using Uniform Manifold Approximation and Projection and the clustering method Hierarchical Density Based Spatial Clustering for Applications with Noise can improve the results achieved in these cases.

On the next section, we applied unsupervised learning with UMAP dimension reduction and HDBSCAN clustering and compared its performance with various metrics against common dimension reduction methods like PCA and t-SNE and against other clustering methods like k-means. In the end we reached a conclusion about which is the best method for a small data set and then we applied the same method into another data set but this time more complex with less available information.

Chapter 3

Exemplary Implementation with the Hydraulics Data Set

3.1 Information about the dataset

The Condition Monitoring of Hydraulics system data set was made by ZeMA GmbH in Saabrücken. It addresses the condition assessment of a hydraulic test rig based on multi-sensor data. Four fault types were superimposed with several severity grades impeding selective quantification.

The dataset was experimentally obtained with a hydraulic test rig. This rig consisted of a primary working and a secondary cooling-filtration circuit which were connected via the oil tank. The system cyclically repeated constant load cycles (duration 60 seconds) and measured process values such as pressures, volume flows and temperatures while the condition of four hydraulic components (cooler, valve, pump and accumulator) was quantitatively varied. [12]

Each of the fixed load cycles was divided into 13 time intervals (Figure 14) for which the different features were extracted from the measurements.

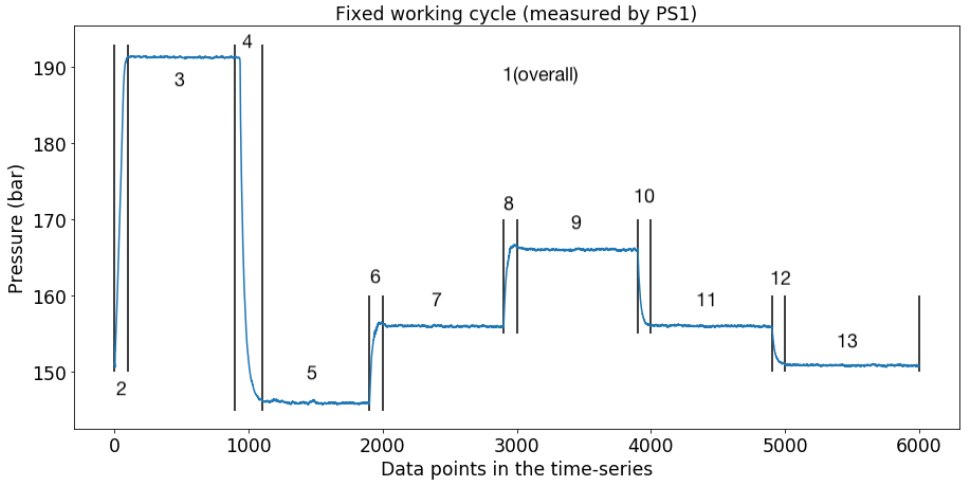


Figure 14. Fixed working cycle with 13 time intervals for feature extraction

The test system then performed several hundred working cycles during which the different fault conditions (types and grades) were simulated in all combinations (Figure 15) again taking the different time scales of the expected effect into account (i.e. reduced cooling has a long time constant while the valve has a short time constant).

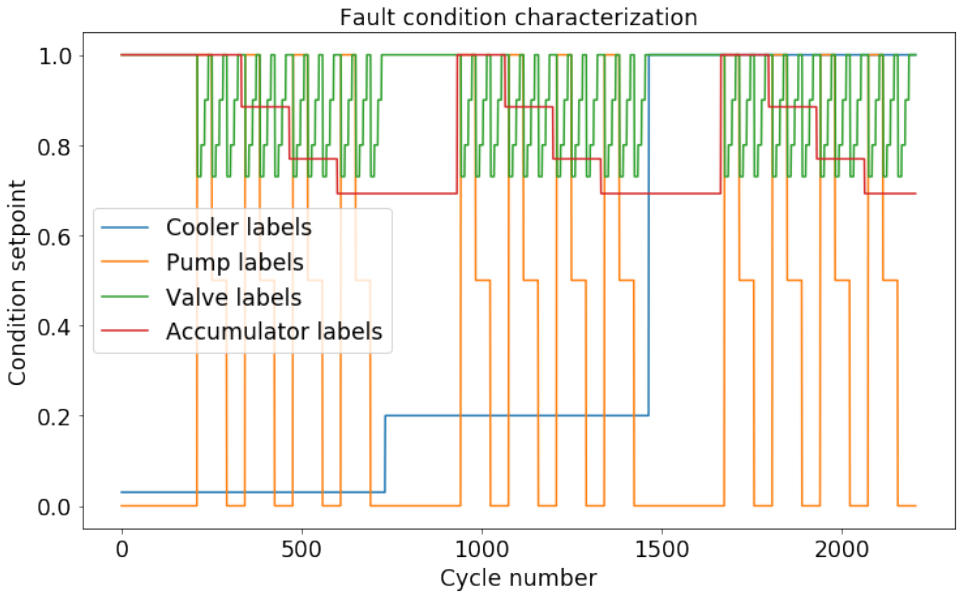


Figure 15. Fault condition characterization

As each sensor had a different sampling frequency, the number of readings per cycle varied from feature to feature. The sensors involved in this dataset were:

Sensor	Physical quantity	Unit	Sampling rate
PS1-PS6	Pressure	bar	100 Hz
EPS1	Motor power	Watt	100 Hz
FS1-FS2	Volume flow	l/min	10 Hz
TS1-TS4	Temperature	°C	1 Hz
VS1	Vibration	mm/s	1 Hz
CE	Cooling efficiency (virtual)	%	1 Hz
CP	Cooling power (virtual)	kW	1 Hz
SE	Efficiency factor	%	1 Hz

Table 5. Attributes of the dataset

There are 17 attributes in total. There was no possibility of doing feature selection because we were using unsupervised learning. Feature selection would have allowed us to select the most important features for every component and not corrupt the data with features that don't give information about the state of the component.

With these attributes the objective is to distinguish between the states of the different parts of the hydraulic system. A diagram of the composition and layout of the hydraulic test rig is shown in Figure 16

The components that were tested for their fault conditions are the following:

3.2 Preprocessing the dataset

Each file contained the signal of each of the sensors. These captured 60 seconds of data at various sampling rates. In order to make sense of the data, a feature extraction was done with the raw data coming from the sensors. Each run of 60 seconds was characterized by these parameters: variance, median, maximum, minimum, kurtosis and skewness. The result of calculating these

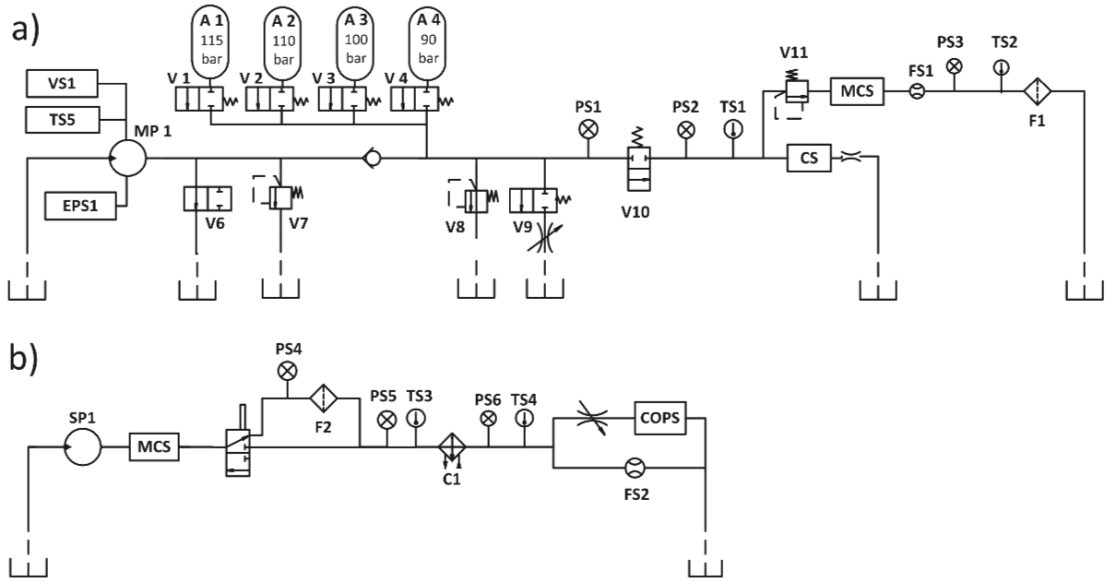


Figure 16. Diagram and layout of the hydraulic test rig. a) working circuit with main pump MP1, b) cooling and filtration circuit with cooler C1

Component	Condition	Control parameter	Possible range
Cooler (C1)	Cooling power	Fan duty cycle	0-100% (0.6-22kW)
Valve (V10)	Switching char.	Control current	0-100% of nom. current
Pump (MP1)	Internal leakage	Bypass orifices	3X0.2mm 3X0.25mm
Acc. (A1-A4)	Gas leakage	Precharge pressures	90,100,115,130 bar

Table 6. Components and their fault conditions

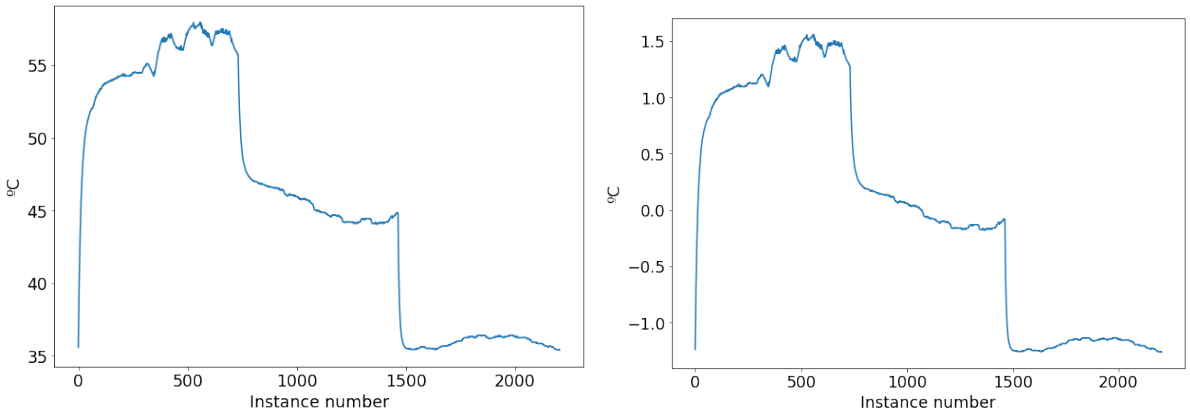
metrics for every time interval in the 17 features was that the new data set had 1230 features. The new data set had a size of 2205 data points (or minutes) by 1230 features. There were no missing values or categorical features.

When preprocessing the data set before clustering, it is recommended to standardize the data. This way, equal importance is assigned to every feature and only the variance explained by every feature is taken into account for the dimension reduction and clustering. This was achieved by eliminating the mean and standard deviation of the data set. The resulting data set had a mean of 0 and a standard deviation of 1. This method eliminated the information that might have been contained in outliers in the data. Had we wanted to keep the outliers information in the data set there was another way of standardizing the data set that involved using the median instead of the mean and the variance instead of the standard deviation. For this data set we used the mean and standard deviation.

The difference between the data before and after the scaling is as follows:

On Figure 17, while the range of the figure on the left is [35,60], the range of the figure on the right is [-1,1.5]. The shape of the curve is the same so the information obtained by it is the same.

The next step was to reduce the amount of dimensions in order to then cluster and get more information from the structure of the data. There are several methods for reducing the number of dimensions. For this data set in particular, getting features that were not correlated could have been useful to the clustering algorithms.



(a) Raw data **(b)** Scaled data

Figure 17. Effects of scaling the data on the sensor TS1

3.3 Dimension reduction

3.3.1 PCA

To start, PCA was chosen to be the simplest, first method. One of the objectives of PCA is to be able to represent systems with many features in two or three dimensions. This made it much easier to visualize the possible states of the machine or other uncommon machine behaviour, otherwise impossible to see if one were to look at the data coming from the sensors. The result of transforming the features into principal component is the following:

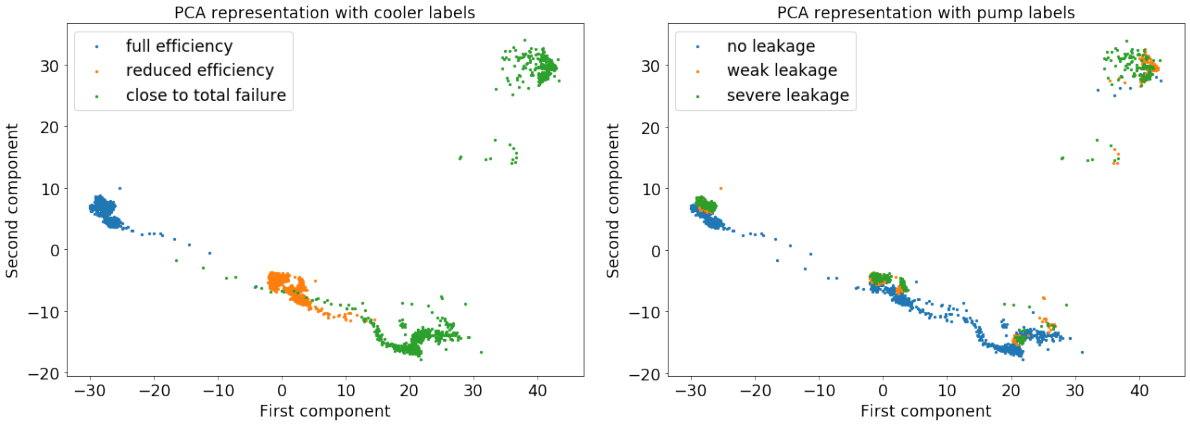


Figure 18. PCA representation of the data set coloured according to the cooler (left) and pump (right) labels

On the left plot of Figure 18, three different states of the cooler C1 are represented with its first two principal components. On the right plot of Figure 18, the different states of the pump MP1 are represented. Figure 18 showed us that there was a clear trajectory from 'full efficiency' to 'reduced efficiency' and finally 'close to total failure' of the cooler component but not for the pump. The pump states can not be distinguished by their cluster because every cluster has all three different states.

One problem clearly seen was that there is an overlap between the different states of the pump. This could have been because the features directly proportional to the PC's variance were important to the states of the pump. In an unsupervised scenario, there was no possible way of reducing and selecting the features most important for the distinction of the three pump states.

Other aspect taken into account was the precision with which the PCA was able to represent the data set. PCA is a method where the attributes of the data set are transformed into n principal components. In this case, we chose to keep only 2 components. The sum of the variance explained by the n components is exactly 100% but keeping all of the components wouldn't have helped the visualization of the data and that is why we only kept the first two. Table 7 shows the variance explained from the first two principal components.

Principal Component	Variance explained
First	0.434
Second	0.136
Total	0.570

Table 7. Variance explained

The variance explained by its first two principal components is not enough to consider the PCA representation accurate. In this case, other ways of representing the data in two dimensions had to be studied.

3.3.2 UMAP

As stated before, PCA didn't represent with precision the progression of the different components of the hydraulic testing rig. This is why other methods for dimension reduction were used. Manifold learning was able to better maintain the structure and variance while also reducing the dimensions to the desired amount, unlike with PCA where the amount of variance depends on the number of components kept. Manifold learning has the benefit of being a non-linear transformation of the data unlike PCA which is linear. UMAP maintained the local structure, which means that if points were close in the UMAP embedding then in the real space they were also similar. Therefore, when some points were grouped together forming a cluster in UMAP, they are similar to each other and different from the points in other clusters. One UMAP embedding that represents the data set is shown in Figure 19. Here, the amount of clusters seen was much higher than in the PCA representation in Figure 18 which allowed many more combinations of states of components. This allowed for a more accurate representation of the behaviour of the system.

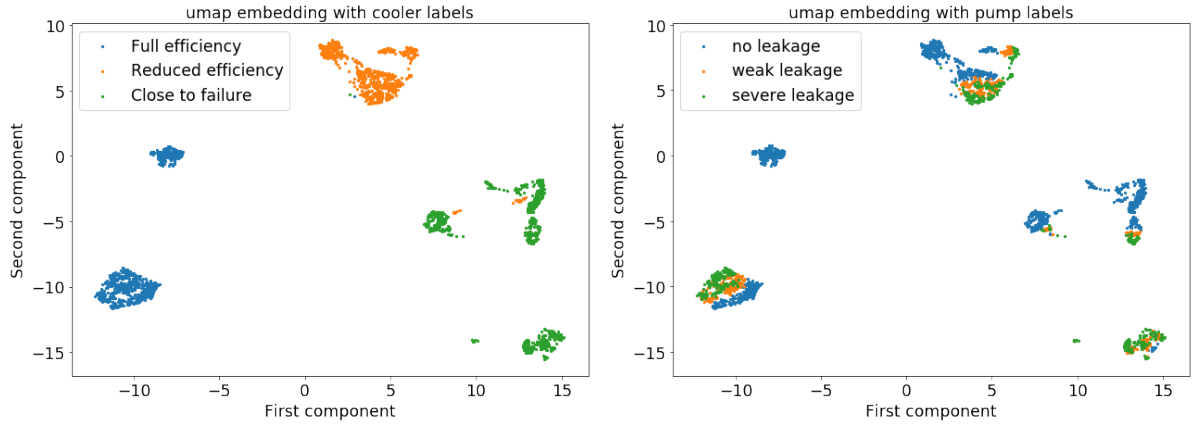


Figure 19. First two UMAP components of the data set

UMAP has its disadvantages with respect to PCA. It has many parameters that the user had to set in order to get the best representation of the data. Not only there is a variability in the parameters chosen by the user, but also in the optimization process a gradient descent algorithm was used, so depending on the random point of initialization, it is possible to get a different result. The only way to make the algorithm deterministic was by always choosing the same random seed for the initialization. In this case, the parameters used to compute the transformation are:

n neighbors	min dist	metric	n components	random seed
50	0.3	correlation	2	42

Table 8. Parameters used for Figure 19

Another use for the UMAP algorithm is to improve the performance of the density-based algorithms like HDBSCAN. This is true because even though the distance between the points in the resulting embedding has no meaning, points that are close forming a cluster are also similar in the real space. UMAP is designed to maximize the distance between the clusters and the density of the clusters. Therefore, a method like HDBSCAN clusters together points based on the local distances, can benefit greatly from a dimension reduction made by UMAP. In the UMAP embedding optimized for the HDBSCAN clustering the minimum distance parameter chosen was 0, as this increases the density of the clusters. The chosen number of resulting components of the embedding was 50 as this data was only used to assigning cluster labels to the points. Then, to visualize this cluster labels we used the embedding with 2 dimensions. The parameters used for the clustering-optimized embedding are shown in Table 9:

n neighbors	min dist	metric	n components
100	0	correlation	50

Table 9. Parameters used for the UMAP embedding optimized for clustering

3.3.3 t-SNE

As with UMAP, t-SNE is a non-linear dimension reduction method that can produce two-dimensional embeddings that maintain the structure of the original data set. The process it is based on is the Stochastic Neighbor Embedding. t-SNE is considered to be slow to compute with data with a high number of dimensions or high number of points but there have been advancements that use approximations that are simpler and less computer intensive. One example of the t-SNE embedding using this dataset is shown in Figure 20. The value of the different parameters used in order to get this embedding are shown in Table 10.

perplexity	exaggeration	iterations	momentum (early)	momentum (opt.)
100	12	550	0.5	0.8

Table 10. Parameters of the t-SNE embedding

When comparing the t-SNE embedding with the UMAP embedding there were no big differences in global or local structure. They both maintained the separation between the cooler states, which a simple K-means algorithm could distinguish with 3 as the number of clusters parameter. They also shared the 8-cluster structure, instead of the 3 clusters visible with the PCA. This helped distinguish between the states of the system when combining the states of all 4 components. The distance between the data points in these embeddings is not related to

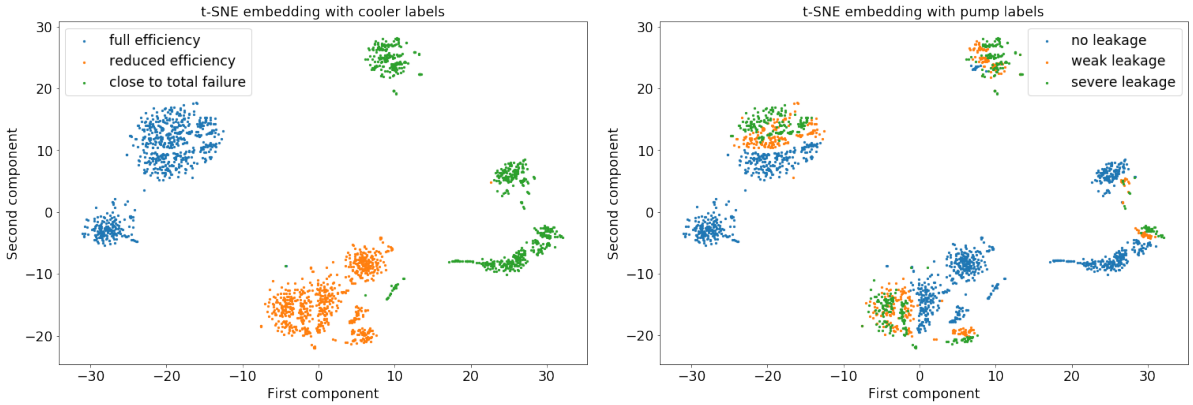


Figure 20. t-SNE embedding of the dataset with the cooler and pump labels

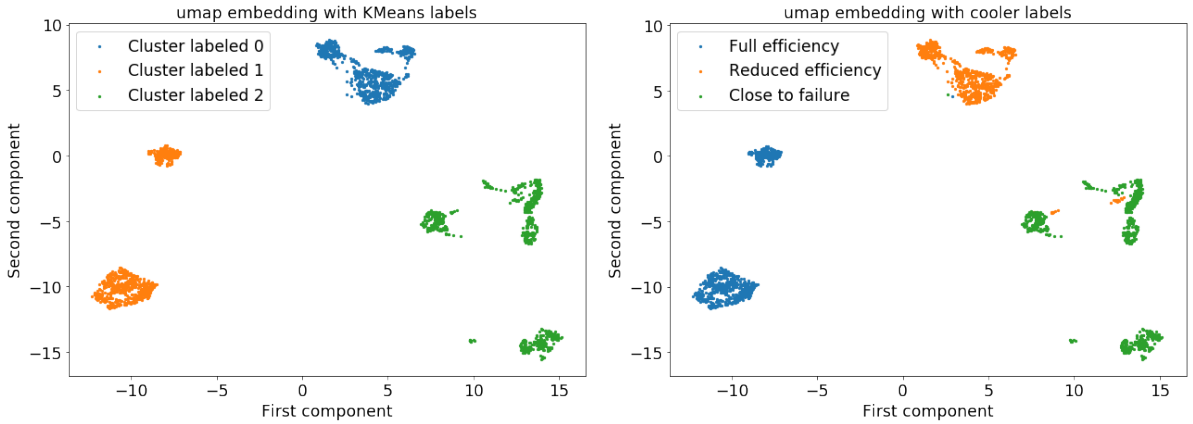
any physical feature. The main aspect is the separation between the clusters and the number of clusters.

3.4 Clustering

After having preprocessed the data, the next step was to get information about the data with clustering in order to discern between the normal conditions and failure conditions. Our objective was to get an optimal amount of points marked as failure, not too many as this would mean an excessive cost to change parts that work well and not too little because this would mean that there could not be enough time to plan to make the maintenance. For this there are various algorithms as previously discussed. First, a simple K-means clustering was done.

3.4.1 K-means on UMAP embedding with cooler labels

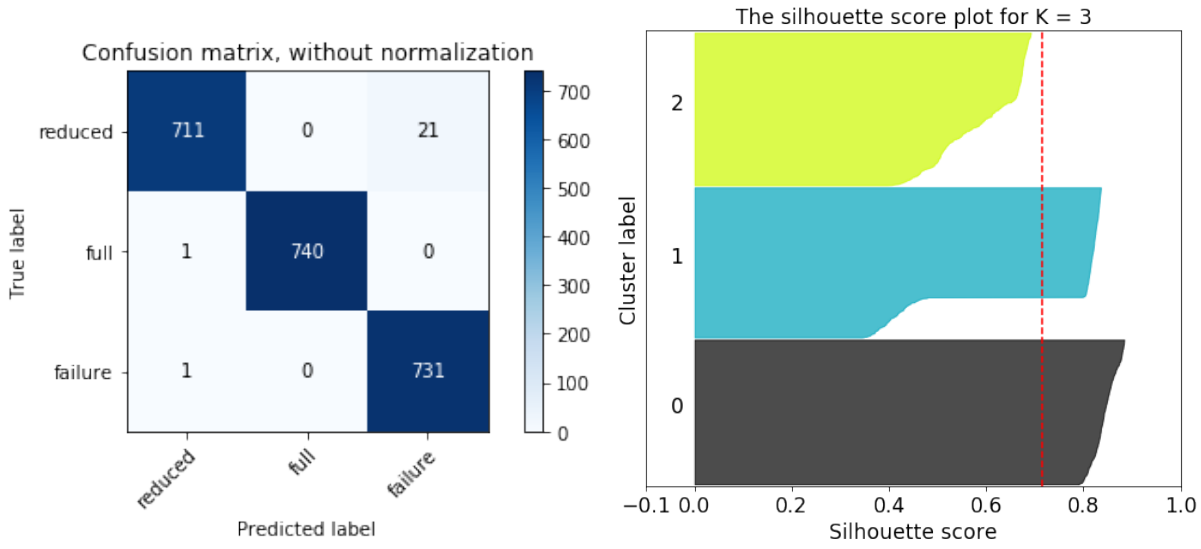
K-means clustering is a clustering method that looks for oval-shaped clusters. It doesn't take into account the density of the clusters, just the cross-entropy or inertia. The UMAP developers don't recommend using their method before clustering with K-means as "UMAP does not always produce clean spherical clusters". This is clearly seen in the left plot of Figure 21. K-means clustered together groups of points with very different densities inside. This could be solved with a higher parameter K in the algorithm.



(a) Clustering with K-means **(b)** Embedding with cooler labels

Figure 21. K-means with UMAP embedding

After a visual inspection of the clusters produced by K-means, we did a more objective measurement of the performance. We started with a silhouette analysis (Section 2.5), which is compatible with unsupervised learning methods (it doesn't need the true labels). On the left side of Figure22 is the confusion matrix, which compares true labels with labels predicted by the K-means algorithm. This allowed us to know how many false positives or false negatives there were.



(a) Confusion matrix **(b)** Silhouette analysis

Figure 22. Performance analysis of K-means clustering with 3 clusters

Other performance metrics are the accuracy score and the adjusted rand score, which both require knowing the true labels, like in the confusion matrix. The results of calculating these metrics are shown in Table 11

Accuracy score	Adjusted Rand Score	Mean silhouette score
0.989	0.969	0.715

Table 11. Performance metric results on K-means using only cooler labels

The conclusion extracted from Figure 22 was that K-means had great performance when just discerning between the cooler states of full efficiency, reduced efficiency and close to failure (from the confusion matrix) but when looking at the silhouette analysis, we saw that cluster labelled 2 didn't have any point with a score higher than the mean silhouette score. This meant that the number of clusters selected for the clustering of the data set was not adequate (as one can see when looking at Figure 21).

Even though K-means had great results when clustering three different states of the cooler, it was not able to know the state of other components in the hydraulic testing rig like the valve V10, the accumulator and the pump MP1. In Figure 23, each component has a different distribution for their respective states.

3.4.2 K-means on UMAP embedding with user generated labels

In order to identify the states of each of the 4 components at any given time, a cluster had to exist per combination of states for every component (which would make 144 states and therefore clusters). This was not the case with the embeddings made by UMAP. UMAP only produced 8

clear clusters that could be distinguished by the clustering method. In Figure 23, we could see that some clusters only had one label per component but there were some clusters that had many points with different labels. This meant that even if the clustering distinguished one cluster from the rest, the information given of that component was non-existent.

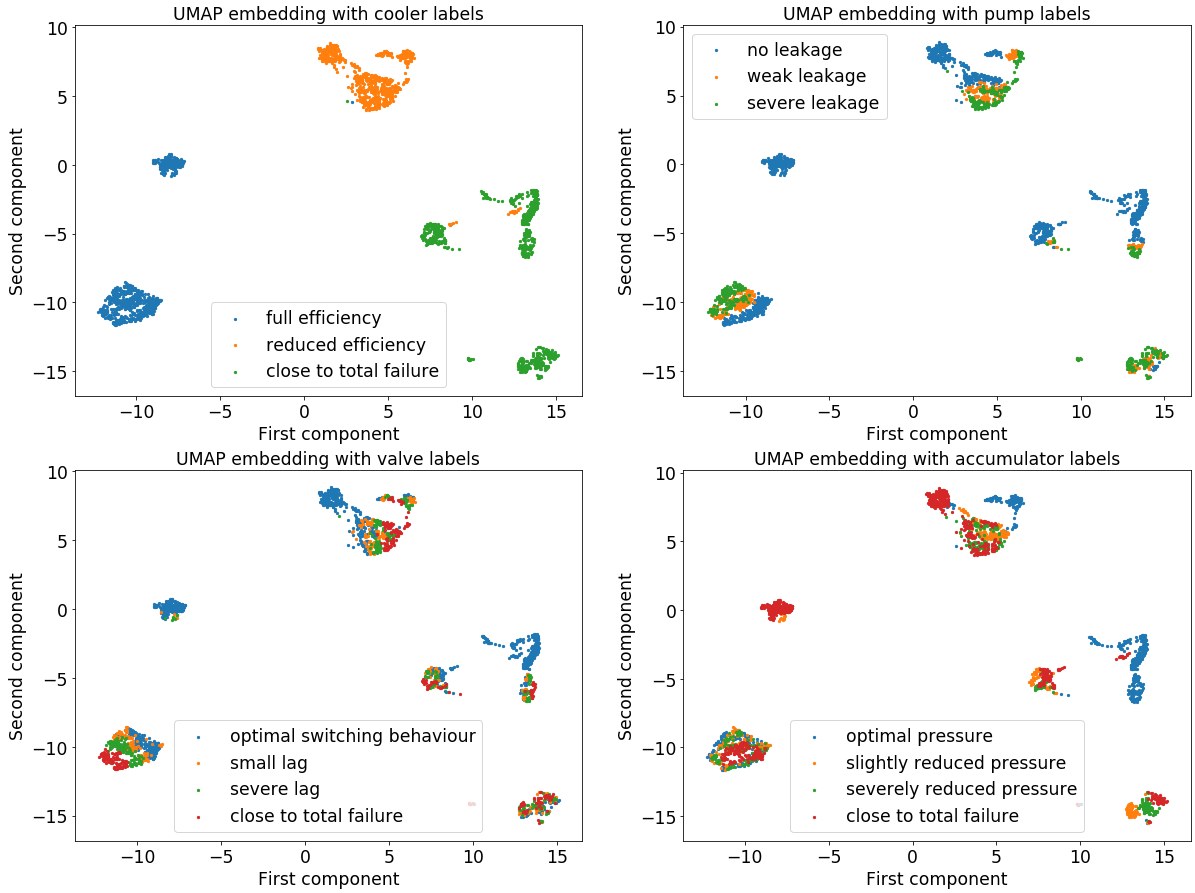


Figure 23. All of the possible states of every component in the system.

Each cluster in the UMAP embedding corresponds to a particular system state with its own set of component states. By combining the states of the different components in each cluster from Figure 23 we created 8 different states shown in Table 12. These states match with the 8 clusters of the UMAP embedding.

Cluster #	Cooler	Valve	Pump	Accumulator
0	Full efficiency	No leakage	Optimal	Failure
1	Full efficiency	-	-	-
2	Close to failure	Weak or severe leak.	-	-
3	Close to failure	No leakage	-	Optimal pres.
4	Close to failure	No leakage	-	-
5	Reduced efficiency	No leakage	Optimal	Failure
6	Reduced efficiency	Weak or severe leak.	-	-
7	Reduced efficiency	-	-	Optimal pres.

Table 12. List of states for the different clusters

In cluster 1, the cooler was known to be in the state of "full efficiency", but the rest could be in any of the 3 or 4 states, from functioning "full efficiency" to being "close to failure". This is represented with a [-] in Table 12. This situation is not ideal, but it is the best that could be done in an unsupervised scenario with these methods, as we could not know which of the features affect the most each of the components, affecting the clustering methods ability to distinguish between states. From looking at Table 12 the states of the cooler could be known with the clustering but not the rest of the components, this is because the features selected to represent this system are related to the state of the cooler. The visual representation of the labels shown in Table 12 are shown in Figure 24.

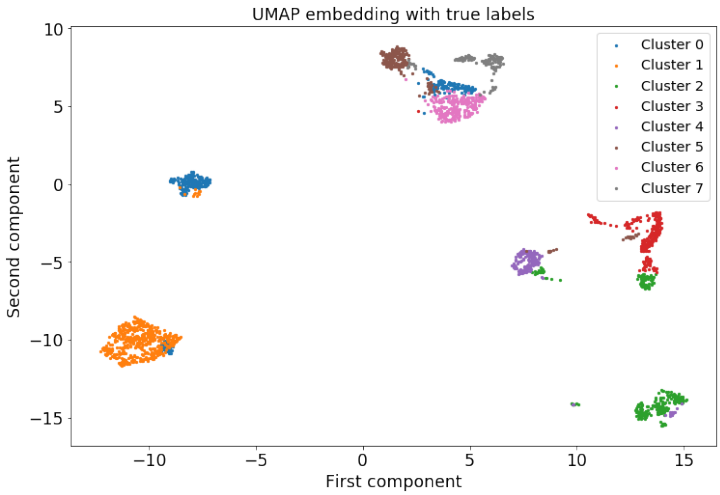


Figure 24. User generated labels represented as a UMAP embedding

With these labels as the true labels to then compare with the performance metrics, we checked if the K-means clustering was able to distinguish between the clusters. Changing the parameter number of clusters to 8 in K-means gave us the result shown in Figure 25.

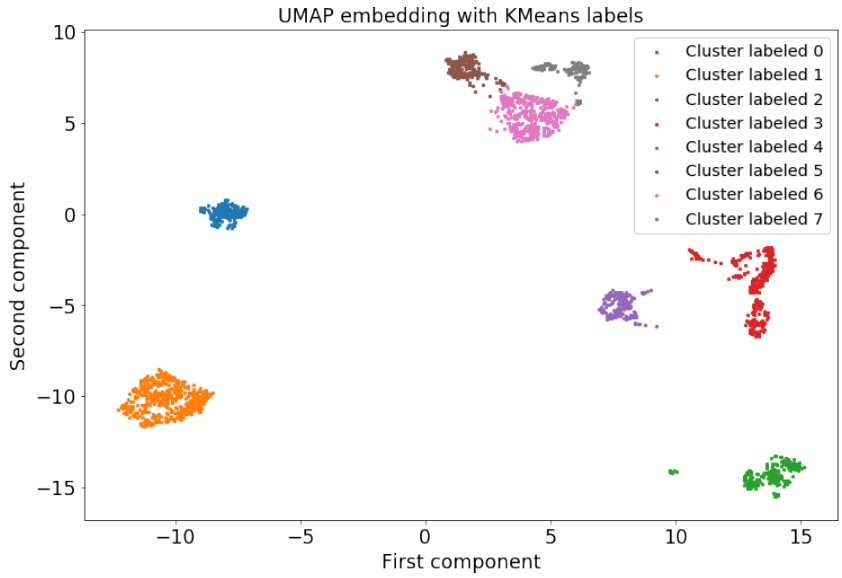
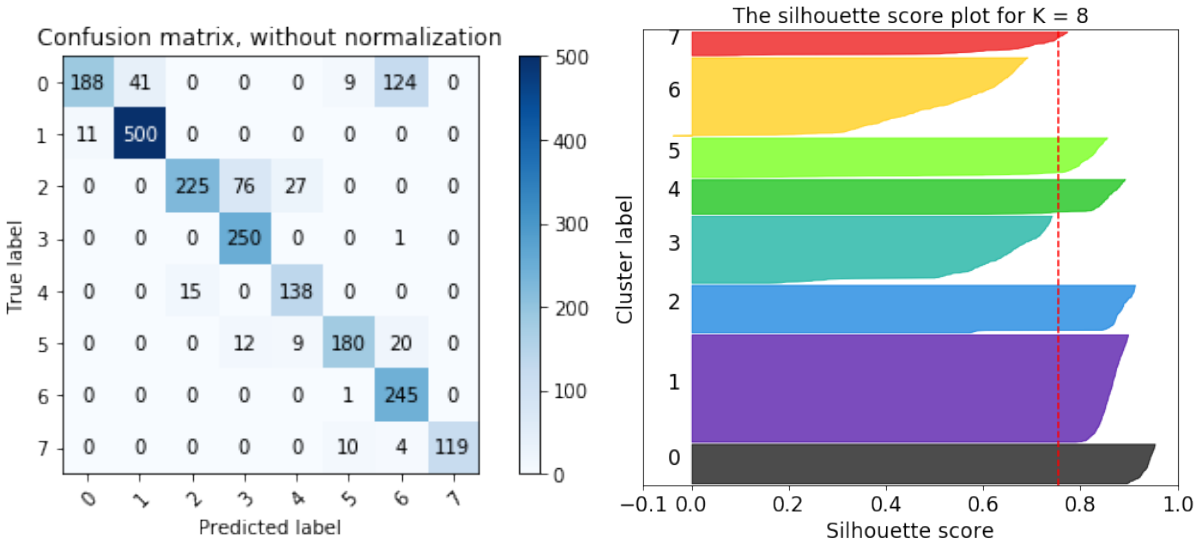


Figure 25. K-means clustering with UMAP embedding

The next step was to calculate the performance parameters. The results of calculating the performance scores is shown in Table 13.



(a) Confusion matrix **(b)** Silhouette analysis

Figure 26. Performance analysis of the K-means clustering with 8 clusters

Accuracy score	Adjusted Rand Score	Mean silhouette score
0.836	0.709	0.754

Table 13. Results of the SA, AS and ARI on K-means with UMAP

As we have said before, K-means is a simple and limited clustering algorithm so it may not be able to produce the desired clusters. In order to get better results we are now going to try the HDBSCAN algorithm to cluster the embedding produced by UMAP.

3.4.3 HDBSCAN on UMAP

HDBSCAN clustering method solves the problem of choosing a number of clusters and focuses on the density of the clusters. The parameters to choose are the minimum number of points in a cluster and the minimum number of samples, which tells the algorithm the minimum density a cluster must have in order to be considered one. We chose both parameters, but by default, the algorithm selects both numbers to be identical. This, combined with the multiple ways to visualize the clustering like the condensed tree, make HDBSCAN a better method for unsupervised learning clustering than K-means.

The computation of the clusters was made using the embedding specifically made for density-based clustering. This embedding has 50 dimensions which explain more precisely the dataset and uses a minimum distance of 0, helping with the density clustering as clusters are more dense. This embedding was not used for the visualization of the data as it would have not been possible to represent more than two dimensions. The algorithm was configured with the following parameters: minimum samples = 10, minimum cluster size = 150, random state = 42.

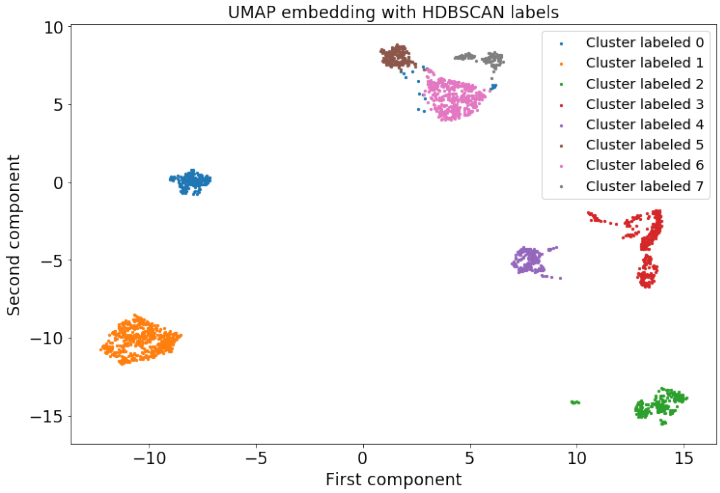
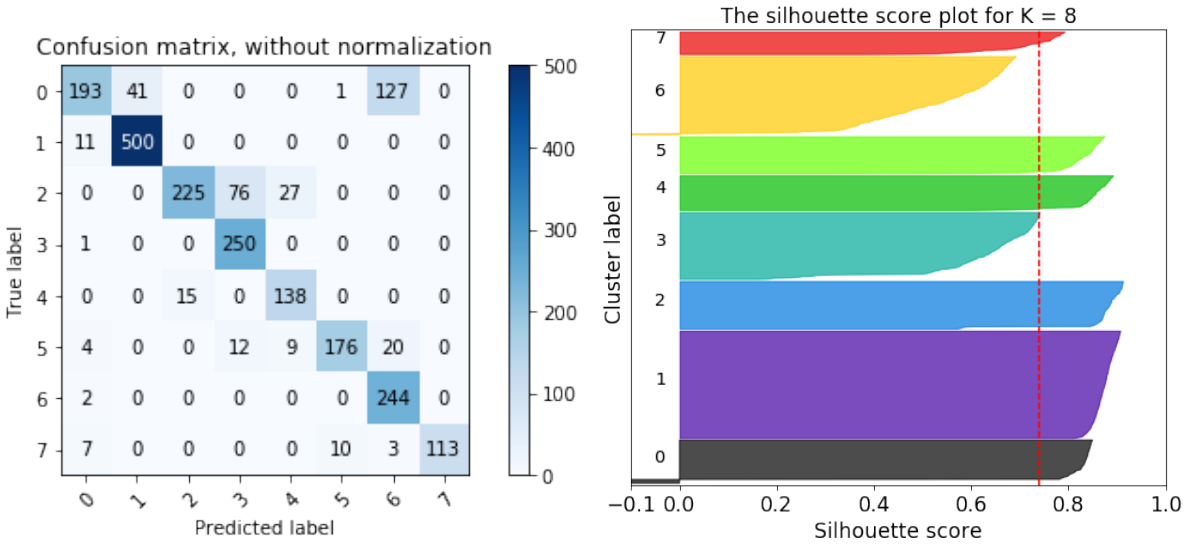


Figure 27. Clustering with HDBSCAN the embedding produced by UMAP

Comparing the HDBSCAN clustering with the one done by K-means, it was clear that they are very similar in appearance but the way they achieved both results was very different. While with K-means, the only thing we had to do is select the number of clusters, we had to optimize the parameters in HDBSCAN to fit the data. In a normal unsupervised learning scenario the labels would not be known and the number of clusters would not be as straight forward as in this case. In this case, K-means was the winner in terms of usability.



(a) Confusion matrix **(b)** Silhouette analysis

Figure 28. Performance analysis of the HDBSCAN clustering

The result of calculating the performance metrics for the HDBSCAN clustering on the UMAP embedding are shown in Table 14.

Accuracy score	Adjusted Rand Score	Mean silhouette score
0.834	0.707	0.741

Table 14. Results of the AS, SA and ARI on HDBSCAN with UMAP

Next step was to cluster the t-SNE embedding to see if the results were better than with UMAP. The tests were done to compare the performance with 8 clusters (not for the cooler labels) as it was seen that the K-means clustering with UMAP does a good enough job and it was tested that the difference was not big enough and therefore it was not of interest to show this test.

3.4.4 K-Means on a t-SNE embedding

In this section, we studied the behaviour of the same methods with a different embedding of the data, in this case t-SNE. The visual representation of the user-generated labels from Table 12 is shown Figure 29

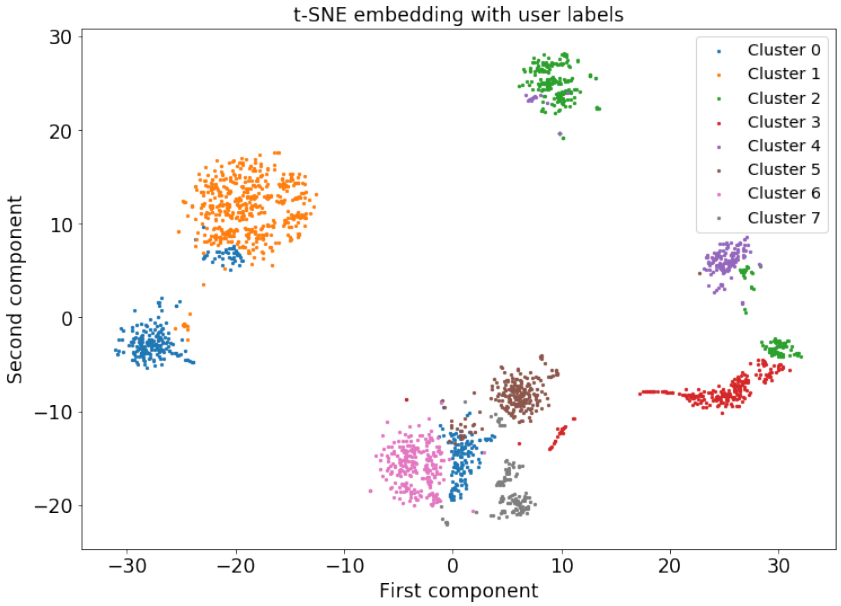


Figure 29. User-generated labels represented with a t-SNE embedding

With this information as input to the K-means method with the 'number of clusters' parameter set to 8 the resulting clustering is shown in Figure 30.

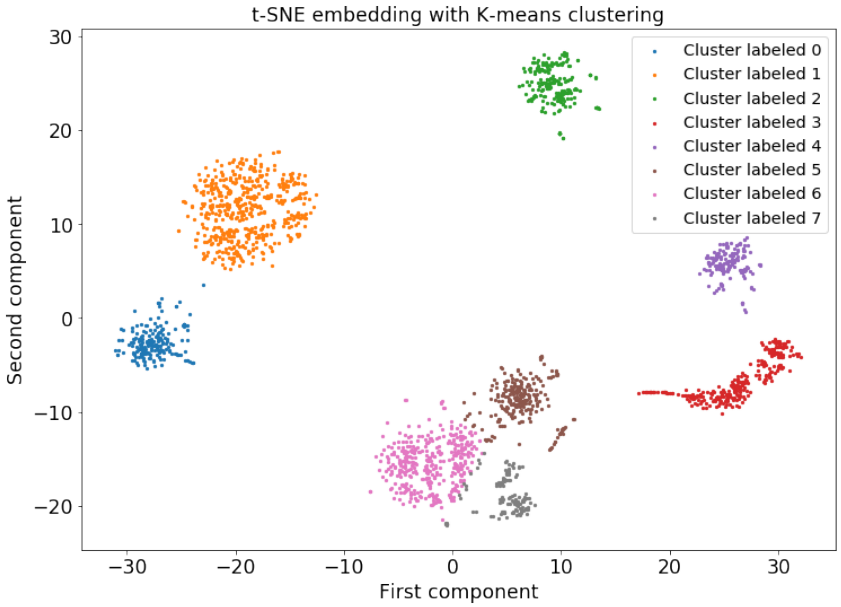
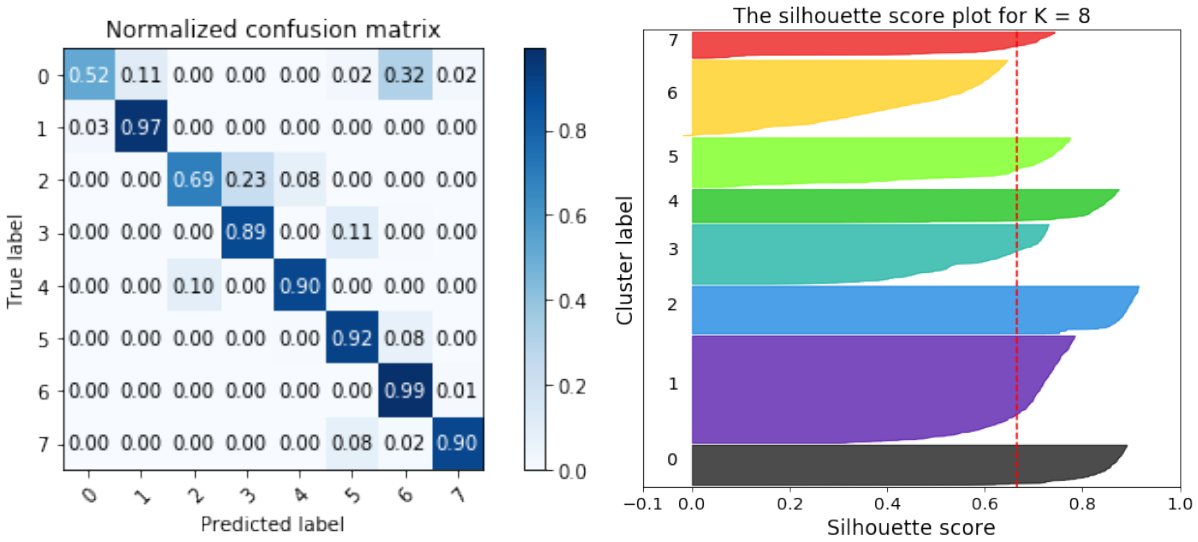


Figure 30. K-means with t-SNE embedding

The results of doing a silhouette analysis and calculating the confusion matrix with the labels from Figure 29 is shown in Figure 31.



(a) Confusion matrix **(b)** Silhouette analysis

Figure 31. Performance analysis of the K-means clustering with 8 clusters in the t-SNE embedding

Next, we calculated the accuracy score, adjusted rand score and mean silhouette score, again, using the labels information from Table 12.

Accuracy score	Adjusted Rand Score	Mean silhouette score
0.834	0.707	0.665

Table 15. Results of the SA, AS and ARI on K-means with t-SNE

Table 15 and Figure 31 show that using t-SNE for the dimension reduction results in the same performance in the supervised learning side (ARI, AS and confusion matrix). K-means clustered in a similar way the points to the true labels and they have the same number of mismatch. On the unsupervised learning side (SA), when looking at the silhouette score, the mean silhouette score is lower with t-SNE but cluster 0 has fewer points with a negative score and cluster 3 has points with a higher score than the mean silhouette score.

3.4.5 HDBSCAN on a t-SNE embedding

In this section, we studied the behaviour of using the t-SNE embedding with HDBSCAN clustering.

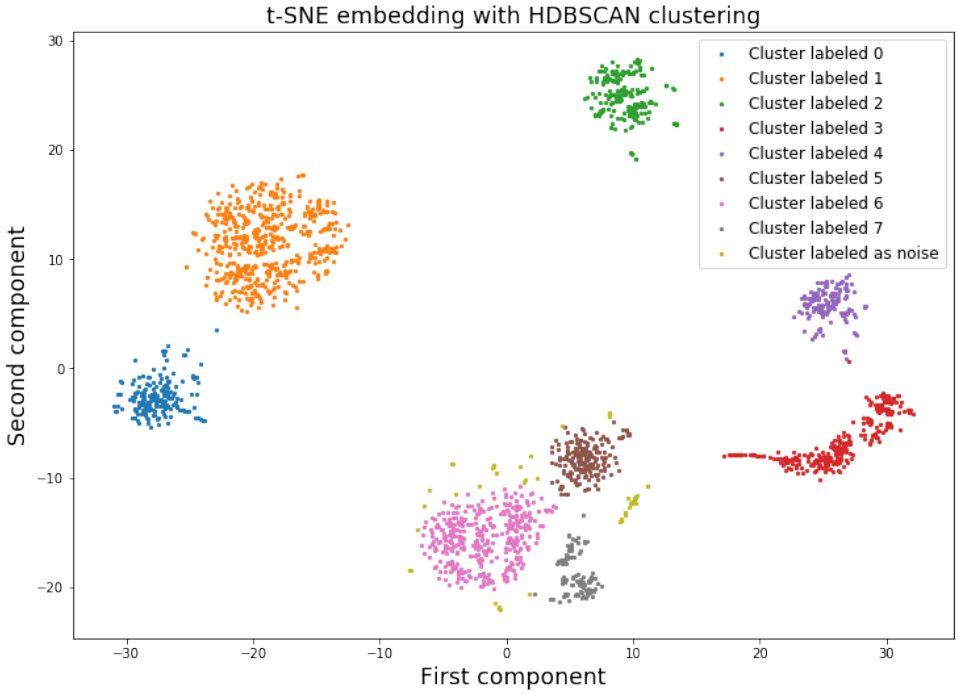
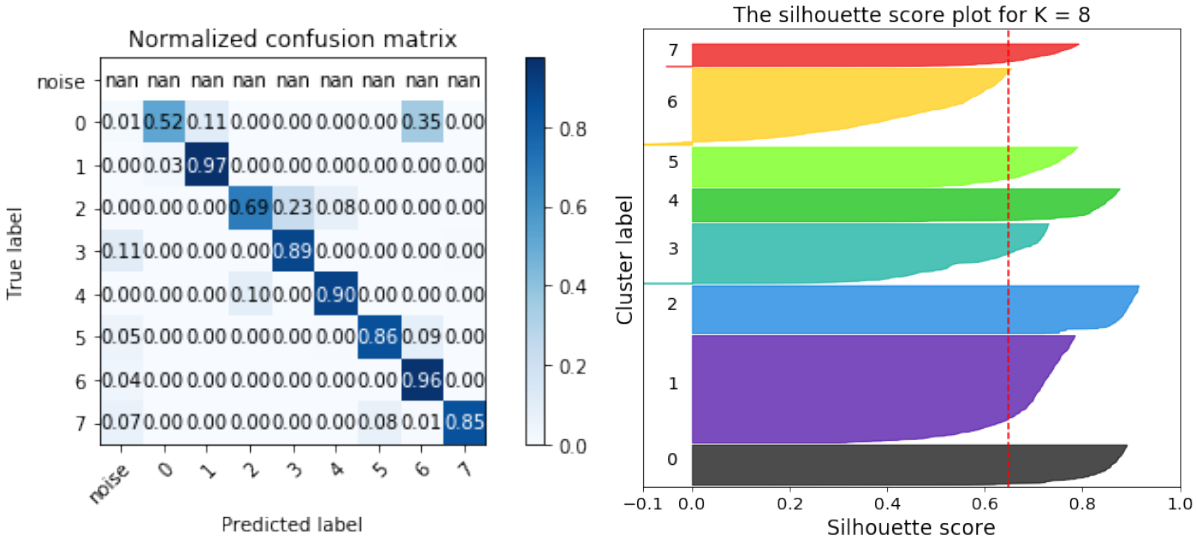


Figure 32. HDBSCAN with t-SNE embedding



(a) Confusion matrix **(b)** Silhouette analysis

Figure 33. Performance analysis of the HDBSCAN clustering with 8 clusters in the t-SNE embedding

Accuracy score	Adjusted Rand Score	Mean silhouette score
0.821	0.707	0.649

Table 16. Results of the SA, AS and ARI on HDBSCAN with t-SNE

Table 16 and in Figure 33 show that HDBSCAN had similar performance to K-means and that the difference between them is due effect of the points labeled as noise. On the unsupervised side, when looking at the silhouette score, the mean silhouette score is lower with HDBSCAN and

more clusters have a negative score. However, cluster 6 has points with a higher score than the mean silhouette score which indicates the clustering is better than the one done by K-means.The fact that HDBSCAN labeled points as noise is good for the analysis but the performance is affected negatively. The confusion matrix in Figure 33 shows a high number of points labeled as noise by HDBSCAN that are actually clusters 0, 3, 5, 6 and 7. These points are a high percentage of points in cluster 3. The confusion matrix shows a good result for most of the clusters except for cluster 0, where it shows a 48% of mislabeled points, 35% of which labeled as cluster 6.

3.5 Conclusions on the hydraulics data set

Regarding the characterization of damage in one of the components, the combination of manifold learning (more specifically UMAP) and K-means offers great performance with this dataset, due to the relation between the states of the cooler and the grouping of points in clusters.

On the other hand, when looking for damage in all four components at the same time, the method starts to show its weaknesses. The problems were the dimension reduction methods and the data available to us (there are no labels good enough to differentiate between all the possible combination of states of the different components in the system). UMAP and t-SNE were able to separate in clusters some of the states of the components. Therefore we may be able to know at any given point the state of some of the components but not the others. Additionally, there are some states (6 and 0) that share a cluster and because of their proximity the clustering method can not discern between them. This increased the error percentage in the assignment of labels to the points. Comparing the results of t-SNE and UMAP for the dimension reduction (both with HDBSCAN for clustering), the objective metrics showed that both have very similar performance (see Table 17). The only significant difference between the two is the points that belong to state 3, on the t-SNE representation are very close to states 5 and 7 resulting in getting clustered together with cluster 5 by the HDBSCAN algorithm.

HDBSCAN with:	Acc. Score	Ad. Rand Score	Mean silh. score
UMAP	0.834	0.707	0.741
t-SNE	0.821	0.707	0.649
K-means with:			
UMAP	0.836	0.709	0.754
t-SNE	0.834	0.707	0.665

Table 17. Result comparison UMAP vs t-SNE

Regarding the speed with which both embeddings were made, UMAP was significantly faster as Table 18 shows, even with the parallelization of the computations (in 6 cores of a CPU).

Dimension reduction method	Compute time (s)
UMAP	25.5
t-SNE	150.1

Table 18. Speed of dimensionality reduction methods

Another way to compare the results of both dimensionality reduction techniques is a subjective one, the visual separation and understanding of the behaviour of the system. On this regard, both embeddings are very similar, with the only difference that t-SNE is less dense than UMAP and the position of the clusters is different (but as we have said before the position of the data points is meaningless in these embeddings).

When comparing clustering methods, the data showed us that the difference in performance between k-means and HDBSCAN is very small. For example, when comparing UMAP clustered by HDBSCAN vs k-means there is only a difference of 0.002 on the adjusted Rand score. The computing time of both clustering methods is also very similar with the chosen implementations, HDBSCAN was more computer demanding but it was not a deciding factor with this data set. The only meaningful difference between both methods was the ease of use of both methods. While k-means needed the number of expected clusters a priori, HDBSCAN needed the minimum size of the clusters. When applying clustering to unsupervised learning, it was easier to know how big the failure cluster should be than how many failure modes or clusters it had to find. Therefore, HDBSCAN is a better method for unsupervised condition monitoring.

These results on their own are not very promising or useful, but comparing them with when the user didn't have any information at all about the data it had, it is an improvement. The results would be much better if the user had the labels beforehand and did a supervised learning kind of condition monitoring. This would have enabled the selection of the best features for the characterization of the different states of the 4 components.

Chapter 4

Application on Real World Data

4.1 Background of the dataset

This data set consists of data recorded with many different sensors of a complex machine composed of many different parts. The machine's main function is to mill with different heads which can be changed with an automatic head changing system. The machine has embedded sensors that record the state of the machine every second non-stop 24 h and 365 days a week. The data set we used for this thesis contains information from three consecutive months, which makes the data set have a size of approximately 6 million data samples. The number of features is also high. In total, there are 87 features in this dataset. This amount of information has some consequences in the unsupervised learning. The amount of computing power and memory needed is higher than what most computers have. Therefore, new methods had to be created (here, dimensionality reduction and preprocessing play a big role).

In these many features there are some that do not contain any information, either by not having any variance (every value of the feature is the same) or because most of its values are missing. There are numerical and continuous features that have no missing values and have information, but they do not give information about the state of failure of the system.

In this dataset there are continuous and categorical features. Most of the categorical features are binary, representing, for example, if one alarm inside the computer was activated. Either these categorical features have to be transformed into binary features with the methods explained in Section 2.2.3 or they have to be eliminated from the data set. On this thesis it has been decided that these categorical features had to remain as they contain valuable information to determine the state of the system and this is the objective of the paper. One example of a continuous numerical feature from the dataset is shown in Figure 34

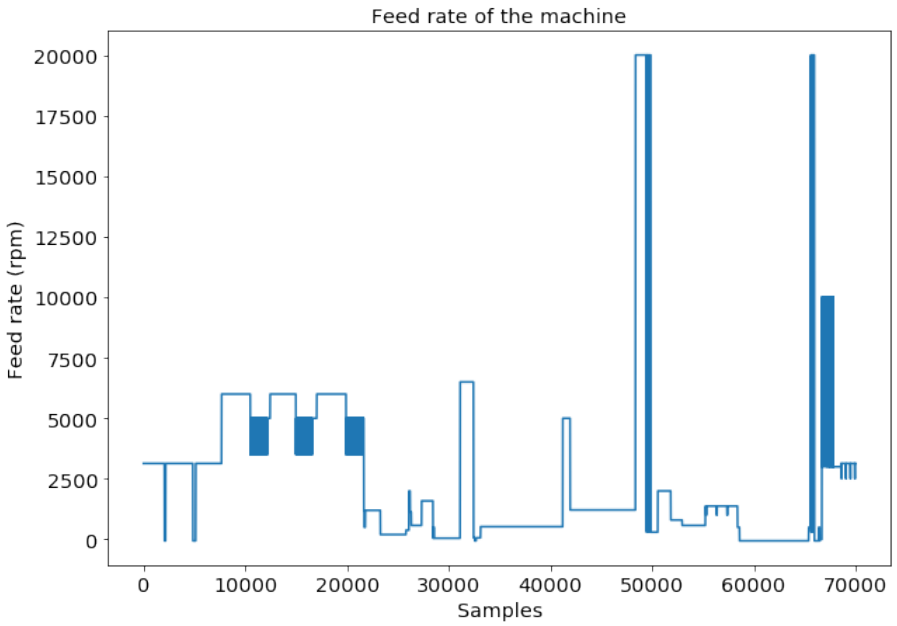


Figure 34. Example of 2000 points from one feature of the dataset.

Even though this paper is focused on unsupervised learning, which requires the user to not know beforehand the labels of the data points, some data of the state of the system is known. The nuance is that the information known is not accurate enough. The information accessible to us only has a precision of between one hour and one day. In order to label the data points the precision has to be of a second, based on the sample rate of the sensors. This information served when selecting a slice of the data set, based on where the system was known to have a failure and where the system was repaired.

The first step when doing unsupervised machine learning is always understanding the data and preprocessing it, after doing the first part, we then preprocessed the data to prepare it for the dimension reduction and clustering.

4.2 Preprocessing the dataset

To start, the data set was prepared for the dimension reduction and clustering algorithms. PCA and k-means for example, require the features to be all continuous and numerical. Other methods accept a distance matrix as input and therefore the binary and categorical features can be included by precomputing the distance matrix with one metric that takes into account both types of data. When it comes to missing values, and scale, every algorithm either requires or benefits from removing them.

4.2.1 Dealing with categorical features

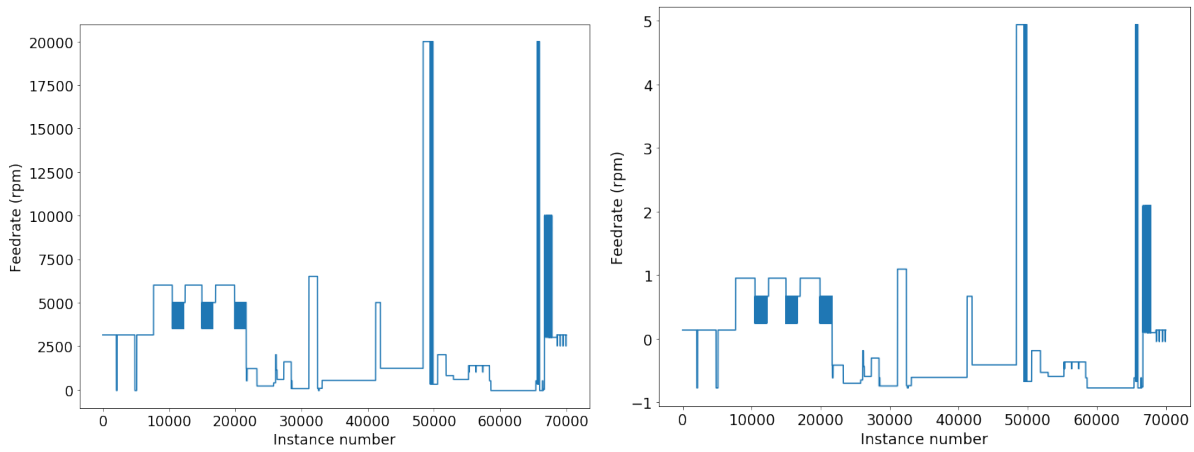
As previously discussed in Section 2.2, when a data set has binary or categorical features but also has continuous features, the usual methods of computing distance between points can't be used (for example, Euclidean distance). To solve this we used the Gower distance. With this distance or dissimilarity, we calculated UMAP and t-SNE embeddings using both continuous and categorical features. PCA only accepts numerical continuous features because it only accepts euclidean distance as its distance metric. All of this means that K-means and PCA only used part of the information available, so the clustering might not have been as accurate as it did not have all the information. When clustering the PCA, UMAP and t-SNE embeddings, the data used as

an input for the method is numerical and continuous and therefore it was possible to calculate the distance with a metric only used for numerical and continuous data.

Another way to include the categorical and binary features into the input data and then transform it with UMAP is to fit an embedding with the continuous numerical data and another embedding with the binary data (each using a specific metric compatible with its data type), then combine both graphs to create an embedding that takes into account both data types.

4.2.2 Scaling the dataset

In this data set there are many features with many different scales. This could have affected the ability of the dimension reduction method to create an accurate representation of reality and the clustering method's ability to compute distances between the points. In order to standardize this data set we used a standardization that used the mean of the original data set to make the new and scaled data set have a mean of zero, the standard deviation to make the new data set have a standard deviation of 1. This, might have eliminated some information that might have been valuable to the dataset such as outliers. If outliers are just noise in the data and do not add much information to the determination of system states, then this was the scaling to choose. A visual representation of the effects of the scaling in the data set can be seen in Figure 35



(a) Raw data **(b)** Scaled data

Figure 35. Comparison between scaled and raw data for a slice of the data set

As you can see the difference between the two plots in Figure 35 is that while the plot on the left (not scaled) has a range of 20000, the plot on the right (scaled) has a range of 5. This was done while preserving the information contained in the data because the shape was maintained.

The next step in this method was to reduce the number of dimensions in order to visualize the data set in two dimensions and be able to extract information from clustering it. We started by using a common dimension reduction method like PCA and then we compared it to two manifold learning methods: t-SNE and the new UMAP.

4.3 Dimension reduction

4.3.1 PCA

This dataset has many features, many of which do not give any information about the states of the system and only disturb PCA's ability to separate the data into clusters. In some cases,

using PCA before doing other more aggressive dimension reduction methods can be a good idea as it can alleviate the computing burden when calculating the pairwise distances in high dimensions. In this case, UMAP does not see any improvement in the computation time because of this reduction in dimensions, as Leland McInnes says "...I would also note that you probably do not need the PCA step. I suspect UMAP will not be that much slower on the full dimensional space. The ultimate limiting factors here are time and RAM....". Something else to note is that PCA is a linear dimensionality reduction technique. It works best with data sets that change linearly and this data set is clearly nonlinear as we can see in Figure 34.

Figure 36 shows a plot of the first two components of the PCA transformation of a small slice of the data set containing information of 2 days of continuous sensor data. We have chose to use this slice of data because it is known to us that there was a problem with one of the electric motors and that there were some vibrations affecting the performance of the machine on day 1. We also knew that the machine was stopped for 11 hours during day 1 and that one part of the machine was replaced and that the next day (day 2) there were no other incidences.

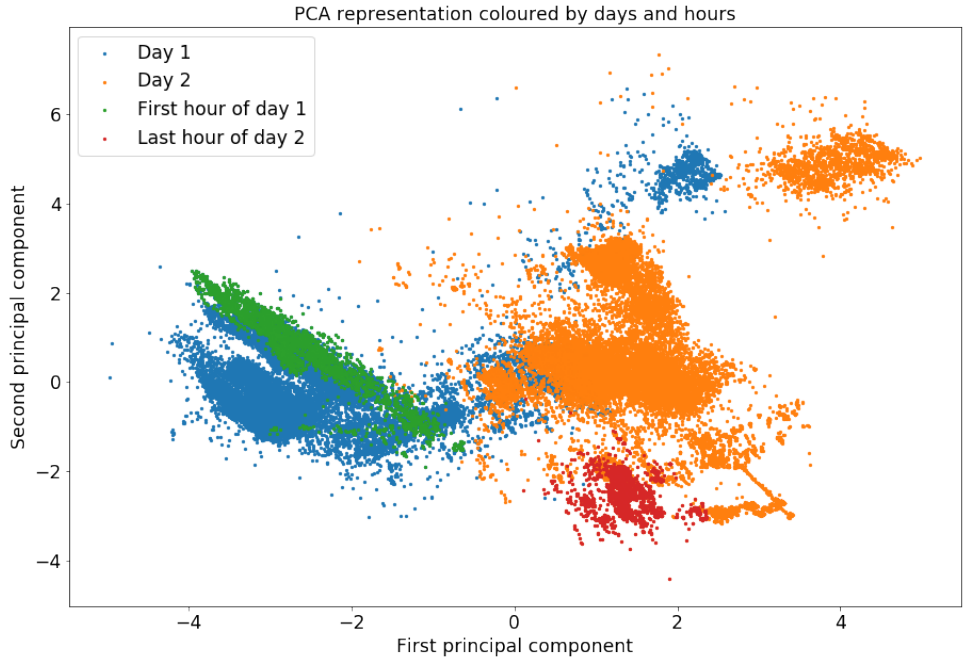


Figure 36. PCA with two days of the dataset

In Figure 36, there is a clear difference between both days, with some overlap in the clusters of points with a value of around 0 in both principal components. Another aspect to point out is that there are two small regions different from the rest of the data on the top right side of the plot. These points happened during the same hour and that could mean that some special incident happened during this hour. In Figure 36 the points corresponding to the first and last hours of the data set were also coloured. This was done in order to check if there were any difference in the data that can distinguish a machine without any incidences and one that had to be repaired. In this plot, there is a progression from left to right (first principal component is causing the variance) from -4 to 2 and from top to bottom (second principal component causing the variance) from 2 to -2. As there is a progression between both hours with points going from the first to the last, even though the data shows there is a difference, a clustering method might have many problems correctly dividing the data set into clusters.

Next we created a new embedding that could better separate the states into better defined clusters so the clustering method could distinguish them.

4.3.2 UMAP

UMAP deals with the problems PCA as a linear dimension reduction method. It is more computer intensive than PCA (100x) [23] and therefore it is not as useful when reducing the computing time of other algorithms we may want to use down the line, clustering for example. In this case, UMAP is a tool to visualize the data and make decisions with the information that a topological representation of the data gives the user. Before, PCA was not able to use all of the data available (continuous and categorical and binary data) and only could use the continuous numerical data. Using UMAP for the dimension reduction of the continuous features we got the following embedding (shown in Figure 37):

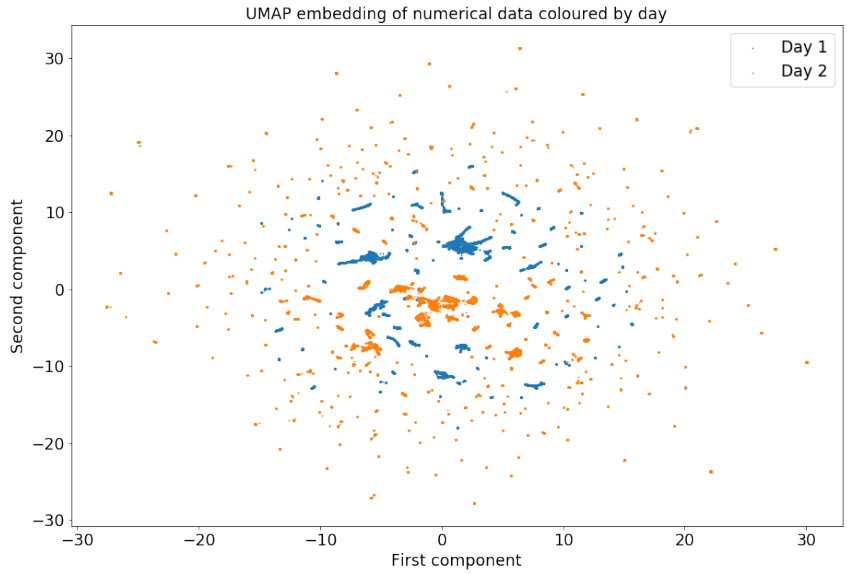


Figure 37. UMAP embedding of numerical data

random seed	metric	min dist	n neighbors	initialization
42	euclidean	0	500	spectral

Table 19. Parameters for the UMAP embedding with numerical data

What we saw is what looked like a random cloud of clusters formed by the local structure of the data but nothing global could be extracted from this embedding. Therefore, we needed more information from this data set in the form of categorical and binary data. In order to make UMAP compatible with the types of data we had, we had two options to choose from. The first was to compute the distances beforehand and then input these distances into UMAP. When computing the distance with the Gower metric we got the embedding shown in Figure 38. The parameters used to create this UMAP embedding are shown in Table 20.

random seed	metric	min dist	n neighbors	initialization
42	precomputed	0	500	random

Table 20. Parameters for the UMAP embedding with Gower distance

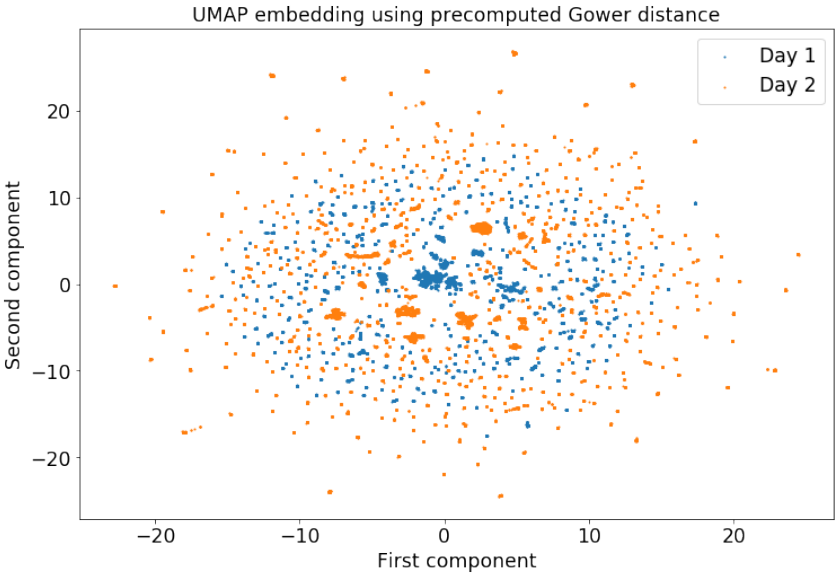


Figure 38. UMAP embedding of data with Gower distance

The result shows that the points are randomly placed. This is due to the random initiation of the algorithm. Normally the algorithm starts by doing an spectral optimization and then initiating from those points but there was an error in the code from the UMAP implementation we were using and we were not able to do so. This way, the algorithm started the optimization from random points and managed to group together points near the regions where the algorithm had started but the global structure was lost in the process. If we were to input this into a clustering algorithm like k-means we would have got a result that would not help in the differentiation of states of failure and normal.

The second way was to fit two graphs, one per data type (each with its own data metric, Euclidean for the numerical and continuous data and Hamming for the binary data) and then combine them giving each one a weight to one of the graphs (in order to give more or less important to one metric or another). After fitting two graphs (one to each type of data type) to then combine them, the result is what is shown in Figure 39. The parameters we used to get this embedding are shown in Table 21. The main parameters, min dist and n components where left to be the default ones for the specific implementation we are using and we have checked that changing these numbers does not change the result in a significant way, like in Figure 40

random seed	metric continuous	metric binary	min dist	n neighbors	weight
42	euclidean	hamming	0.1	15	0.5

Table 21. Parameters for the UMAP embedding of both data types

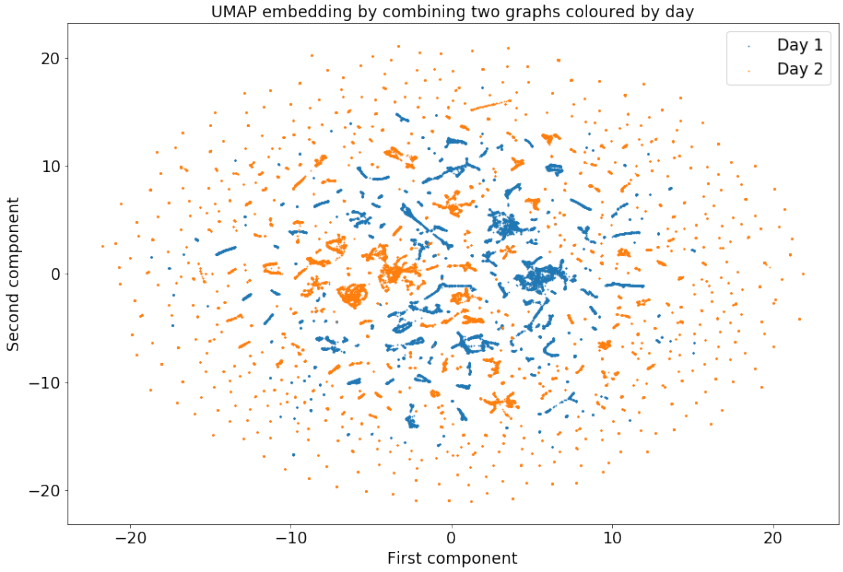


Figure 39. UMAP embedding of real data with second method

The resulting embedding is very similar to the one in Figure 38. The figure shows many little clusters randomly distributed in the space. Days 1 and 2 as whole days are not easy to separate from each other but the points grouped together are from the same day. Changing the number of neighbors parameter should result in clusters that better maintained the global structure rather than just focusing on the local structure. The result of using these parameters for the embedding is shown in Figure 40.

random seed	metric cont.	metric bin.	min dist	n neighbors	weight
42	euclidean	hamming	0	200	0.5

Table 22. Parameters for the UMAP embedding of both data types with more neighbors

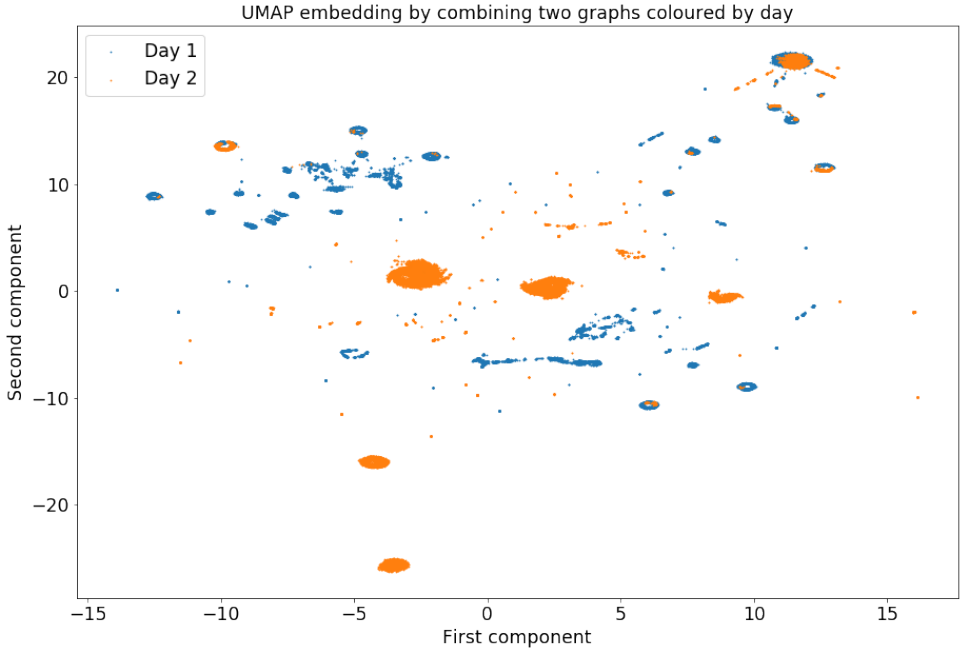


Figure 40. UMAP embedding of real data with other parameters

By increasing the number of neighbors in the UMAP embedding we could see that the clusters became much bigger and the days were much closer together than before. Now, even though the initialization was made randomly, the points found a way to be closer together. With this embedding clustering the days can be easier with HDBSCAN but not with k-means. In section **??** we continued to increase the number of neighbors. We found out that the resulting embedding has its points closer together but the global structure is lost, so we decided to stick with the embedding shown in Figure 39.

4.3.3 t-SNE

In this scenario, where the size of the data set is big, it is very common that some features do not give any information about the state of the system. They only disturb the dimension reduction algorithms and their ability to form clusters representing its possible system states. This is what t-SNE tries to solve, help preserve the local structure while at the same time preserving global structure. We start by using just the numerical and continuous features and avoid including the categorical and binary features. For this slice of the data set, the best result came from the parameters shown in Table 23. The result of doing the dimension reduction with these parameters is shown in Figure 41.

perplexity	exaggeration	random seed
1000	48	42

Table 23. Parameters for the t-SNE embedding of the continuous and numerical features

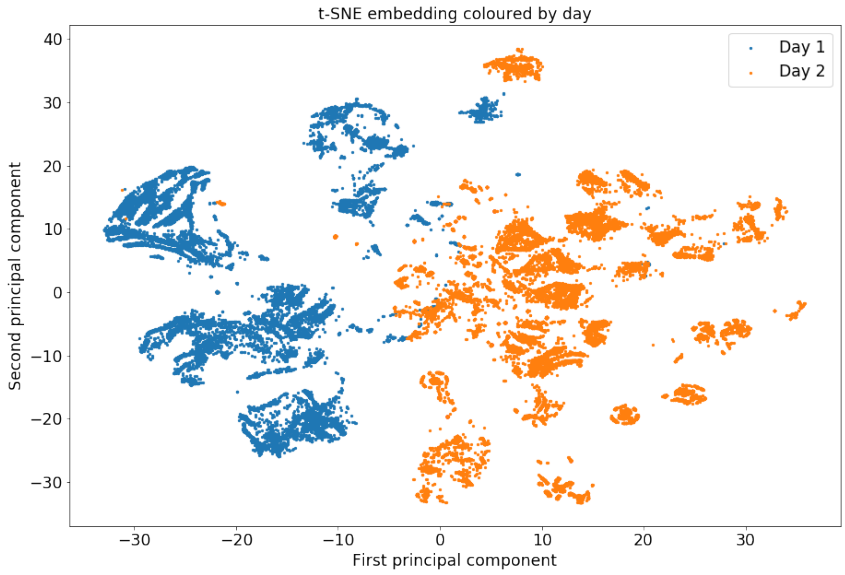


Figure 41. t-SNE embedding of the continuous and numerical features

Even though the result is satisfactory, there are many features with information not being used for the embedding. In this case, the chosen metric is the Gower metric like the one described in 2.2. The parameters chosen for the embedding are shown in Table 24. The resulting embedding when using these parameters is shown in Figure 42. In the embedding, the separation between the days is not as evident as in the embedding done with the data set containing only the numerical and continuous features. Only subjectively speaking, in this case using only the numerical data set has given better results when separating between the two days. Each of the clusters seen

could represent one state of the machine but we were not able to confirm because we didn't have enough information, we didn't have the labels of each data point.

Perplexity	exaggeration	random seed
1000	48	42

Table 24. Parameters for the embedding using the precomputed Gower distance

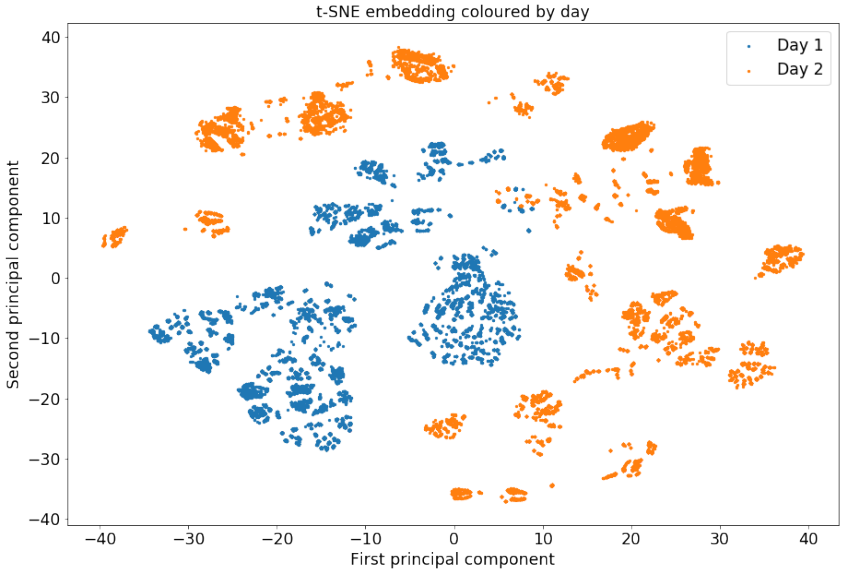


Figure 42. t-SNE embedding using the precomputed distance

After transforming the data set into the three different two-dimension data sets we applied k-means and HDBSCAN clustering methods to them and compared their performance in labeling the points in the different areas we had seen in the review of the embeddings.

4.4 Clustering

4.4.1 K-means with PCA

At first, we tried to get in the clustering of the data set the two cases that we were looking for: 'normal' and 'failure'. We did this by setting the number of clusters parameter to 2. The result is shown in Figure 43.

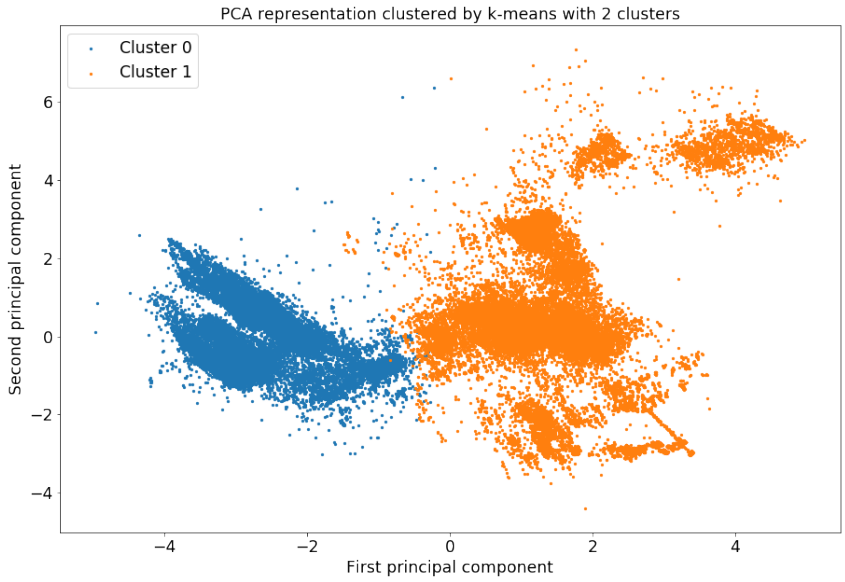


Figure 43. K-means clustering of the PCA representation, with 2 clusters

Comparing the k-means clustering in Figure 43 to the clusters formed by days 1 and 2, there is a clear resemblance. There is an area (on the top right corner) where the algorithm had not been able to cluster the day 1 properly. On the center of the figure, where both the first and second principal components take a value of 0, k-means labeled the points as cluster 1 while on Figure 36 we had seen an overlap between both days.

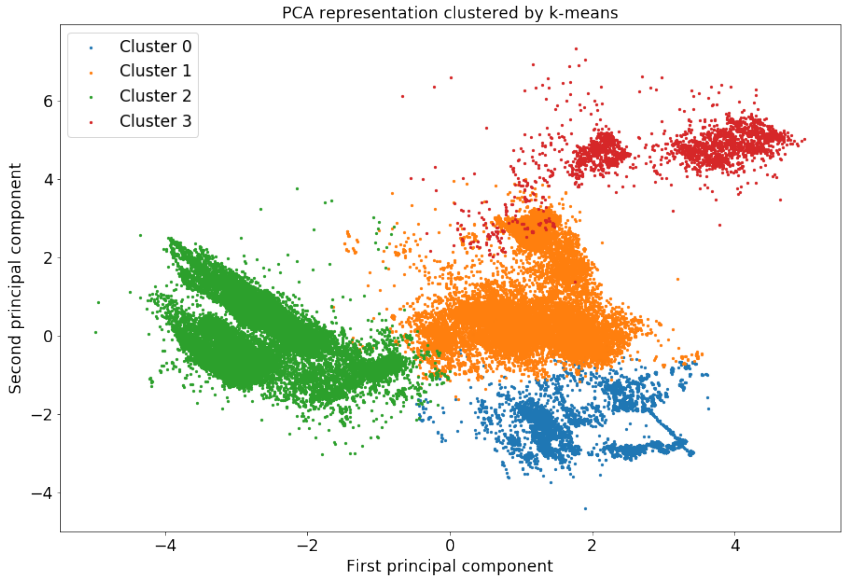


Figure 44. K-means clustering of the PCA representation, with 4 clusters

In order to get k-means to cluster the points at the top right corner as their own cluster we increased the number of clusters to 4. On Figure 44 we saw the algorithm had been able to separate between the different clusters that had appeared in the PCA representation but these clusters were not the clusters that we were looking for which distinguished between the days of failure and days of normal functioning of the machine.

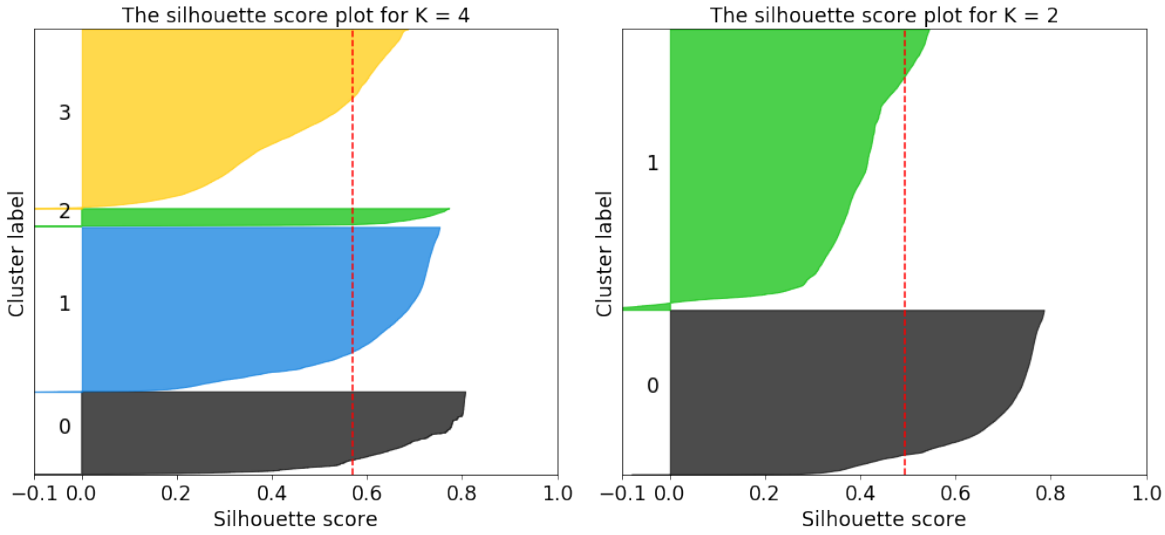


Figure 45. Silhouette analysis of K-means clustering with the PCA representation

k=2	k=4
0.494	0.570

Table 25. Mean silhouette score of both clusterings

K-means is not prepared for situations where the data is in clusters of different size and with noise and randomness. Here, an algorithm that can discern between the areas of high and low density could prove to be more successful. In order to see if the poor performance was a result of the clustering method or the data set used as input we changed the input with the UMAP embedding from section 3.

4.4.2 K-means with UMAP

We started by doing a clustering on the UMAP embedding with a low number of neighbors from Figure 39. The results of this clustering were not satisfactory because k-means is not suited for this kind of data set. The results of this clustering can be seen in the Appendix section **??**.

Afterwards we clustered the embedding with the medium (and optimal) number of neighbors from Figure 40. This embedding should yield better performance in the clustering because the structure shows 5 distinct different areas with space between them, unlike what Figure 39 shows. The result of doing the k-means clustering with this UMAP embedding is shown in Figure 46.

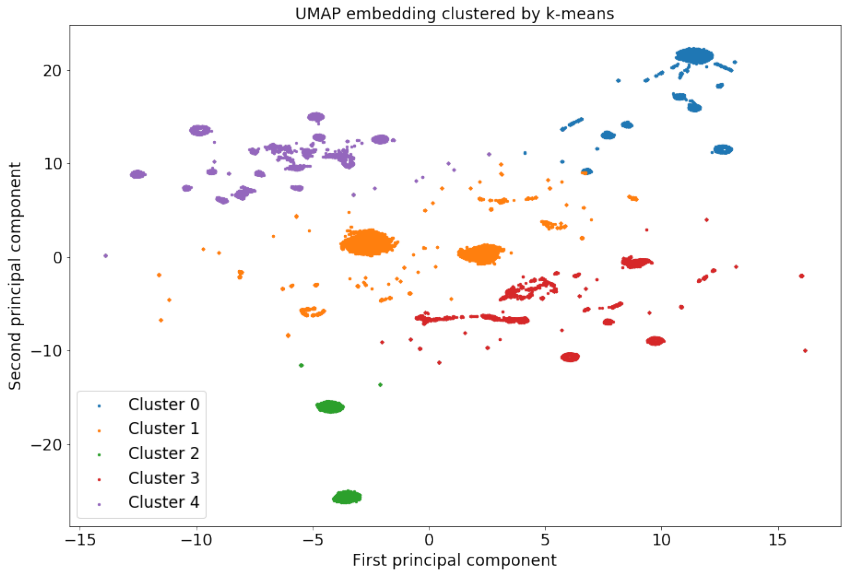


Figure 46. K-means clustering of the UMAP embedding with a medium number of neighbors

The clusters formed in Figure 46 are very similar to the clusters formed in Figure 40. There, areas with both days could be easily distinguished from areas with points of just one day. These areas however are not easily linked with the states of failure or normal condition like we set out to do.

We then did a the silhouette analysis to calculate the performance of the clustering. The mean silhouette score for all the points is 0.604 and the silhouette score plot is shown in Figure 47. The silhouette score plot shows that cluster 3 (coloured bright green) has no points over the mean silhouette score, this means that this cluster could be wrongly clustered or at least it has a bad shape. The rest of the clusters have high silhouette values so, according to this performance metric clustering this UMAP embedding with a k-means clustering with 5 clusters is a good choice.

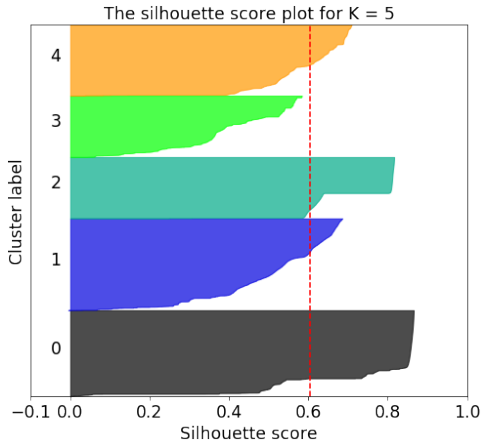
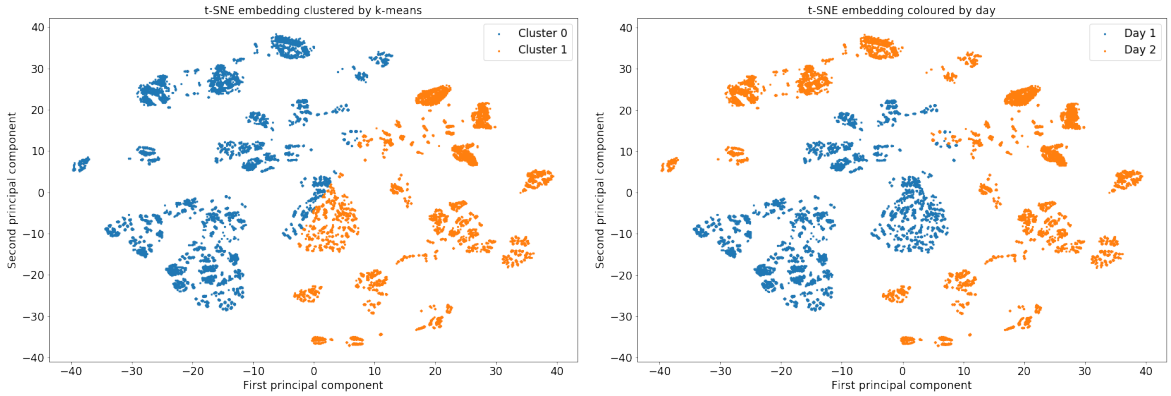


Figure 47. Silhouette score plot of the UMAP embedding with K-means clustering

4.4.3 K-means on t-SNE

This section shows the results of clustering the t-SNE embedding with k-means for both the data set with the numerical data and for the embedding created with the Gower distance. First we applied k-means to the embedding done with both the continuous and the categorical data with

the Gower distance. The k-means algorithm was configured to select two clusters, hoping to see the distribution of clusters from the days like on Figure 42. This is shown in Figure 48.



(a) K-means clustering **(b)** t-SNE embedding labeled by the day

Figure 48. Comparison between clustering and labeled days of the t-SNE embedding with the Gower distance with 2 clusters

In this case, k-means is not able to select correctly the points from day 2 that are on top of day 1 points as being day 2 points. This is because k-means is only able to cluster by the sum of errors of the distance to the centroids of the clusters. In order to calculate the performance of the clustering with a metric, we calculate the silhouette score of the points. The mean silhouette score of this clustering is 0.350 and the complete silhouette analysis is shown in Figure 49.

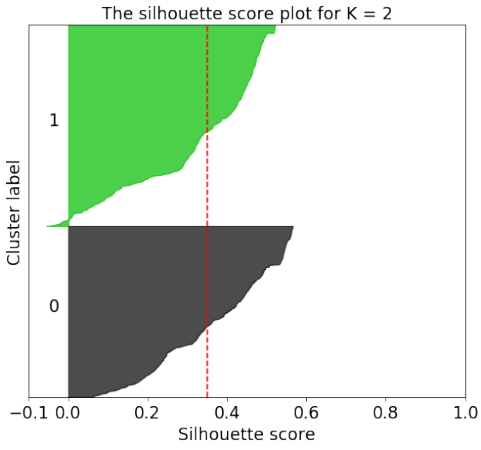
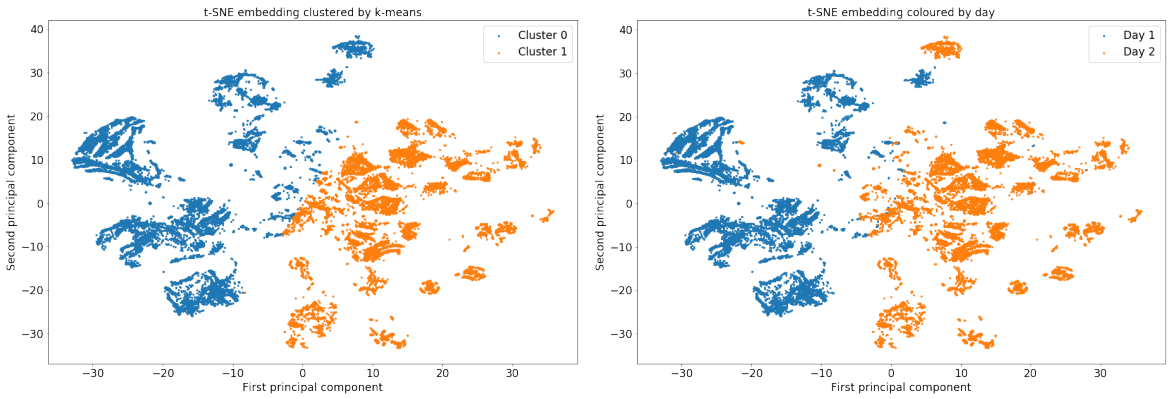


Figure 49. Silhouette analysis of the k-means clustering of the t-SNE embedding with Gower distance

Next data set (t-SNE with just numerical and continuous data) should be easier for k-means to cluster, as there is a separation between both clusters that k-means is able to find. The result of doing a k-means clustering with this data set is shown in Figure 50.



(a) K-means clustering **(b)** t-SNE embedding labeled by the day

Figure 50. Comparison between clustering and labeled days of the t-SNE embedding with continuous features with 2 clusters

Here, k-means was better able to label the points to the correct cluster. Having said that, let's see if this is translated into a better silhouette score than on the other t-SNE embedding with k-means clustering. The mean silhouette score of this k-means clustering is 0.368. The complete silhouette analysis is shown in Figure 51.

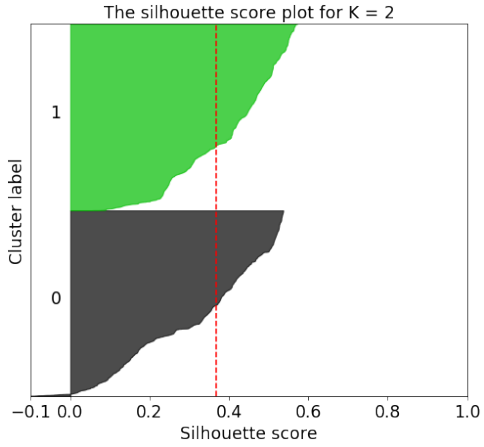


Figure 51. Silhouette analysis of k-means on the t-SNE embedding of numerical data

Here, the plot shows that the silhouette score is very low and that this clustering is not ideal. Maybe with a different number of clusters (in this case higher) the sum of errors would be smaller. In our case this is not something of interest as we only want to distinguish between the 2 days.

The next step is to use a method that is able to separate the clusters taking into account the density of points.

4.4.4 HDBSCAN with PCA

As HDBSCAN is a more complex clustering method than k-means and takes into account the density of the clusters, it may be better suited to separate the clusters of the PCA representation. The result of doing the clustering with the parameter from Table 26 is shown in Figure 52.

min cluster size	min samples
6000	100

Table 26. Parameters for the HDBSCAN clustering of the PCA representation

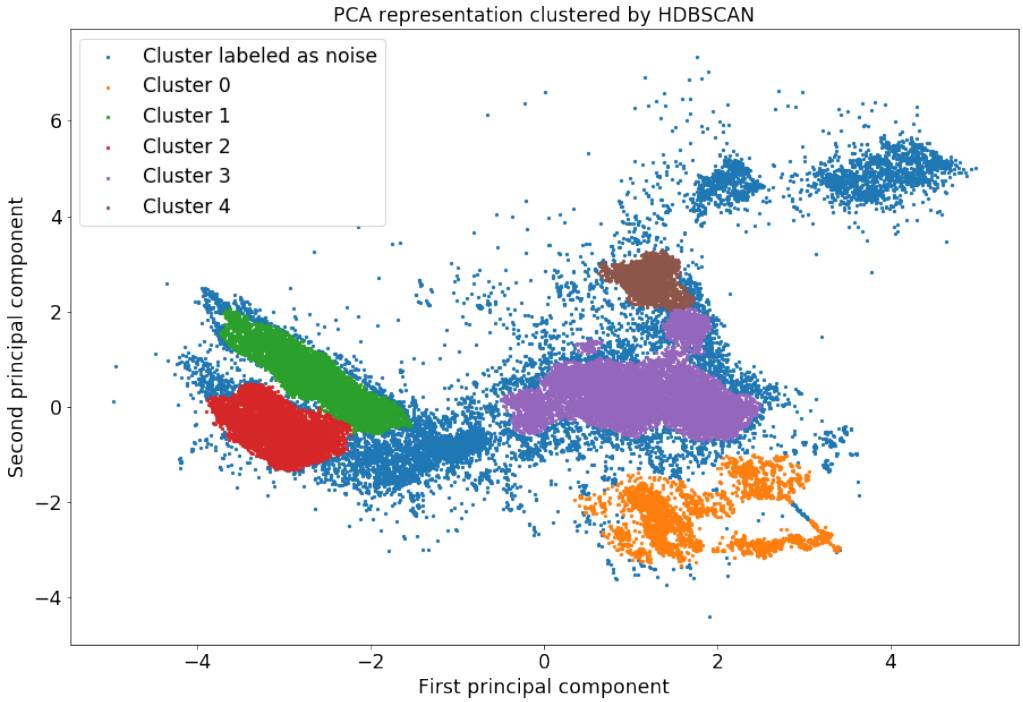


Figure 52. HDBSCAN clustering the PCA representation

In HDBSCAN, the number of clusters can't be selected and instead, the method selects the labels with the restraint of the minimum size of the clusters. Therefore a good balance of density and cluster size has to be found for every data set and the parameters shown in Table 26 are the chosen parameters for this clustering. The results we can extract from looking at Figure 52 is that HDBSCAN is able to separate very distinct clusters that match the separation between the days 1 (clusters 0, 1 and 3) and 2 (clusters 2, 4 and 5) and also the distinction between the first (cluster 0) and last hours (cluster 2) of the data set. HDBSCAN labels much of the data set as noise (17.8% of the data set is considered noise). This gives us information that we didn't have before.

We can calculate the performance of this clustering with the silhouette analysis. The mean silhouette score for this case is 0.401 and the complete silhouette analysis is shown in Figure 53. The mean silhouette score is lower than the clusterings done by k-means for 2 or 4 clusters. This could be the result of including the points labeled as noise (which all have a negative silhouette score) in the calculation of the mean. The rest of the clusters all have high values of silhouette score and they all have points higher than the mean score.

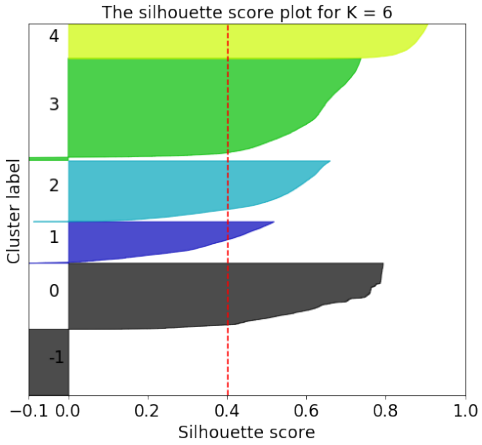


Figure 53. Silhouette analysis of HDBSCAN clustering of the PCA representation

4.4.5 HDBSCAN with UMAP

The very different densities and sizes of the clusters in the UMAP embedding with the optimal number of neighbors make HDBSCAN a better method (a priori) for clustering the data set. This configuration of clusters and densities is also why we selected these parameters for the clustering. With a small number of minimum samples, the space between the points can be big and therefore, less points are clustered as noise. Some points are labeled as noise are the points between the clusters and the ones that are very far from any of the clusters. Figure 54 shows the result of the clustering with HDBSCAN with the parameters shown in Table 27.

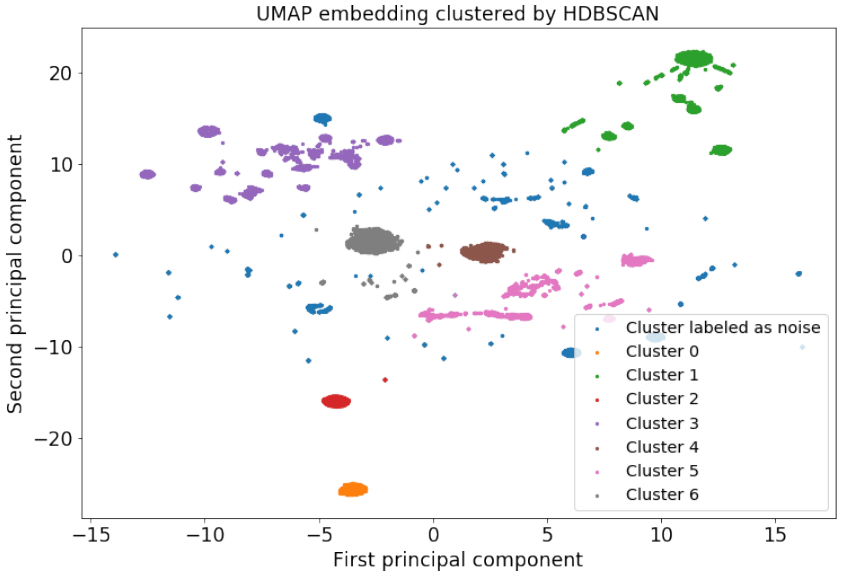


Figure 54. HDBSCAN clustering with UMAP embedding

min cluster size	min samples
3000	200

Table 27. Parameters for the HDBSCAN clustering of the UMAP embedding

The labeling of the noise added information we hadn't had before. The only way to compare this clustering with the k-means clustering is with a silhouette analysis, shown in Figure 55. The

mean silhouette score for this clustering is 0.548, which is lower than the k-means clustering of the same data set. This could be partly influenced by the inclusion of the points labeled as noise, which all have negative silhouette scores, in the calculation of the mean silhouette score. On both clusterings there are some clusters whose points' have a smaller silhouette score than the mean silhouette score.

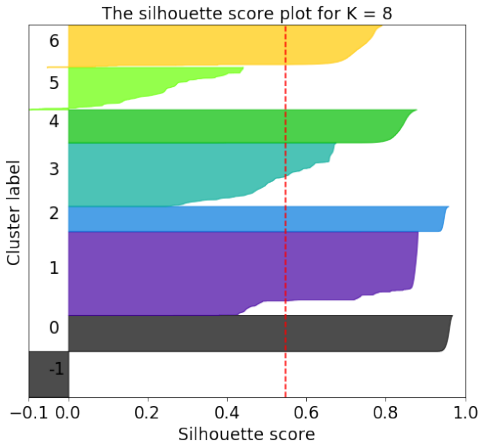


Figure 55. Silhouette analysis of HDBSCAN clustering of UMAP embedding

4.4.6 HDBSCAN on t-SNE

In this data set, there are many big clusters that represent different states that we are not able to identify. With this situation, HDBSCAN is very probable to find many clusters in the data rather than just two from the k-means method. Figure56 shows an HDBSCAN clustering example on the t-SNE embedding of the data set with both the numerical and continuous features and the categorical and binary features using the Gower distance. The parameters used for the clustering are shown in Table 28. HDBSCAN was able to find 6 clusters and labeled some data as noise. We can not say with confidence whether one cluster represents a failure state or not.

min cluster size	min samples
3000	200

Table 28. Parameters for the HDBSCAN clustering of the t-SNE embedding with binary features

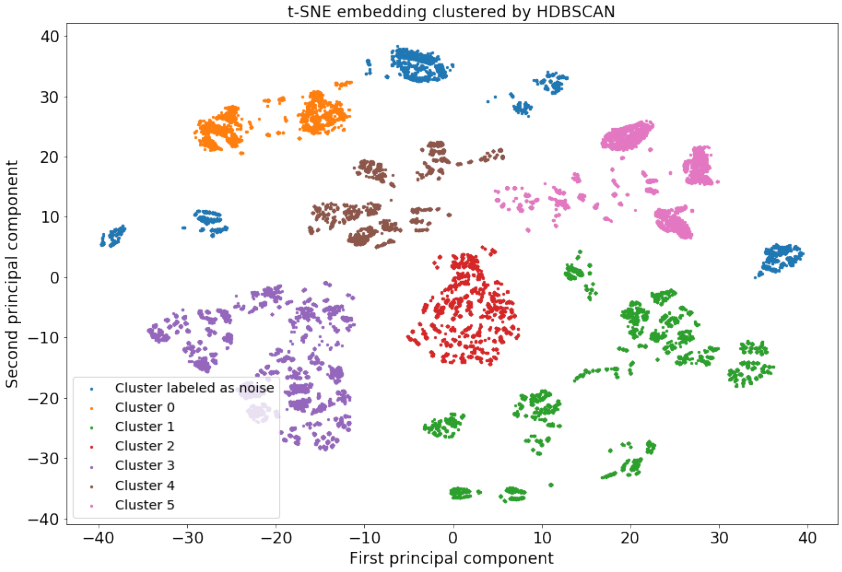


Figure 56. t-SNE with Gower distance clustered by HDBSCAN

We performed a silhouette analysis of the clustering and saw a mean silhouette score of 0.324, which is lower than the clustering made with the same data set with k-means. The complete silhouette analysis is shown in Figure 57.

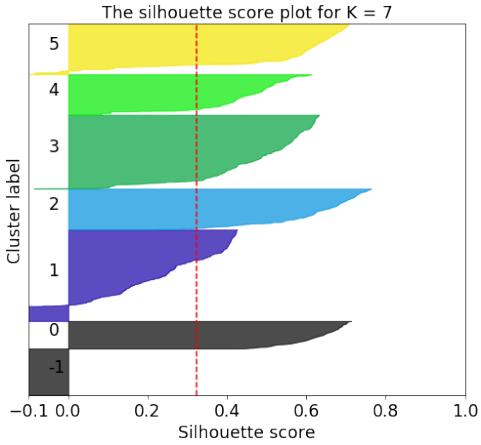


Figure 57. Silhouette analysis of the HDBSCAN clustering of the t-SNE embedding with Gower distance

Then we clustered the t-SNE embedding of numerical and continuous data with HDBSCAN. Figure 58 shows the results of this clustering. The parameters used to calculate this clustering are shown in Table 29.

min cluster size	min samples
3000	200

Table 29. Parameters for the HDBSCAN clustering of the t-SNE embedding with continuous features

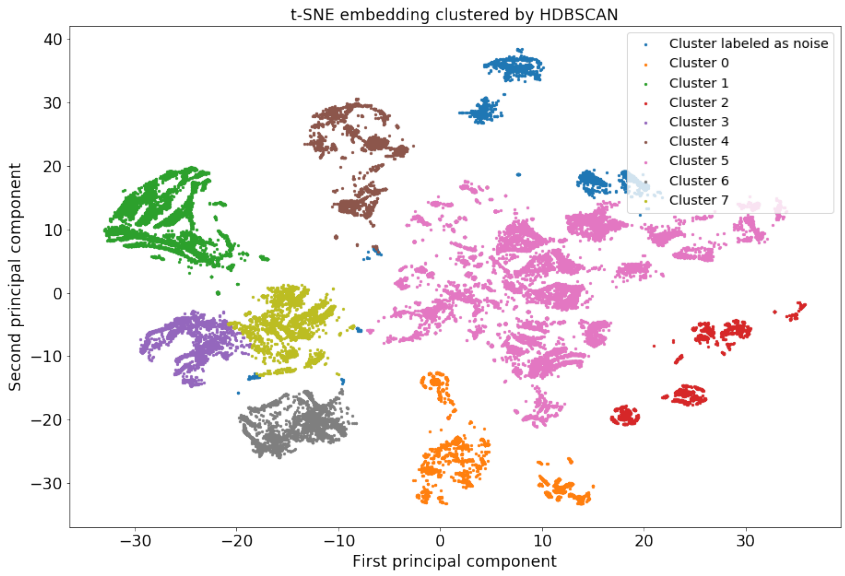


Figure 58. t-SNE embedding with continuous data clustered by HDBSCAN

The mean silhouette score of this clustering is 0.362. The complete silhouette analysis is shown in Figure 59. This silhouette analysis results are inconclusive. On one hand, some of cluster 5 points have a negative silhouette score and the mean silhouette score is very low. On the other hand, the maximum silhouette score of every cluster is higher than the mean silhouette score of the whole data set.

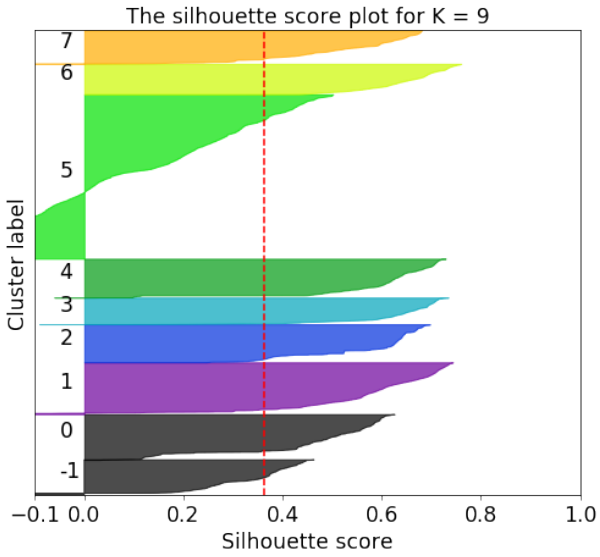


Figure 59. Silhouette analysis of the HDBSCAN clustering of the t-SNE embedding with numerical data

In both clusterings, HDBSCAN was able to correctly label the points as being in different clusters depending on the density of the points but these (at first sight at least) are not the states of the machine or the failure modes.

4.5 Conclusions from the real world data

For this data set, the best dimension reduction method for separating the points into clusters for latter computing the labels with one clustering method, was UMAP. In this case (contrary to what happened on the hydraulic data set) UMAP and t-SNE give very different results. While UMAP produces (in one of the embeddings) very dense and well separated clusters, t-SNE produces clusters that are less dense and that are more difficult to cluster with a clustering method. In terms of speed, UMAP was also faster than t-SNE (even with the t-SNE optimizations and the multicore processing). The comparison between both dimension reduction transformations is shown in Table 30.

Dimension reduction method	Compute time (min)
UMAP (continuous data)	40
UMAP (continuous and binary)	88
UMAP (two graphs)	237
t-SNE (continuous)	163
t-SNE (continous and binary)	219

Table 30. Comparison of the compute time of the different dimension reduction methods

PCA is not shown in the table but it was a clear winner in this department because its computation is almost instant (800 ms). The difference in computing time between the different rows of Table 30 comes from the computing of the distance matrix for the cases with continuous and binary data (which takes 69 minutes) and also in the two graphs case, it is sensible to have a higher computing time as the computer has to calculate to different embeddings and then combine them.

Even if the method is faster, if it gives considerably worse results, it won't be used. Table 31 shows the different performance metrics of the different clusterings with the embeddings. UMAP has the best result in both K-means and HDBSCAN clustering when compared to the rest of the dimension reduction methods. K-means with UMAP gets a better result than HDBSCAN with UMAP but only by 0.05 points. This difference in a metric that does not take into account the real labels should not be a deal-breaker. In this case, the method that is the most reliable and most easy to use should be the one that is recommended. When clustering the embedding of this data set selecting the number of clusters is not easy and it is much easier to select a minimum number of points inside a cluster. A company that wants to learn more about a process knows the sampling rate of their sensors and therefore knows approximately the size of the data set they are studying, but if there is no information about the amount of states the system can be in, the number of clusters can be difficult to select.

Methods	Mean silhouette score
K-means with PCA (2 clusters)	0.494
K-means with PCA (4 clusters)	0.570
K-means with UMAP	0.604
K-means with t-SNE	0.368
HDBSCAN with PCA	0.401
HDBSCAN with UMAP	0.548
HDBSCAN with t-SNE	0.324

Table 31. Comparison of performance between the different clustering methods

Even though unsupervised learning can give some more information about the data set than what was available before the unsupervised learning study, it will always be better to have the label information. This will result in a much better result and the ability to act with the information in order to do the condition monitoring of the machine.

With this method it is very difficult to say whether a cluster represents a failure or not. While on the data set of the hydraulic system we had the labels that told us whether a point had a failure or not, here there is not enough information that would indicate if a point is a failure or not. This means that this method is not enough to detect and react to failures in machines on a real life situation of a production.

Chapter 5

Conclusions

This thesis has proved that unsupervised learning is a useful method to extract information from data of which we knew nothing or little to nothing. It can also help detect failures in the system when the quality of the data is sufficient. However, it can not be seen as the only mechanism to detect failures in data with no labels. Even with high quality data, the performance of the method is not high enough to rely solely on it.

In both the hydraulics and the real data sets, UMAP was the superior dimensionality reduction method. UMAP compared to PCA was able to better group together points in order to improve the clustering performance, it was also able to find more clusters in both data sets, better explaining the structure of the whole data set unlike PCA where much information was lost in its many components. UMAP and t-SNE created similar embeddings but UMAP had significantly lower computing times. This affects the user's ability to apply this method with less powerful hardware and makes reaching a decision in the maintenance much quicker. In addition, as the data set increases its size, UMAP is a better option as t-SNE will stop being a feasible solution due to the big increase in computing time relative to the size of the data set.

When comparing clustering methods, HDBSCAN and k-means were very similar in performance. Both methods offered similar results when the correct parameters were chosen. The main difference between both clustering methods is that in order to make k-means work, the user has to know the number of clusters beforehand and this is not always possible (specially in the unsupervised learning case). Even when a user knows how its machine works, it may not know how many failure modes or activities the machine might have done during the length of the data set (which could give the user a clue about the amount of clusters to select in k-means). HDBSCAN is also a great tool to get more information about the data set it is working with. Condensed tree dendograms and the ability to include new data points into the clustering (after having fitted the model) are additional resources the user might use that were not available with k-means. Therefore, in this case HDBSCAN is the superior method for doing clustering in unsupervised learning.

In one hand, the proposed method showed good performance with the hydraulics data set because the data was correctly acquired and only included features that affected the components whose state we wanted to study. On the other hand, in the end we could not say with certainty, based on the cluster a point was labeled into, which state each of the 4 components was in. With the real-world data set, the performance was not as good as with the hydraulics as the data quality was not as good. Even though the method was able to extract information not available before to the user, the state of the machine was not clear when clustering the data set.

In conclusion, this thesis has proved that the proposed method is better than other commonly used methods for unsupervised clustering for condition monitoring with lower computing times

and higher clustering quality. However, the method's ability to discern between the states of a machine is highly dependant on the data quality.

5.1 Next steps

As this project had to come to an end, we were not able to complete everything we had planned. But if someone else wants to continue our research is welcome to do so with the following recommendations.

1. Continue to improve the UMAP algorithm. The algorithm's performance when using mixed type data is poor.
2. Investigate the effect of changing features that give no information about machine damage with features that do and quantify that effect.
3. Create an algorithm that can be implemented into low-power embedded systems with technologies like multiprocessing (which UMAP and HDBSCAN currently do not support).
4. Compare this result with the performance a supervised learning method would achieve with the same data set. Calculate the cost for a company to get the information necessary to turn an unsupervised learning problem into a supervised one and calculate if the change would be economically feasible.

Bibliography

[1] M. M. Alamdari et al. “A spectral-based clustering for structural health monitoring of the Sydney Harbour Bridge”. In: Mechanical Systems and Signal Processing 87 (Mar. 2017), pp. 384–400. URL: http://www.sciencedirect.com/science/article/pii/S0888327016304538.

[2] D. A. Azzawi et al. “Comparison of immunity-based schemes for aircraft failure detection and identification”. In: Engineering Applications of Artificial Intelligence 52 (2016), pp. 181–193. URL: http://www.sciencedirect.com/science/article/pii/S0952197616300379.

[3] E. S. Berner and T. J. La Lande. “Overview of Clinical Decision Support Systems”. In: Clinical Decision Support Systems: Theory and Practice. Ed. by E. S. Berner. New York, NY: Springer New York, 2007, pp. 3–22. ISBN: 978-0-387-38319-4. URL: https://doi.org/10.1007/978-0-387-38319-4_1.

[4] S. van Buuren and K. Groothuis-Oudshoorn. “mice: Multivariate Imputation by Chained Equations in R”. In: Journal of Statistical Software, Articles 45.3 (2011), pp. 1–67. URL: https://www.jstatsoft.org/v045/i03.

[5] R. J. G. B. Campello et al. “Density-Based Clustering Based on Hierarchical Density Estimates”. In: Advances in Knowledge Discovery and Data Mining. Ed. by J. Pei et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 160–172. ISBN: 978-3-642-37456-2.

[6] P. Cawley. “Structural health monitoring: Closing the gap between research and industrial deployment”. In: Structural Health Monitoring 17.5 (Sept. 2018), pp. 1225–1244. URL: http://journals.sagepub.com/doi/10.1177/1475921717750047.

[7] P. Deka. Predictive and Prescriptive Maintenance for Manufacturing Industry with Machine Learning. Towards Data Science. Dec. 24, 2018. URL: https://towardsdatascience.com/predictive-and-prescriptive-maintenance-for-manufacturing-industry-with-machine-learning-2078afa76bfb.

[8] N. Dervilis. “A machine learning approach to Structural Health Monitoring with a view towards wind turbines”. PhD thesis. University of Shefield, 2013.

[9] M. Ester et al. “A density-based algorithm for discovering clusters in large spatial databases with noise”. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining. AAAI Press, 1996, pp. 226–231. URL: http://dl.acm.org/citation.cfm?id=3001460.3001507.

[10] B. A. Galler and M. J. Fisher. “An Improved Equivalence Algorithm”. In: Commun. ACM 7.5 (May 1964), pp. 301–303. URL: http://doi.acm.org/10.1145/364099.364331.

[11] J. C. Gower. “A General Coefficient of Similarity and Some of Its Properties”. In: Biometrics 27.4 (1971), pp. 857–871. URL: http://www.jstor.org/stable/2528823.

[12] N. Helwig et al. “Condition monitoring of a complex hydraulic system using multivariate statistics”. In: 2015 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings. May 2015, pp. 210–215. ISBN: 978-1-4799-6114-6.

[13] P. Henriquez et al. “Review of Automatic Fault Diagnosis Systems Using Audio and Vibration Signals”. In: 44.5 (June 2014), pp. 642–652.

[14] G. James et al. An Introduction to Statistical Learning: With Applications in R. Springer texts in statistics 103. Springer Publishing Company, Incorporated, 2014. 426 pp. ISBN: 978-1-4614-7137-0.

[15] I. Jolliffe. “Graphical Representation of Data Using Principal Components”. In: Principal Component Analysis. New York, NY: Springer New York, 2002, pp. 78–110. ISBN: 978-0-387-22440-4. URL: https://doi.org/10.1007/0-387-22440-8_5.

[16] I. Jolliffe. “Introduction”. In: Principal Component Analysis. New York, NY: Springer New York, 2002, pp. 1–9. ISBN: 978-0-387-22440-4. URL: https://doi.org/10.1007/0-387-22440-8_1.

[17] S. Khan et al. “No Fault Found events in maintenance engineering Part 2: Root causes, technical developments and future research”. In: Reliability Engineering and System Safety 123 (2014), pp. 196–208. URL: http://www.sciencedirect.com/science/article/pii/S0951832013002974.

[18] N. L. D. Khoa et al. “Structural Health Monitoring Using Machine Learning Techniques and Domain Knowledge Based Features”. In: Human and Machine Learning: Visible, Explainable, Trustworthy and Transparent. Ed. by J. Zhou and F. Chen. Cham: Springer International Publishing, 2018, pp. 409–435. ISBN: 978-3-319-90403-0. URL: https://doi.org/10.1007/978-3-319-90403-0_20.

[19] D. Leith et al. “Condition monitoring of electrical machines using real-time expert system”. In: (Sept. 1988), pp. 297–302.

[20] C. F. Lorenzo and W. C. Merrill. “An intelligent control system for rocket engines: need, vision, and issues”. In: IEEE Control Systems Magazine 11.1 (Jan. 1991), pp. 42–46.

[21] L. van der Maaten and G. Hinton. “Visualizing Data using t-SNE”. In: Journal of Machine Learning Research 9 (2008), pp. 2579–2605. URL: http://www.jmlr.org/papers/v9/vandermaaten08a.html.

[22] L. McInnes. How HDBSCAN works. 2018. URL: https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html#how-hdbscan-works.

[23] L. McInnes. Performance Comparison of Dimension Reduction Implementations. 2018. URL: https://umap-learn.readthedocs.io/en/latest/benchmarking.html.

[24] L. McInnes et al. “hdbscan: Hierarchical density based clustering”. In: The Journal of Open Source Software 2.11 (Mar. 2017). URL: https://doi.org/10.21105%5C%2Fjoss.00205.

[25] L. McInnes et al. “UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction”. In: (Feb. 9, 2018).

[26] F. Meziane and S. Vadera. “Artificial intelligence applications for improved software engineering development: new prospects”. In: (June 12, 2019).

[27] A. Mousse. How to understand the drawbacks of Hierarchical Clustering? Feb. 2016. URL: https://stats.stackexchange.com/questions/183873/how-to-understand-the-drawbacks-of-hierarchical-clustering.

[28] F. Pedregosa et al. Clustering. 2011. URL: https://scikit-learn.org/stable/modules/clustering.html#.

[29] F. Pedregosa et al. Manifold learning. 2011. URL: https://scikit-learn.org/stable/modules/manifold.html.

[30] F. Pedregosa et al. Model evaluation: quantifying the quality of predictions. 2011. URL: https://scikit-learn.org/stable/modules/model_evaluation.html.

[31] F. Pedregosa et al. Preprocessing data. 2011. URL: https://scikit-learn.org/stable/modules/preprocessing.html.

[32] P. Policar. openTSNE: Extensible, parallel implementations of t-SNE. 2018. URL: https://opentsne.readthedocs.io.

[33] B. Polyak and A. Juditsky. "Acceleration of Stochastic Approximation by Averaging". In: SIAM Journal on Control and Optimization 30.4 (1992), pp. 838–855. eprint: https://doi.org/10.1137/0330046. URL: https://doi.org/10.1137/0330046.

[34] F. Pozo et al. "Detection of structural changes through principal component analysis and multivariate statistical inference". In: Structural Health Monitoring 15.2 (2016), pp. 127–142. eprint: https://doi.org/10.1177/1475921715624504. URL: https://doi.org/10.1177/1475921715624504.

[35] R. C. Prim. "Shortest connection networks and some generalizations". In: The Bell System Technical Journal 36.6 (Nov. 1957), pp. 1389–1401.

[36] W. Rand. "Objective criteria for the evaluation of clustering methods". In: Journal of the American Statistical Association 66.336 (1971), pp. 846–850.

[37] D. Robot. What is Unsupervised Machine Learning? 2019. URL: https://www.datarobot.com/wiki/unsupervised-machine-learning/.

[38] L. Rokach and O. Maimon. "Clustering Methods". In: Data Mining and Knowledge Discovery Handbook. Ed. by O. Maimon and L. Rokach. Boston, MA: Springer US, 2005, pp. 321–352. ISBN: 978-0-387-25465-4. URL: https://doi.org/10.1007/0-387-25465-X_15.

[39] SAS. Machine Learning: What it is and why it matters. URL: https://www.sas.com/en_ae/insights/analytics/machine-learning.html.

[40] R. Schiban. Strategies for a Successful IoT Predictive Maintenance Program. Hitachi Solutions. June 6, 2017. URL: https://us.hitachi-solutions.com/blog/6-tools-for-a-successful-predictive-maintenance-program/.

[41] M. Sharma. What steps should one take while doing Data Preprocessing. July 25, 2018. URL: https://hackernoon.com/what-steps-should-one-take-while-doing-data-preprocessing-502c993e1caa.

[42] G. Singh et al. "Machine Learning-Based Framework for Multi-Class Diagnosis of Neurodegenerative Diseases: A Study on Parkinson's Disease". In: IFAC-PapersOnLine 49.7 (2016). 11th IFAC Symposium on Dynamics and Control of Process SystemsIncluding Biosystems DYCOPS-CAB 2016, pp. 990–995. URL: http://www.sciencedirect.com/science/article/pii/S2405896316305389.

[43] K. Smarsly et al. "Machine learning techniques for structural health monitoring". In: July 2016.

[44] L. Van Der Maaten. "Accelerating t-SNE Using Tree-based Algorithms". In: J. Mach. Learn. Res. 15.1 (Jan. 2014), pp. 3221–3245. URL: http://dl.acm.org/citation.cfm?id=2627435.2697068.

[45] K. Worden and C. Farrar. Structural Health Monitoring. A Machine Learning Perspective. Chichester, West Sussex, U.K. ; Hoboken, N.J: John Wiley and Sons, 2013, pp. 17–38. ISBN: 978-1-119-99433-6.

[46] K. Worden and G. Manson. "The application of machine learning to structural health monitoring". In: Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 365.1851 (2007), pp. 515–537. eprint: https://royalsocietypublishing.org/doi/pdf/10.1098/rsta.2006.1938.

URL: https://royalsocietypublishing.org/doi/abs/10.1098/rsta.2006.1938.

[47] Q. Zhao. “Clustering Validity in Clustering Methods”. In: (June 2012). URL: http://epublications.uef.fi/pub/urn_isbn_978-952-61-0841-4/urn_isbn_978-952-61-0841-4.pdf.

[48] Y.-L. Zhou et al. “Structural damage detection using transmissibility together with hierarchical clustering analysis and similarity measure”. In: Structural Health Monitoring 16.6 (2017), pp. 711–731. eprint: https://doi.org/10.1177/1475921716680849. URL: https://doi.org/10.1177/1475921716680849.