Tampere University

Thomas Etienne Foucault

COMPARING CLUSTERING METHODS FOR MOBILE NETWORK ROOT CAUSE DETECTION

Master's Thesis
Faculty of Information Technology and Communication Sciences
April 2020

ABSTRACT

Thomas Etienne Foucault: Comparing clustering methods for mobile network root cause detection
Master's Thesis, 52 pages
Tampere University
Master's Degree Programme in Data Engineering and Machine Learning
April 2020
Examiners: Prof. Tapio Elomaa (Tampere University), Asst. Prof. Sergey Andreev (Tampere University)
Supervisors: Prof. Tapio Elomaa (Tampere University), Asst. Prof. Sergey Andreev (Tampere University), Dr. Tech. Adrian Burian (Nokia), Mr. Janne Helenius (Nokia)

Mobile networks represent a considerable industry globally and are known to rely on robust and highly reliable systems. As a consequence, faults in the system may induce a significant loss of credibility and revenues for mobile operators. However, while mobile network systems implement more features, they also become more complex and difficult to troubleshoot. In response to this issue, this thesis explores the capabilities of cluster analysis methods in order to facilitate the tasks of troubleshooting experts and reduce the cost of mobile networks maintenance.

A comparison of eight different clustering methods is proposed. Each of them is a combination of a dimensionality reduction algorithm (Principal Component Analysis or Self-Organizing Maps) and a clustering algorithm (K-means, OPTICS, or Growing Neural Gas with Post-Pruning), with two exceptions which do not use dimensionality reduction.

The results show that OPTICS performs poorly for this task most of the time. However, K-means and Growing Neural Gas with Post-Pruning demonstrate interesting capabilities for detecting several mobile network faults. Both methods present advantages and disadvantages.

Keywords: Mobile Networks, Classification, Unsupervised Learning, Root Cause Analysis, Anomaly Detection, Cluster Analysis

PREFACE

The work presented in this thesis has been conducted at Nokia and made as a completion of a Master's degree programme in Data Engineering and Machine Learning proposed by Tampere University.

I would like to express my gratitude to all my teachers, supervisors and colleagues who supported me in this project, without whom this work would have been much more difficult. In particular, I would like to thank my supervisors from Tampere University, Tapio Elomaa and Sergey Andreev, for their precious advice for writing and structuring this document. In addition, I also thank my colleagues and supervisors from Nokia, Adrian Burian and Janne Helenius, who offered their valuable expertise and guidance to conduct this work the best possible way.

Finally, my appreciation goes to my family and friends who are great sources of motivation. Their endless support through my years of education was crucial for me to turn my wishes into reality.

Tampere, 28 April 2020

Thomas Etienne Foucault

TABLE OF CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

1D	One-dimensional
2D	Two-dimensional
3G	Third Generation
3GPP	Third Generation Partnership Project
4G	Fourth Generation
5G	Fifth Generation
BMU	Best Matching Unit
DB index	Davies-Bouldin index
dBm	Decibel-milliwatt
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
eNodeB	Evolved Node B
EPC	Evolved Packet Core
E-UTRAN	Evolved Universal Terrestrial Radio Access Network
GNG	Growing Neural Gas
GNG-PP	Growing Neural Gas with Post-Pruning
IQR	Interquartile Range
KDE	Kernel Density Estimation
KPI	Key Performance Indicator
KS test	Kolmogorov-Smirnov test
LTE	Long Term Evolution
MGNG	Merge Growing Neural Gas
OPTICS	Ordering Points To Identify the Clustering Structure
PCA	Principal Component Analysis
PDF	Probability Density Function
RCA	Root Cause Analysis
RSRP	Reference Signal Received Power
RSRQ	Reference Signal Received Quality
SINR	Signal to Interference and Noise Ratio
SL	Significance Level
SMS	Short Message Service
SOM	Self-Organizing Map
SON	Self-Organizing Network
UE	User Equipment

1. INTRODUCTION

Mobile networks represent a large industry globally with 5.2 billion individual mobile phone subscribers and 1.06 trillion US$ of revenue in 2019 [27]. Over the past, the industry faced a fast growth of user demands and consequently expanded its presence worldwide. Therefore, mobile technology capabilities improved considerably, from the first commercial 3G networks launched in 2001 [15], to the deployment of 4G *Long Term Evolution* (LTE) in 2009 [19], and 5G in 2018 [54].

Due to the high demand, mobile networks are expected to be highly reliable and accessible. As a consequence, problems in a network can have a significant financial impact and may induce a loss of credibility for mobile operators. However, mobile network systems are now very complex and produce an increasing volume of data that are difficult to interpret. Therefore, in order to reduce the troubleshooting costs for mobile operators, this study proposes a comparison of methods to improve the understanding of mobile network data and facilitate the work of experts.

In response to the growing interest for reducing troubleshooting costs, *Self-Organizing Networks* (SONs) were introduced into LTE [1]. They establish a concept of algorithms delivering automation for – among others – configuration, optimization and healing. In particular, *self-healing* is a category of SON that aims at automatically solving or mitigating faults in a network [2]. Therefore, due to the growing interest of mobile operators in self-healing, this study focuses on methods to improve mobile network troubleshooting.

A problem can generally be split in multiple parts [4]. First, the symptoms are consequences of problems and are a sign that a problem exists. Then, first-level causes that directly produce the symptoms. Following are higher-level causes which do not cause problems directly but form a chain of causes leading to the problem. Finally, the root cause is the source whose consequences created the problem (Figure 1).

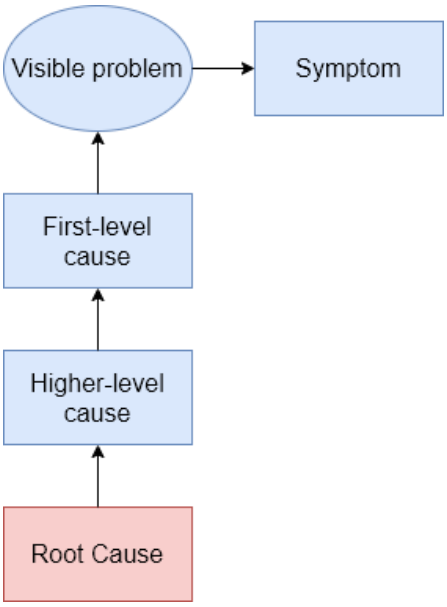


Figure 1. *The different levels of a problem, adapted from* [4].

There are essentially two approaches of solving problems:

1. Removing the symptoms or solving the first- or higher-level causes. This usually provides a temporary relief, but the problem will eventually emerge with new causes and symptoms.
2. Eliminating the root cause and preventing the problem from happening again.

The first approach finds its use in automatic systems where finding a quick solution is critical. The second one, however, proposes a long-term solution for a more robust system: this is where *Root Cause Analysis* (RCA) comes into use. First defined in [60], RCA is a process for identifying root causes of a problem and finding the actions needed to solve them. More precisely, it consists of a set of tools and techniques that can be used to achieve this goal.

RCA has an essential role for troubleshooting mobile networks. However, their complexity and the increasing volumes of data make them difficult to diagnose and correct by experts. This tedious task can be replaced by a system that analyses the incoming data automatically using machine learning methods.

Machine learning is a subfield of Artificial Intelligence which includes algorithms that can learn to predict and recognize patterns without being specifically programmed. Machine learning algorithms build models relying on sample data in order to perform efficiently using the learned behaviors. There exist three main approaches to learning from data:

- **Supervised learning** relies on data from which the elements we want to predict are already known i.e. the data is labeled. Such algorithms learn to map the known behaviors to patterns of data.

- **Unsupervised learning** uses data that is unlabeled. It mostly focuses on identifying similarities and dissimilarities in the data to find structures or groups of data points.
- **Semi-supervised learning** is between supervised and unsupervised learning. In this case, part of the samples is labeled while the rest is unlabeled. Having a small amount of labeled data can significantly increase the results of learning.

Supervised or semi-supervised learning have been applied for diagnosing mobile networks [23]. However, these methods often rely on artificial data or expert knowledge and are difficult to use on real cases. In reality, experts in troubleshooting rarely have the time to annotate the data they study. Consequently, even though there exist datasets used for troubleshooting that can be analyzed, they do not contain labels indicating whether a data point corresponds to a fault or not [26]. Therefore, unsupervised learning techniques are better suited than others in this context because they can be used on existing unlabeled datasets.

One of the main categories of unsupervised learning methods is *cluster analysis* [18]. Cluster analysis is the procedure of grouping data elements together in groups (called clusters) such that elements in the same cluster are similar to each other and dissimilar to objects in other clusters.

This study explores the capabilities of unsupervised cluster analysis methods to interpret the incoming data and present network operators with a selected set of essential information. Therefore, we conduct a comparison of the performance of different cluster analysis methods to find relevant patterns in a mobile network. The detection of these patterns allows experts to identify the root causes of a possible issue faster, hence reducing expert workload and the troubleshooting costs for mobile operators. In order to qualify the algorithms accurately, they are evaluated based on their classification performance, consistency, computation time, and sensitivity to parameter tuning. Finally, this study provides recommendations based on the evaluation of clustering methods.

Chapter 2 contains a review of existing work to solve the indicated problem. Next, Chapter 3 details the clustering methods selected for the study. Then, Chapter 4 describes the implementation of the methodology. Further, Chapter 5 presents the results of the experiments and an analysis. Finally, Chapter 6 proposes a summary of the entire work in the form of recommendations, and potential future research paths.

2. RELATED WORK

Improving the mobile network reliability is a topic that has raised a lot of interest from researchers and mobile operators. As a consequence, a variety of papers have been published for this purpose. The large number of publications can be explained by the variety of topics, from studies concerning different models, frameworks or methods, to the kind of causes to detect.

Two kinds of causes appear frequently in publications: *cyberattacks* [47], [58] and *network faults* [23], [38]. For example, a method for detecting anomalies produced by diverse attacks is introduced in [47]. In particular, the authors of the latter study successfully tested their approach on two network datasets including signaling denial of service attacks against the mobile network and attacks against mobile users. Another study [58] focuses on detecting mobile networks of viruses (botnets) based on abnormal activities.

Then, researchers also attempted to improve mobile networks reliability by detecting network faults. These faults are generally produced by a wrong configuration of the mobile network or special events that may affect the quality of service provided by the network, such as bad weather or an accumulation of users in a dense area. In this study, we refer to network faults as abnormal behaviors perceived in a mobile network, such as high interference between several network cells or lack of coverage.

In [38], the author introduces a method to monitor the resource consumption patterns of a mobile network in order to detect possible faults introduced after a software update – hence caused by a wrong configuration. Another study [23] proposes a data discretization method to improve the diagnosis of known network faults such as a lack of signal coverage or the overload of a network cell.

In regard to these two categories of causes, this study focuses on the detection of mobile network faults patterns. Several studies that focus on complementary topics to the detection of faults have been proposed, like a mobile network performance evaluation method [59] or an improvement of experts knowledge acquisition techniques for self-healing [30]. However, papers proposing methods for detecting patterns of mobile network faults are scarcer.

A method based on unsupervised learning is proposed in [23]. The diagnosis is done in two steps: the data points features are first discretized in two states (high and low) using cluster analysis to determine a threshold for each feature. Then, the different combinations of high and low features are mapped to a set of causes determined by experts using

a Bayesian network. The approach is tested against synthetic data containing cases for all the expected causes. In contrast, our study focuses on real data from which the potential causes are unknown.

Another approach based on Merged Growing Neural Gas (MGNG) [5] is studied in [25]. MGNG is an artificial neural network that learns to represent time-series data with an undirected graph. In the latter study, the quantization error of the algorithm is used to detect anomalies. The authors show the model produces a large quantization error when processing anomalous data points, after training it on faultless data. Although the results are interesting, this latter study provides neither any performance metric, nor explanation on what causes the detected anomalies correspond to.

Then, a method for detecting mobile network faults is introduced in [26]. In addition to detecting different patterns for causes using cluster analysis, the study proposes an automatic tool to map the patterns to causes. This is made possible with the annotation of the clusters by experts after these have been found. Again, the results look promising but there is no clear comparison between the performance of this method and other ones. Therefore, we decide to study the first step of the approach – the detection of patterns – and compare it with other clustering methods.

Due to the great interest of the industry in network anomaly detection techniques, there have been several related contributions. For instance, two papers propose an extensive survey on network anomaly detection techniques [3], [35]. However, these studies provide a general overview of the research area and do not specifically focus on mobile networks, nor provide recommendations for them. In addition, these papers study network intrusions detection techniques, which is not the kind of causes our study focuses on.

A research closer to our study was made in [55], which compares five clustering algorithms using performance metrics. The study reveals an interesting point where misuse detection systems – using well-known attacks – have worse performances than clustering algorithms to detect new attacks. However, though closer to our work than the previously mentioned surveys, this paper concerns network intrusions and not network faults.

Finally, to the best of our knowledge, the study that is most similar to our work was conducted in [49]. The paper introduces a method that compares two clustering algorithms – K-means clustering and hierarchical clustering – to detect anomalies in mobile networks. In contrast to our study, the latter uses only one feature corresponding to the activity of mobile users, counting the number of inbound and outbound calls and text

messages. Therefore, the anomalies detected by their system are limited to an abnormally high or low user activity in an area.

In this study, we propose a method for comparing several clustering techniques and evaluating their performance for detecting different patterns of mobile network behaviors. In addition, our method uses several *Key Performance Indicators* (KPIs) to detect different patterns leading to mobile network faults. Then, at the end of this study, we suggest recommendations on the best clustering algorithms to use based on several evaluation metrics obtained with our method.

3. METHODOLOGY

The main objective of this study is to provide a detailed recommendation on clustering methods for detecting patterns in mobile networks. It is not reasonable to compare all existing clustering methods due to their significant number. To this end, this chapter details the various selected algorithms for dimensionality reduction and clustering in Sections 3.1 and 3.2, respectively.

3.1 Dimensionality reduction techniques

The *curse of dimensionality* is a well-known problem in machine learning. First mentioned in [7], it refers today to the fact that clustering high dimensional data is much more difficult than working with lower dimensions. Indeed, the distances between all data points appear to be approximately the same as the number of dimension rises [10]. Consequently, clustering data points that are equidistant from each other is meaningless in most cases.

The number of dimensions causing the curse of dimensionality to happen is unknown. However, the difference in distance between the data points decreases (by a factor of 4) for datasets with up to 20 dimensions [10]. While our dataset – detailed in Section 4.1 – is composed of relatively few dimensions (6), the classification performance is likely to improve by using *dimensionality reduction*.

Dimensionality reduction is the process of reducing the number of components of a dataset by approximating its most significant features. In addition to improving the clustering, it can significantly reduce the algorithm's running time owing to the smaller vectors to process. This is an important aspect for troubleshooting mobile networks, in which time is a crucial factor.

Part of our experiments are about applying different dimensionality reduction methods to a dataset before clustering. Numerous algorithms for this purpose exist, though we select two of the most popular ones: *Principal Component Analysis* (PCA) [28], [50] and *Self-Organizing Map* (SOM) [33]. The main interest in comparing these methods is in their different ways to process the data: PCA is a statistical procedure while SOM is an artificial neural network. Both methods are detailed in the following subsections in the order of ascending complexity.

3.1.1 Principal component analysis

PCA [28], [50] is one of the oldest and most popular methods used for linear dimensionality reduction. It transforms the input to more significant and less correlated vectors based on its eigenvectors.

The algorithm can be split in the following steps:

1. Calculate the covariance matrix,
2. Calculate the eigenvalues and eigenvectors of the covariance matrix,
3. Sort the eigenvectors and select k most significant ones, and
4. Project the dataset to the selected eigenvectors,

where k is the target number of dimensions for the input dataset.

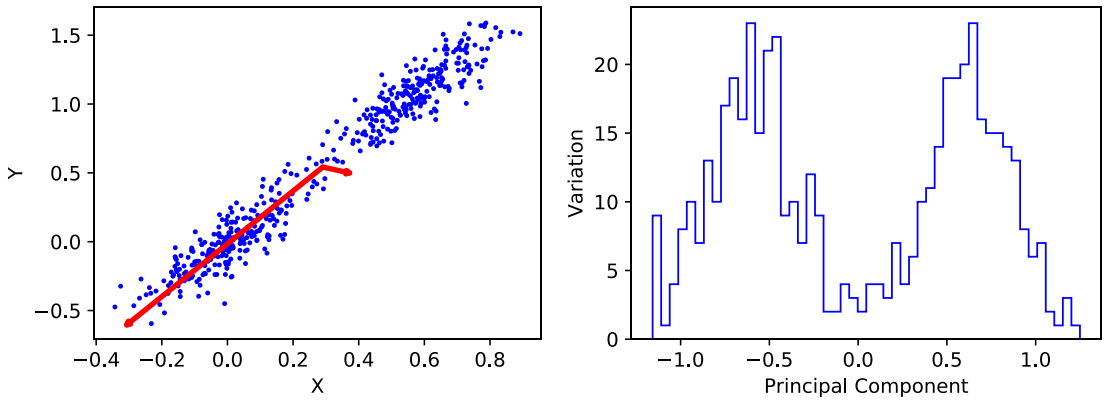
First, the covariance matrix is a square and symmetric matrix of dimensions $d \times d$, where d is the original dataset dimension. The matrix contains the covariance for every pair of variables such that the rows and columns of the matrix represent each variable of the dataset. A matrix cell represents the covariance of the row variable and the column variable it is associated with. As an example, here is the covariance matrix A for $d = 3$, where the variables are X, Y, and Z:

$$A = \begin{bmatrix} \mathrm{cov}(X,X) & \mathrm{cov}(X,Y) & \mathrm{cov}(X,Z) \\ \mathrm{cov}(Y,X) & \mathrm{cov}(Y,Y) & \mathrm{cov}(Y,Z) \\ \mathrm{cov}(Z,X) & \mathrm{cov}(Z,Y) & \mathrm{cov}(Z,Z) \end{bmatrix}. \quad (3.1)$$

Once the covariance matrix has been obtained, we can calculate the d eigenvalues and d eigenvectors of A. The eigenvectors denote the directions of the highest variances of a set of vectors. As an example, the two arrows in Figure 2a represent the eigenvectors of a two-dimensional (2D) generated dataset containing two elongated blobs.

Then, the next step is to sort the eigenvectors by descending order of eigenvalues. The eigenvalues represent the significance of each variable such that the eigenvector associated with the highest eigenvalue is the most significant component. Hence, we can select the first k components into a vector W, as they are the most significant, and calculate the new dataset $Y = W^T \times D$ where D is the original dataset.

To illustrate, the longest arrow of Figure 2a represents the principal component for an example dataset. Therefore, the histogram (Figure 2b) corresponds to the 1D projection of the dataset along the principal component. In this case, the dimension is reduced from 2 to 1 by choosing the most significant component.



(a) (b)

Figure 2. *(a) A plot of randomly generated 2D data. The arrows represent the dataset eigenvectors. (b) The histogram of the dataset presented in (a) after application of PCA to reduce the dimension from 2 to 1.*

3.1.2 Self-organizing map

SOM [33] is an artificial neural network trained to output a representation of the input data into a map of lower dimensions while conserving the topology of the data. The method is comparable to vector quantization, in such a way that the SOM produces nodes associated with weight vectors that are spatially and globally ordered. SOM is commonly used for visualization and clustering purposes [22] in various domains [14], [56], [63], including also mobile networks [26].

There are essentially two types of main algorithms for SOM [31]: 1) the original algorithm processing one data point at a time, which was mostly built on the theory; and 2) a batch training that processes all inputs simultaneously. We use the latter method here because it requires fewer parameters and it converges much faster than the former method. Additionally, most commercial implementations of SOM use the batch training process, including the MATLAB implementation [32] which has inspired the Python implementation [44] that we use in this study.

We use a 2D map with hexagonal tiles, rather than rectangular ones for their greater accuracy [31]. In contrast, the choice of the map size does not follow apparent rules and is more of a trial-and-error process. Still, the map width and length ratio is recommended to have the ratio of the two largest principal components to guarantee a faster convergence in learning than for square maps [31]. The principal components are obtained as described in Section 3.1.1.

Each node is represented by a model m_i, where i is the index of the node in a map of *n* nodes such that $i \in \{1,2,\dots,n\}$. The models are set up using the *linear initialization* method which consists in sampling vectors along the two largest principal components

of the dataset. While initialization with random vectors is possible, using linear initialization makes the model converge faster [31].

A model m_i is essentially a weight vector of length d, the dimension of the input dataset, and is associated with a list of data items. A node can be a winner node, also called *best matching unit* (BMU) of a data point x. The BMU of x is the closest node to x. The distance measure used is the Euclidean distance. Hence, the list of data associated with a model m_i correspond to all the data vectors for which m_i is their BMU. Therefore, the BMU for a data item x_j is calculated as:

$$BMU(x_j) = \underset{i\in\{1,2,\dots,n\}}{\operatorname{argmin}} ||x_j - m_i||, \tag{3.2}$$

where j is the row index of the data points.

In one iteration cycle, for each BMU, all nodes are updated so that their model approaches the input data items. The equation to update the models is:

$$m_i(t+1) = \frac{\sum_{j=1}^{N} h_{ci}(t)\cdot x_j}{\sum_{j=1}^{N} h_{ci}(t)}, \tag{3.3}$$

where N is the number of rows of the input dataset, h_{ci} is a *neighborhood function* and c denotes the index of the current BMU for which models in the neighborhood are updated. Therefore, every model will be updated concurrently such that their new weight vector will be an average of all data inputs weighted by a neighborhood function.

The two most popular neighborhood functions of SOM are the *bubble* and the *Gaussian* functions, respectively calculated by the following formulas:

$$h_{ci}(t) = \begin{cases} 1 & \text{if } ||m_c - m_i|| < \sigma^t \\ 0 & \text{if } ||m_c - m_i|| \geq \sigma^t \end{cases} \text{ and} \tag{3.4}$$

$$h_{ci}(t) = \exp\left(\frac{-||m_c - m_i||^2}{2\cdot(\sigma^t)^2}\right), \tag{3.5}$$

where σ^t is the radius of the neighborhood function at the training iteration t.

The bubble function will either allow or not a node to be updated in the neighborhood, while the Gaussian function will output a decreasing value as a node is further away from the BMU.

We use a two-steps method for training called rough and fine training which generally achieve successful trainings in practice [26], [32]. In the *rough training* phase, the goal is to order the map globally by using a large initial radius linearly decreasing to a final and smaller radius in a few training cycles. Then, the *fine training* aims at converging the map into a stable state, using the last radius value used in the rough training phase and keeping it constant through the iterations. Finally, we follow the experiment setup in [26]

by using the bubble neighborhood function (3.4) in the rough training phase and the Gaussian function (3.5) in the fine one.

Note that SOM can also be considered as a cluster analysis algorithm due to its vector quantization nature, but it usually means that we need to set the number of nodes equal to the number of clusters we want to analyze [16], [22], which is unknown in our case.

3.2 Clustering methods

Once the data dimensions have been reduced, we apply a cluster analysis method to group the data points into clusters following a similarity measure. The notion of "cluster" cannot be precisely defined [21], which is one of the reasons explaining why there are multiple clustering algorithms. Over time, researchers have established various cluster models to answer different definitions of what a cluster is. For the scope of this thesis, we select some of the most popular models used in literature:

- A centroid model based on calculating a mean vector for each cluster: *K-means* [37], [39].
- A density model, focusing on clustering data points based on their density: *OPTICS* [6].
- A neural model using the topology of the data: *Growing Neural Gas* (GNG) [24].

In addition to their popularity, our selection is motivated by the models being part of three different categories. These three algorithms are detailed in the following subsections in the same order they were listed: from simpler to more complex.

3.2.1 K-means clustering

K-means clustering [37], [39] is one of the most popular cluster analysis algorithms used in data mining. Its principle is to group the input data into k clusters of equal variances by associating observations to the nearest mean vector. This vector corresponds to the mean of all observations it is associated with.

Although several variants of the algorithm exists, we use the original Lloyd's algorithm [37] as implemented in the scikit-learn library [12]. The algorithm performs the following steps:

1. Initialize k vectors $\mu_i, i \in \{1,2,\dots,k\}$, with random samples from the input dataset.
2. Assign each observation to its nearest vector μ_i based on Euclidean distance. All observations associated with the same vector μ_i are part of the same cluster C_i.
3. Update each vector μ_i to correspond to the mean of all observations in C_i.

4. Repeat the steps 2 and 3 iteratively until none of the μ_i change significantly anymore.

One drawback of this algorithm is that the results can be different depending on the vectors chosen randomly during the initialization step. Therefore, the algorithm is executed several times to find the best cluster setting. This is done by minimizing the within-cluster sum-of-squares:

$$\underset{C}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{x \in C_i} ||x - \mu_i||^2 , \tag{3.6}$$

where x is an observation within cluster C_i.

As previously mentioned, K-means clustering requires the number of clusters k as input. However, we do not rely on a priori knowledge of the data and cannot infer the correct number of clusters empirically.

Gómez-Andrades et al. [26] propose a method to find the most appropriate number of clusters for K-means clustering. Their approach starts with computing different sets of clusters $S_k = \{C_1, C_2, \dots, C_k\}$ for a series of k such that $k = 2,3, \dots, P$, where P is a reasonable number of possible clusters.

Then, they calculate the *Davies-Bouldin* (DB) *index* [17] for all sets S_k. The DB index is a cluster separation measure evaluating the overall significance of a cluster set. It is possible to calculate the similarity measure R_{ij} between clusters C_i and C_j by the following equation:

$$R_{ij} = \frac{D_i + D_j}{M_{ij}}, \tag{3.7}$$

where D_i is the average distance between each point of C_i, and M_{ij} is the distance between the centroids of C_i and C_j. Therefore, a high value of R_{ij} indicates that C_i and C_j describe a similar pattern. Contrarily, a low value of R_{ij} induces that C_i and C_j describe different patterns. Hence, the DB index can be calculated as:

$$DB = \frac{1}{k} \sum_{i=1}^{k} \max_{i \neq j} R_{ij} , \tag{3.8}$$

where $\max_{i \neq j} R_{ij}$ corresponds to the measure R_{ij} for C_i and its most similar cluster.

Once the DB indices are computed for each set S_k, we select the one set for which the index is the smallest because it indicates a better cluster separation. However, the chosen set might still contain clusters for which the data distribution is similar. In other words, they denote the same behavior in the network and will be difficult to set apart.

The next step is to use the *Kolmogorov-Smirnov* (KS) *test* [41] to verify the null hypothesis that two arrays of sample observations are from the same distribution. The p-value obtained by the test [40] determines the probability of having samples that confirm the null hypothesis. Hence, we apply the KS test to each pair of clusters for each KPI. If any pair of clusters for which any p-values obtained for their KPIs is greater than a certain threshold called *significance level* (SL), the null hypothesis cannot be rejected (i.e., the two clusters are statistically similar). In case there is such a pair in the previously chosen set S_k, we select the set S_{k-1} and perform the test again until there is no similar pair of clusters anymore and $k \geq 2$.

3.2.2 OPTICS clustering

OPTICS (Ordering Points To Identify the Clustering Structure) [6] is an algorithm for finding density-based clusters. It is based on DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [20], one of the most popular density based clustering algorithm [43]. The main advantage of OPTICS over DBSCAN is that it can detect clusters of different density.

Another advantage appears when choosing the parameters. Both algorithms use two parameters: ε is the distance within which a minimum of $MinPts$ points must be located to be considered a cluster. While, for DBSCAN, ε is a fixed distance, OPTICS sees it as a maximum distance from which the closest $MinPts$ points are selected. The distance parameter can also be omitted by considering it infinite, but it largely affects the complexity and computation time of the model. The parameters of OPTICS have a significant influence on its outcome. Consequently, the simplification of the parameter selection is even more important due to the limited knowledge on the input data.

OPTICS essentially creates an ordering of the points based on two measures. The core distance of a point p is

$$\text{core-dist}_{\varepsilon,MinPts}(p) = \begin{cases} \text{Undefined} & \text{if } |N_\varepsilon(p)| < MinPts \\ MinPts\text{-distance}(p) & \text{otherwise,} \end{cases} \quad (3.9)$$

where N_ε is the set of points within the distance ε of p, and $MinPts\text{-distance}(p)$ stands for the distance between p and the $MinPts^{\text{th}}$ closest point of $N_\varepsilon(p)$. The core distance is undefined if there are not enough points (fewer than $MinPts$) within the ε-neighborhood. Consequently, if it is defined, the point p becomes a core point.

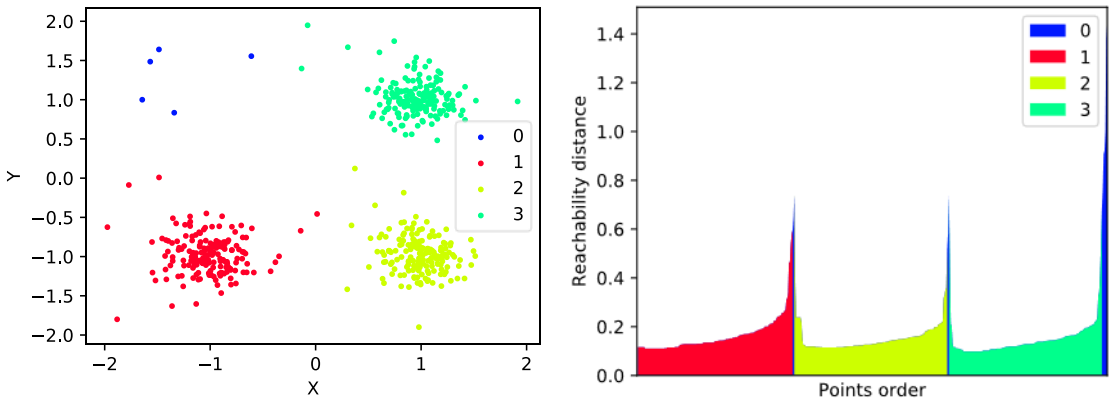
Then, the reachability distance of a point q with respect to p is calculated as follows:

$$\text{reach-dist}_{\varepsilon,MinPts}(q,p) = \begin{cases} \text{Undefined} & \text{if } |N_\varepsilon(p)| < MinPts \\ \max\left(\text{core-dist}_{\varepsilon,MinPts}(p), ||q-p||\right) & \text{otherwise.} \end{cases} \quad (3.10)$$

In case p is not a core point, the reachability distance of q w.r.t. p is undefined. Otherwise, it is equal to the maximum value between the core distance and the distance between p and q.

The algorithm processes each point once. The core-distance of a point is first calculated. If defined, the algorithm computes the reachability distance of its neighborhood and proceeds to the point with the next smallest reachability distance. Points are added to a priority queue in the ascending order of their reachability distance. When there are no more points reachable from the current queue, the next unprocessed point from the dataset is selected and the content of the queue is moved to an ordered list. The algorithm iterates until all points have been processed.

The result of the algorithm is the "cluster ordering" of the input data and can be visualized in a reachability-plot (Figure 3b). Unlike DBSCAN, OPTICS does not extract clusters automatically. Among the existing algorithms, we use the most popular one: the ξ cluster extraction as proposed by the authors of OPTICS [6].



(a) (b)

Figure 3. *OPTICS with ξ cluster extraction on 3 Gaussian clusters with added noise. (a) the scatter plot and (b) the reachability plot. Colors denote different clusters. The points labeled 0 were not clustered. ξ=0.05.*

This method is named after its parameter ξ which denotes a steepness percentage. The main principle is to choose where a cluster starts and ends using the steepness of the reachability plot. Typically, a cluster starts with a downward slope and ends with an upward one (Figure 3b). A higher value of ξ significates the slopes need to be steeper in order to be a cluster limit. Hence, a high ξ value will make the algorithm detect the most significant clusters i.e. the larger "valleys" in the reachability plot. On the contrary, lower values of ξ will produce the detection of less significant clusters.

3.2.3 Growing neural gas with post-pruning

Growing Neural Gas (GNG) [24] is an alternative to SOM as a vector quantization method. Similar to SOM, it conserves the topology of the initial dataset. However, knowledge on the input data is not as important because GNG requires neither the knowledge of the exact number of neurons, nor an a priori shape as parameters to use it as a clustering algorithm.

GNG includes nodes s_i associated with vectors w_{s_i} and non-weighted edges to describe the topology of the data. At each iteration, the algorithm generates a vector x from a probability density function of the data $P(x)$. The nodes are updated so that the closest node s_i to x (the winner node) and its direct neighbors $n \in N_{s_i}$ will move towards the vector x. The direct neighborhood N_{s_i} consists of nodes connected to s_i by one edge. Therefore, the nodes are updated by the following rule:

$$w_{s_i}(t+1) = w_{s_i}(t) + \varepsilon_a \cdot \left(x - w_{s_i}(t)\right) \text{ and} \tag{3.11}$$

$$w_n(t+1) = w_n(t) + \varepsilon_b \cdot \left(x - w_n(t)\right), \tag{3.12}$$

where ε_a and ε_b are the learning rates for the winner node and the direct neighbors, respectively.

Then, a new edge is created between the winner node and the second-nearest node to x. In addition, new nodes are added every λ iterations between two other nodes to decrease the quantization error.

Edges and nodes are eliminated based on an edge aging system. The age of an edge increments whenever it is connected to the winner node of the iteration. Then, the age is reset to 0 if it was supposed to be created during the current iteration. Otherwise, an edge is removed if it is older than α_{max} iterations. A remaining node is also removed if left without any edge. This system is necessary as updating nodes may invalidate the edges created previously.

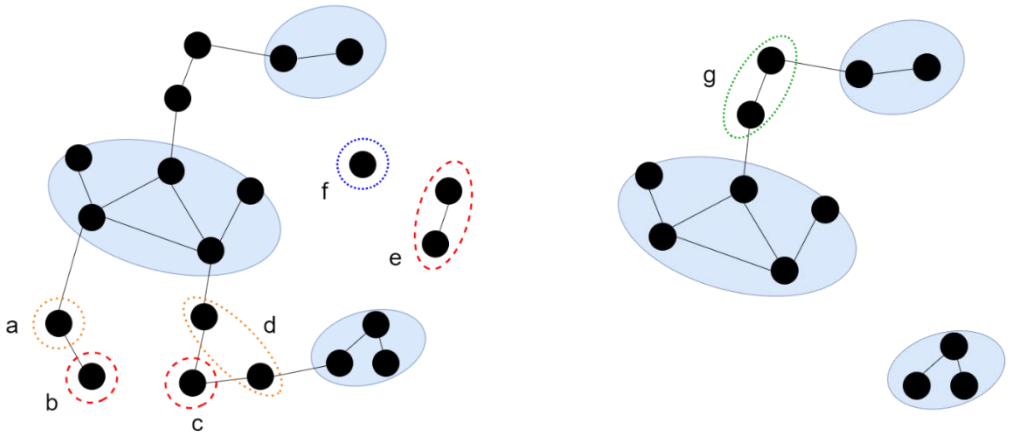
One risk with GNG is that the few last training vectors x chosen from $P(x)$ do not represent the input data well enough. Consequently, the model may not truly represent the dataset either. To mitigate this issue, we use a variant of GNG called Growing Neural Gas with Post-Pruning (GNG-PP) [13] which eliminates and repositions nodes to reduce the quantization error. Most importantly, the post-pruning process eliminates nodes that are not representative of the data and would otherwise be counted as clusters.

The algorithm first assigns each data point to its closest node using the Euclidean distance. This step allows us to separate the nodes in two sets: the ones which have data

points assigned to them, and those which have none. We assign these latter nodes to a set V. The nodes in V are not useful for reducing the quantization error but can remain useful for their connections to other nodes. Considering the different groups of nodes in Figure 4, a set of rules determines their outcome:

1. A node $v \in V$ and its possible edges are removed from the graph if:
    a. v has no edges – Group f,
    b. All the edges of v connect to other nodes in V – Groups b, c, and e, or
    c. It is connected to only one useful node $s \notin V$ – Groups a and d.
2. Otherwise, v is not removed if is connected to more than one useful node reachable within 2 edges – Group g.

Note the step 1c is modified from the original method where v is supposedly relocated in a zone that would decrease the quantization error. This step is not useful for clustering because it affects neither the number of clusters, nor the data items that are assigned to a cluster. Therefore, the method can be simplified with our modification.



(a) (b)

Figure 4. *A Growing Neural Gas graph before (a) and after (b) pruning. The shaded areas represent an input data distribution. V consists of all nodes outside of these areas. The circled groups correspond to different cases considered by the algorithm for pruning the graph.*

Once a graph is pruned (Figure 4b), the next step is to apply a clustering algorithm. While it is possible to use a traditional method like K-means to cluster a graph for SOM, this is not convenient for GNG-PP. For SOM, all the nodes are connected to their neighbors and form a graph with only one component. In contrast, GNG-PP produces a graph with potentially several components which can be interpreted as separate clusters. Hence, we apply a simple algorithm to mark all nodes part of the same connected component as one cluster. The data items are then assigned the same cluster as their closest node.

4. EXPERIMENT SETUP

After the methods are introduced, we implement a comparison process for the selected clustering methods as depicted in the Figure 5. The input dataset is first presented in Section 4.1. and then prepared for clustering through a pre-processing method described in Section 4.2. Then, the different combinations of dimensionality reduction and clustering algorithms are detailed in Section 4.3. Next, Section 4.4 introduces the hyperparameter optimization method used to carry out the best results from the clustering methods. Further, the evaluation process and metrics are explained in Section 4.5. Finally, the software tools used for the implementation are listed in Section 4.6.

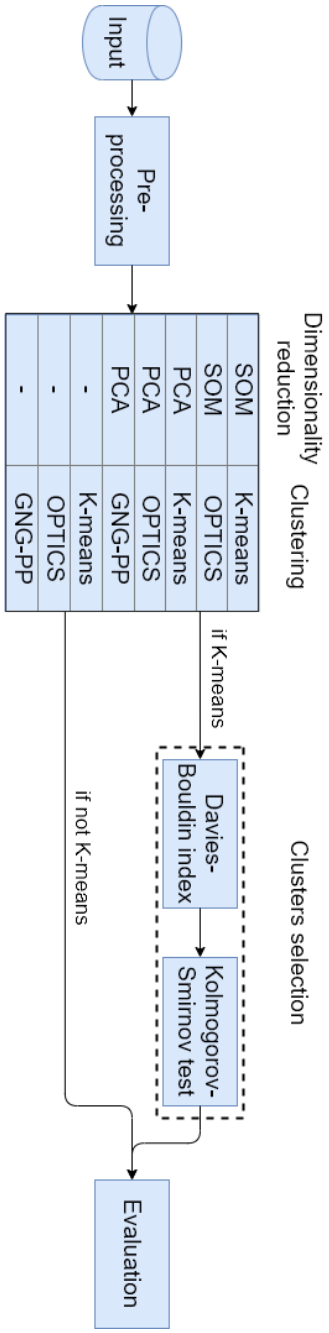


Figure 5. *Illustration of the implementation.*

4.1 Dataset

The dataset used in this study represents LTE mobile network measurements. LTE is a specification of mobile networks globally. Its architecture is based on three main components. The *User Equipment* (UE) represent the mobile phones, tablets, or any device that can connect to the network. The *Evolved Packet Core* (EPC) corresponds to multiple modules handling the connections and the data traffic. Finally, the *Evolved Universal Terrestrial Radio Access Network* (E-UTRAN) is the air interface system allowing UEs to

connect to the EPC [48]. The system is composed of *Evolved Node Bs* (eNodeB), which correspond to the hardware allowing the connections (Figure 6).

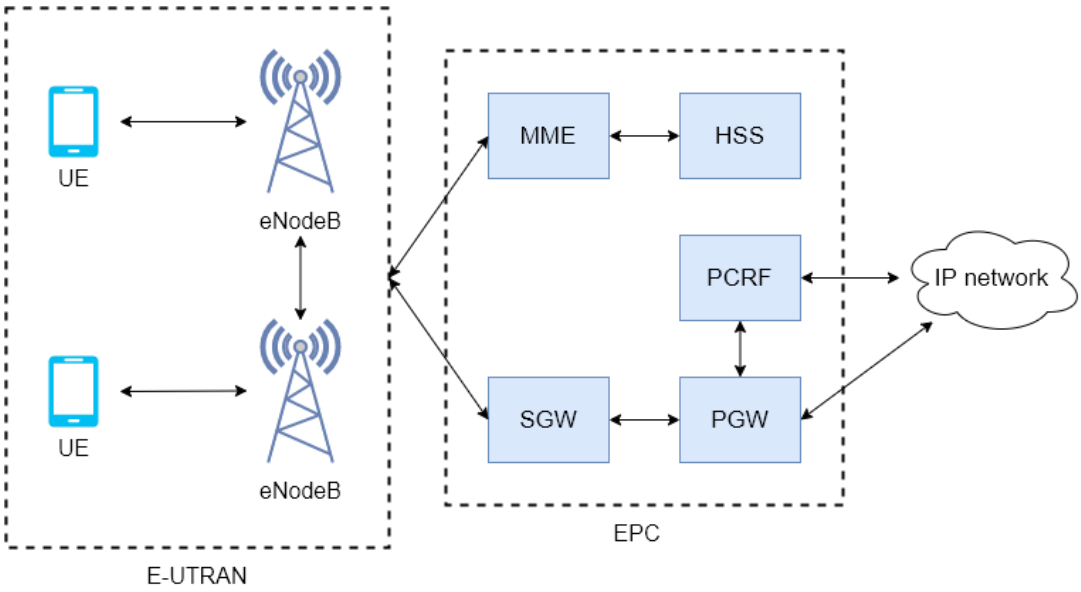


Figure 6. *LTE network diagram, adapted from* [11].

The dataset consists of about 6 million timestamped data points with 133 features recorded for 4.5 hours in a European medium-sized city. All the measurements were triggered by a connection establishment or release event between an eNodeB and a UE, following monitoring specifications from 3GPP (3rd Generation Partnership Project). The release and establishment events also include those which are related to *handovers* between LTE cells. A handover operation corresponds to the transfer of the connection between a UE and an eNodeB to another one, which generally has a better signal. This operation usually happens when the user moves away from an antenna and closer to another. In addition, a total of 319 *network cells* were monitored in this dataset. A cell represents a land area served by at least one radio station.

4.2 Pre-processing

While a measurement contains important information regarding the quality of the connection, not all the 133 dimensions are useful for detecting patterns for troubleshooting the network. For example, the dataset contains multiple columns corresponding to identification numbers or other specific measurements unrelated to the health of the network. In addition, a problematic behavior in a mobile network can rarely be assessed with a single data sample. However, patterns can be found in a collection of data points recorded over a certain amount of time.

Finally, the large volume of the input data would translate to a high computation cost for the selected algorithms because they read data as input and, consequently, the execution time of an algorithm may be significantly slower.

For all these reasons, to improve the results and reduce the computation time, we decide to pre-process the data before applying the clustering methods.

4.2.1 Aggregation and feature selection

First, as this study aims at supporting future RCA applications, we need to be able to locate cells for which a potential cause is found. To solve this problem, the data points are grouped cell-wise to obtain multivariate time-series for each cell. Then, we aggregate the data cell-wise on a certain time-window. The choice of the time-window offers essentially different levels of granularity and computation cost.

During the aggregation, we collect several KPIs based on expert knowledge, in order to analyze the air interface coverage and quality related issues:

- *RSRP 5th percentile*: The Reference Signal Received Power is measured in decibel milliwatts (dBm) and correspond to the strength of the signal received by a UE, in the downlink direction. We select the 5th percentile of the measurements as most of them are relatively high and we seek to focus on problematic cells.
- *RSRQ 5th percentile*: The Reference Signal Received Quality is also measured in decibel milliwatts. It denotes the quality of the signal received by a UE. Like the RSRP, this value is measured in the downlink direction.
- *SINR 5th percentile*: The Signal to Interference plus Noise Ratio describes the quantity of the signal that is actually interpretable in comparison to the received interference and noise. This value is measured in the uplink direction by the eNodeB.
- *Accessibility*: This measure denotes the ratio of successful connections from a UE to the network over all connection attempts.
- *Coverage*: It is the ratio of measurements representing a sufficiently high coverage over all measurements. Hence a UE has a high coverage if the source cell (the one the UE is connected to) RSRP is high enough (more than -115dBm) or if it has a neighbor cell it can connect to with a high enough RSRP value (more than -112dBm).
- *Interference*: A neighboring cell to the UE is considered interfering with the source cell if it operates on the same radio frequency and if the RSRP of both cells are

similar (less than 5dBm of difference). Therefore, the interference KPI corresponds to the ratio of the interfering neighboring cells out of all the measured neighboring cells.

4.2.2 Statistical standardization

The dataset contains features sampled from different distributions. For example, a ratio from 0 to 1 for Accessibility and decibel milliwatts from about -140dBm to -44dBm for RSRP. Standardization is a solution to prevent different feature ranges and mean values to influence the comparison. Therefore, the dataset is standardized with the z-score formula (4.1) so that each feature set of values has a mean equal to 0 and a standard deviation equal to 1:

$$z = \frac{x - \bar{x}}{s}, \tag{4.1}$$

where z is the z-score, x is a sample value of a feature set F, $\bar{x}$ denotes the mean of F and s the standard deviation of F [34].

4.3 Combination of methods

The comparison conducted in this study applies to a set of combinations of the presented dimensionality reduction algorithms and clustering methods. The set is listed in Figure 5.

In this study, the codebook of vectors generated by the SOM is used as input for both K-means and OPTICS. However, GNG-PP is not applied to SOM as they both perform a form of vector quantization. This would increase the quantization error significantly as the number of inputs is considerably reduced. Therefore, we consider that comparing both methods separately is more relevant.

PCA is used in combination with all three clustering methods presented. Contrary to SOM, PCA does not reduce the number of data points but only the dimensions. Therefore, GNG-PP can efficiently perform vector quantization before clustering.

Finally, evaluating the performance of using a dimensionality reduction algorithm before clustering can be of interest. Although the models are exposed to the curse of dimensionality, the dataset has a rather low (6) number of dimensions. Then it makes sense to compare the clustering methods on the pre-processed dataset as well.

4.4 Hyperparameter optimization

In machine learning, hyperparameters are parameters that are set to an algorithm before training. Consequently, choosing the right hyperparameters for an algorithm can drastically change its output.

While hyperparameters can be chosen manually, it is not always the optimal method because small adjustments may have a large impact on the output. This effect can make hyperparameter selection a difficult task for humans and it applies to many of the hyperparameters used in the clustering methods compared in this study. Therefore, we use a hyperparameter optimization algorithm in order to obtain better performances than if we choose the hyperparameters manually.

There are several approaches to hyperparameter optimization. First, *Grid Search* can be seen as a “brute force” method. Its procedure is to test all combinations of subsets of the hyperparameter space and computing a performance measure for each combination. One of the main issues of Grid Search is its exponentially increasing computation time when adding more hyperparameters.

Random Search is a variant of Grid Search. It replaces the fixed set of hyperparameters combinations by choosing them randomly and it has been shown to outperform Grid Search [8]. However, other methods have been shown to perform better and converge faster than Random Search, including *Bayesian optimization* [9].

Bayesian optimization is a method that builds a probabilistic model of an unknown function and attempts to maximize a target variable. In our case, the unknown function is a clustering method and the target is a classification performance metric. Therefore, the Bayesian optimization algorithm takes ranges of hyperparameters as input and will compute the unknown function with different parameters combinations. The algorithm learns from its experiences and will select the next combination of parameters based on the ones that have already been attempted.

Hence, there is one function per clustering method which hyperparameters need to be optimized. One function will apply a clustering method on each set of time-window aggregations. In other words, one set corresponds to the aggregation of data points by network cell for a certain time-window, as explained in Section 4.2.1. This is done so that, in a real situation, clustering can be achieved in a reasonable amount of time on a localized set of data incoming from the network monitoring tools.

Then, the classification performance metric corresponds to the median of all *Silhouette values* calculated for each set. Silhouette [51] is a method to measure the quality of

clusters of data. The Silhouette value is calculated based on the similarity of a data point to its own cluster and to other clusters. It is comparable to the Davies-Bouldin index used for cluster selection using K-means clustering. However, we choose Silhouette here as it describes the results of clustering better: the Silhouette value ranges from -1 to 1, where a value closer to -1 means data points are wrongly clustered (they should be part of another cluster) and 1 means they are correctly clustered. In comparison, the Davies-Bouldin index is ranged from 0 to the infinite and a small index corresponds to a better clustering.

In addition, the median is used here instead of the mean of all Silhouette values. This is because we assume a clustering method is not only good if the average performance is high, but also if the algorithm performs consistently well. Therefore, the median offers a better metric to be maximized by the Bayesian optimization method.

4.5 Evaluation

It is well known that there is no “free lunch” in machine learning [61], [62]. In other words, there is no specific algorithm that can solve all problems, but rather algorithms that are better suited for one or another use case. Therefore, we propose a set of four metrics to evaluate the most relevant clustering method to use depending on the requirements of the user.

First, the *classification performance*. The most important metric is the performance of the method itself, or in other words, the quality of the clusters. Diverse measures have been introduced in Sections 3.2.1 (DB index) and 4.4 (Silhouette). After several experiments, we decide to use the Silhouette as it describes better the quality of the clusters with manual verification. In particular, the classification performance corresponds to the mean of Silhouette values obtained from clustering the time-windows subsets one at a time.

Then, the *consistency* of a method is measured. This metric allows us to evaluate how the quality of the results varies from one input to another. Therefore, the consistency corresponds to the standard deviation of the Silhouette values of the time-window subset clustering. This way, in combination with the classification performance metric, we make sure the evaluated algorithm has a similar performance on different input distributions and not only for one fortunate situation.

Next, the *computation time* of a method is calculated. Depending on the use case, it may be important to have a trade-off between computation time and performance. This is measured as the average time used to process a clustering method (dimensionality reduction and clustering included) over the clustering of all time-window subsets.

Finally, we measure the *sensitivity* to hyperparameter tuning of an algorithm. The methods that are analyzed in this study present different hyperparameters. The latter are often essential to obtain the best possible output using the method, as described in Section 4.4. Therefore, the sensitivity metrics aims at measuring how difficult it is to find the optimal hyperparameters for a method, as they may change based on input. For this matter, we calculate the sensitivity s as formulated in [36] by the following:

$$s = \sqrt{\frac{\sum_{x \in Q}(f(x, D) - p(x, D))^2}{|Q| - 1}}, \tag{4.2}$$

where Q is a subset of all possible hyperparameter combinations for a method m, f is the evaluation function of m, p denotes the classification performance, and D stands for the collection of time-window subsets. In our case, f correspond to the calculation of the Silhouette value. Then, in order to obtain a measurement, we must perform Grid Search as introduced in Section 4.4, where Q represents the set of parameter combinations to process.

In addition to these measurements, we aim at identifying the meaning of the clusters obtained from the different methods in order to evaluate their capacity to detect relevant patterns. Therefore, we present a set of *probability density functions* (PDFs) for each KPI and for each cluster to experts so they can annotate the different detected clusters. These annotations are useful to determine the diversity of clusters a method can detect.

Further, the PDFs are estimated with *kernel density estimation* (KDE), which provides smoothed functions based on random samples. KDE requires a bandwidth parameter h to control the smoothness of the estimation, which considerably influences the results. While several techniques for choosing the right bandwidth h to provide smooth functions which can represent the data distribution accurately, this study uses Silverman's rule-of-thumb [53] calculated the following way:

$$h = 0.9 \cdot \min\left(\sigma, \frac{IQR}{1.34}\right) \cdot n^{-\frac{1}{5}}, \tag{4.3}$$

where σ is the standard deviation of the samples, IQR is the interquartile range, and n is the number of samples.

4.6 Software

In order to implement the different algorithms and compare their efficiency, a set of software tools were used. The project is implemented in the Python programming language with several libraries:

- NumPy [46] is the fundamental package for scientific computing. The project implements it to manipulate matrices of data and perform mathematical operations such as sorting, reshaping, averaging, calculating percentiles, etc.
- SciPy [57] is built on top of NumPy and provides advanced statistical and mathematical functions. It is used for computing the KS test and the Pearson correlation coefficient.
- Pandas [42] provides data structures and data analysis tools. Its use in the project covers data extraction and removing inadequate items.
- Scikit-learn [12] includes various machine learning features. The project uses its implementation of K-means clustering, DB index, OPTICS, and sample datasets generation.
- SOMPY [44] is a library for SOM. Its structure is similar to the official MATLAB SOM Toolbox [32]. Small corrections have been made to correspond to the needs of this project.
- NeuPy [52] implements various neural network models. The training of GNG models happens through this library.
- Matplotlib [29] is a popular 2D plotting library. It is used throughout the entire project for all generated charts such as scatter plots, bar charts, SOM, etc.
- Bayesian Optimization [45] is a library for – as the name indicates – computing Bayesian optimization. It implements one of the most popular method for Bayesian optimization, using Gaussian processes.

5. RESULTS AND DISCUSSION

This chapter presents the results obtained in this study and their analysis.

The first step of the experiment is to prepare the dataset. As previously mentioned, we aggregate the input data by subsets of a certain time-window. The choice of the time-window affects the granularity of the analysis. Due to the measurements being recorded for a short amount of time (4.5 hours), we decided to select a small time-window in order to keep a significant number of data points to process. Therefore, we chose a 10 minutes time-window for this analysis, as it provides a reasonable amount of data and computation time. In addition, it is also expected that possible network events can be detected within 10 minutes aggregations, based on expert knowledge.

Before clustering, we performed a hyperparameter optimization procedure using Bayesian optimization. The latter method requires bounds values – the minimum and maximum values – for each parameter to optimize as input. Hence, the parameter bounds of the different clustering methods, as well as the results obtained with Bayesian optimization are described in Table 1 to Table 5. Additionally, the hyperparameters values that are set manually based on trial-and-error and visual inspection can be found in the below tables for comparison purposes. Note that the parameter bounds were chosen based on the default values given by the software packages detailed in Section 4.6, as well as the methods documentations provided with the software tools, and trial-and-error experiments.

In addition, some of the combinations do not need hyperparameter optimization since no hyperparameters need to be set manually, namely for the combinations *None* – K-means and PCA – K-means. Here, and in the remainder of the document, *None* stands for a method that is not using any dimensionality reduction algorithm. In addition, some parameters are calculated following the instructions of the literature, as for the map width and map length of the SOM, which follow the recommendations in [31].

Table 1. *Hyperparameter optimization bounds and results for SOM – K-means.*

	Rough training epochs	Fine training epochs
Minimum	25	12
Maximum	60	45
Optimized	50	19
Manually chosen	25	30

Table 2. *Hyperparameter optimization bounds and results for None – OPTICS and PCA – OPTICS.*

	MinPts	ξ
Minimum	2	0.005
Maximum	20	0.5
Optimized	2	0.005
Manually chosen (*None* – OPTICS)	4	0.19
Manually chosen (PCA – OPTICS)	10	0.13

Table 3. *Hyperparameter optimization bounds and results for SOM – OPTICS.*

	Rough training epochs	Fine training epochs	MinPts	ξ
Minimum	20	15	2	0.005
Maximum	60	50	20	0.5
Optimized	53	49	4	0.054
Manually chosen	25	30	4	0.05

Table 4. *Hyperparameter optimization bounds and results for None – GNG-PP.*

	Epochs	α_{max}	λ	Max. nodes number	ε_a	ε_b
Minimum	15	10	5	70	0.1	$1.0 \cdot 10^{-4}$
Maximum	40	20	100	150	0.5	0.1
Optimized	36	20	18	102	0.5	0.1
Manually chosen	9	6	60	100	0.03	0.004

Table 5. *Hyperparameter optimization bounds and results for PCA – GNG-PP.*

	Epochs	α_{max}	λ	Max. nodes number	ε_a	ε_b
Minimum	15	10	5	70	0.01	$1.0 \cdot 10^{-5}$
Maximum	40	20	100	150	0.5	0.01
Optimized	21	20	71	118	0.01	0.01
Manually chosen	13	8	45	150	0.08	0.004

Further, we analyze the results of the different methods in the first three following sections using the evaluation metrics results in Table 6 and the different clusters that were identified by experts in Table 7. The methods are presented in the order of ascending theorical complexity: K-means-based methods, then OPTICS-based methods, and finally GNG-PP-based ones.

Table 6. *Evaluation metrics results.*

	Classification performance (mean)	Consistency (standard deviation)	Sensitivity	Computation time (s)
***None* – K-means**	0.43	0.20	-	0.47
PCA – K-means	0.47	0.14	-	0.48
SOM – K-means	0.39	0.05	0.038	8.17
Average	0.43	0.13	0.038	3.04
***None* – OPTICS**	-0.03	0.04	0.174	0.11
PCA – OPTICS	0.26	0.03	0.245	0.12
SOM – OPTICS	0.07	0.14	0.195	11.74
Average	0.10	0.07	0.205	3.99
***None* – GNG-PP**	0.47	0.20	0.194	1.81
PCA – GNG-PP	0.39	0.18	0.252	1.20
Average	0.43	0.19	0.223	1.51

Table 7. *Clusters identified by experts. Low, Medium, and High correspond to equally distributed ranges of values relative to the KPI data distribution.*

	RSRP	RSRQ	SINR	Accessibility	Coverage	Interference
Low power / Performance degradation	Low	Low or Medium	Low	High	High	Medium
Low coverage	Low or Medium	Low or Medium	Low or Medium	High	Low or Medium	Low
Good / Normal	High	High	High	High	Medium or High	Low
Inter-site interference	Medium	Medium	High	High	Medium or High	Medium
Medium downlink / high uplink	Medium	Medium	High	High	Medium	Medium or High

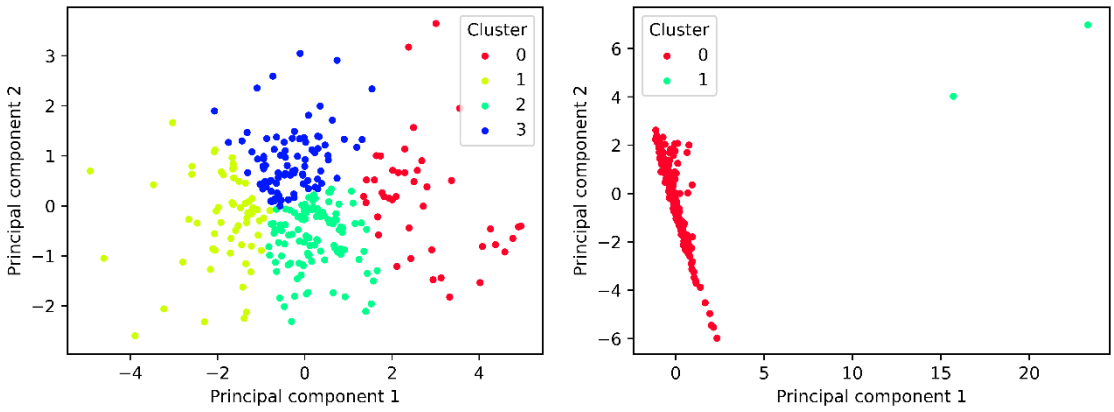
5.1 K-means-based methods

Several key points can be observed from the evaluation metrics results in Table 6. First, K-means-based methods have one of the best classification performances on average (0.43) in comparison to the other two categories (0.10 for OPTICS and 0.43 for GNG-PP). Then, K-means has a low consistency (high Silhouette standard deviation) and the lowest sensitivity to hyperparameter tuning, on average in comparison with the other methods. Finally, the computation time of K-means itself is fast with PCA (0.48s) and *None* (0.47s). Running SOM before K-means considerably increases the computation time (8.17s). This phenomenon is also observed for OPTICS and is mainly due to the training process of SOM.

Therefore, the sensitivity is only calculated for SOM – K-means because no hyperparameter needs to be optimized for the other methods. For this method, only the parameters of SOM are optimized (Table 1) and concern the number of training epochs of the algorithm. The minimum values for Bayesian optimization are set following the instructions for a sufficient training in [31], and thus longer training procedures do not affect the results of SOM significantly, explaining the low sensitivity.

Then, the high standard deviation can be deduced by two assumptions: (1) K-means clusters tend to cover the same size of the data space. As detailed in Section 3.2.1, the algorithm splits the data in clusters such that each data point is assigned to the closest vector μ_i, which represents the mean of all data points associated to it. Therefore, the data points tend to be equally distributed over the data space – not to be misinterpreted

with the number of data points per cluster – in separate clusters, as depicted in Figure 7.



(a) (b)

Figure 7. *Clusters plots using PCA – K-means. (a) represents the worst cluster set (Silhouette value = 0.33). (b) represents the best cluster set (Silhouette value = 0.91).*

Then, the second assumption (2) is that the data space can be considerably different from a time-window subset to another. An example of this phenomenon can be observed in Figure 7 for PCA – K-means. PCA offers a 2D representation of the dataset and generally forms two kinds of data shapes: one represents a blob shape where data points are denser in the center and are more distant between each other as they are farther from the center, as in plot (a). The second typical shape is an elongated blob and constituted of a few outliers (b). Consequently, some cluster sets result with a high Silhouette because clusters are well separated, while others perform worse because the data points are spread around a central point.

Further, by analyzing the PDFs obtained from the clusters, we can deduce that the identification of clusters computed by K-means with SOM and *None* is limited. This limitation can be explained by the low number of detected clusters for the methods using SOM or *None*: 2.3 and 2.1 clusters on average, respectively. As a consequence, very few patterns can be identified with these methods. In addition, the behavior of K-means to detect clusters with a similar data space size makes clusters cover large areas (because of their low number). Therefore, clusters likely contain data points that can be distant from others in the same cluster. Consequently, this phenomenon increases the variance of the PDFs and makes clusters more difficult to distinguish.

Therefore, observations show that methods using SOM and *None* can separate a dataset in two clusters most of the time: a *Low power / Performance degradation* cluster

and a *Good / Normal* cluster (as defined in Table 7). This classification is made by observing that both clusters cover the same ranges of KPIs over different time-window. An example using SOM – K-means is depicted in Figure 8. Here, the cluster 0 has a higher coverage, higher RSRP, higher RSRQ, higher SINR and lower interference (i.e. the *Good / Normal* cluster) than the cluster 1 (i.e. the *Low power / Performance degradation* one). These two clusters can be observed regularly over different time-window subsets.

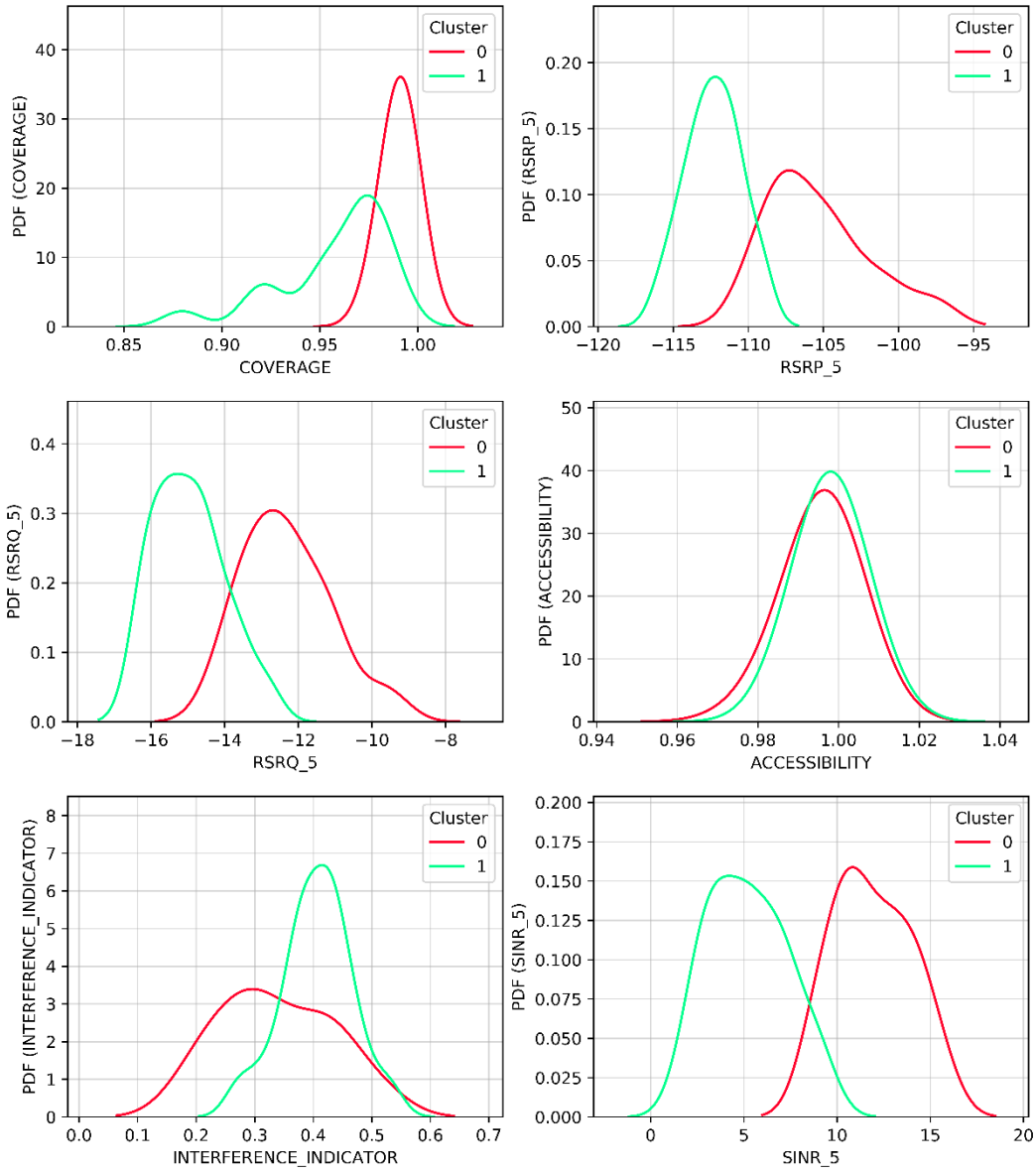


Figure 8. *Sample PDF plots for each KPI obtained using SOM – K-means. Two clusters can be identified.*

Then, in the case of PCA, the average number of detected clusters is higher (3.6) than with the two other solutions. Therefore, we conducted clusters annotation with expert on PCA – K-means. First, we observe that a large number (91%) of the obtained clusters have enough points to be considered reliable. We consider a cluster reliable if it is associated with more than 1% of the data points, which corresponds to strictly more than 3

points on average. Then, this method can also detect *Low power / Performance degradation* and *Good / Normal* clusters in most of obtained the cluster sets, where they are detected together in 93% of the sets, as reported in Table 8. However, in contrast to SOM and *None*, which can almost only detect the two latter causes, the method using PCA can find more patterns, such as *Low coverage* and *Inter-site interference*.

Table 8. *Presence of identified clusters in all time-window subsets for PCA – K-means. Unreliable clusters (with less than three data points) are excluded.*

	Low power / Performance degradation	**Low coverage**	**Good / Normal**	**Inter-site interference**	**Medium downlink / high uplink**
PCA – K-means	100%	37%	93%	19%	0%

One major drawback of clustering with K-means is the presence of overlapping PDFs between clusters. This problem affects the clusters identification since several clusters can be assigned the same cause, which could be problematic for further use of this method, in an automatic identification system for example.

This phenomenon is depicted in Figure 9. We can observe that clusters 0 and 1 present similar PDFs for all the KPIs, and consequently a similar cause: *Low power / Performance degradation*. This issue can usually be observed for blob-shaped data (as for the example in Figure 9), which therefore can be explained by the first assumption (1).

In the case of clustering elongated blobs, clusters may show different behaviors between a group of outliers and the main blob. However, they are rarely well separated and a cluster of outliers usually contains data points from the main blob as well.

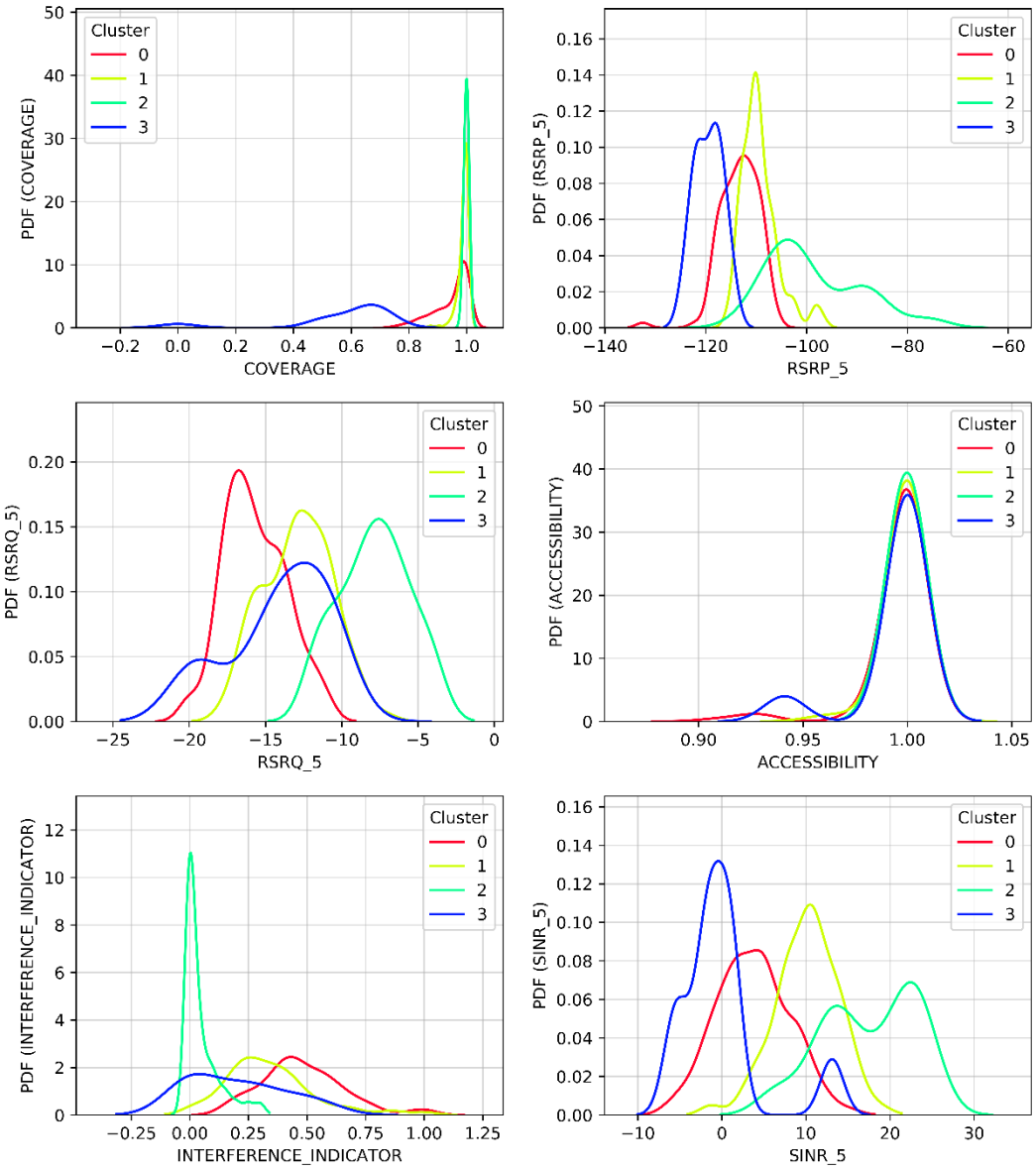


Figure 9. *Sample PDF plots for each KPI obtained using PCA – K-means.*

As a conclusion, K-means-based methods offer a solution that is fast to implement and compute. In combination with PCA, K-means can detect two main patterns in addition to a few smaller causes. However, the identification of these clusters can sometimes be difficult due to the large variance of feature PDFs caused by the tendency of K-means-obtained clusters to cover equally sized areas in the data space.

5.2 OPTICS-based methods

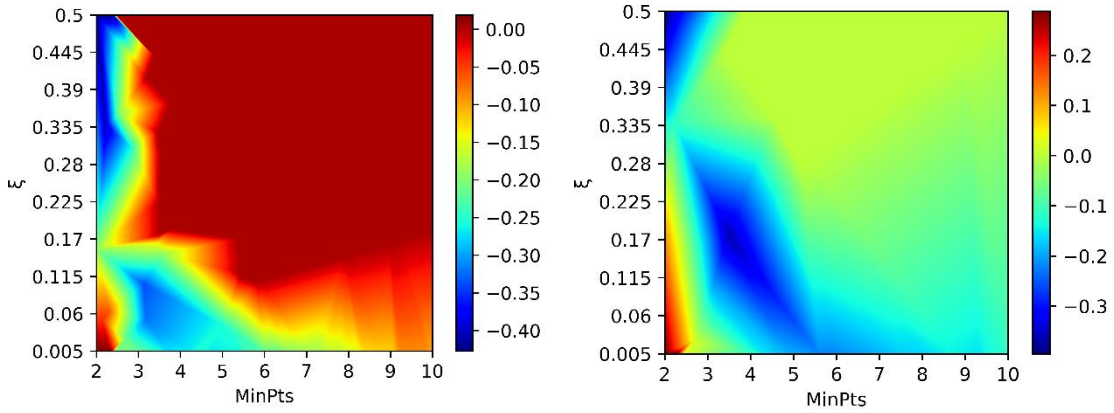
OPTICS clustering offers the worst classification performance on average out of all three clustering algorithms (Table 6). In particular, *None* – OPTICS classification performance is the lowest out of all nine combinations (-0.03). However, regardless of the low values for this metric, OPTICS clustering highlights limitations of Silhouette.

First, the average number of clusters detected using *None* (69.9) or PCA (88.8) is excessively high with relation to the number of input vectors (292 on average) and to potential network faults. On the one hand, for PCA – OPTICS, the classification performance (0.26) remains comparable to the results obtained for K-means and GNG-PP. This can be explained by the way the Silhouette value is calculated: a high Silhouette reflects dense clusters (1) that are distant from each other (2). This is the case for *None* and PCA: clusters that contain few data points and are dense (1), and clusters are well separated (2).

On the other hand, the classification performance using *None* is close to zero (-0.03) which may be explained by the curse of dimensionality. The differences in distance between the data points being reduced with higher dimensions, clusters can be neither as dense as for PCA, nor as distant from each other.

This low performance can be explained by the hyperparameters obtained with Bayesian optimization (Table 2). Here, $MinPts = 2$ and $\xi = 0.005$ i.e. the lowest possible values that could be obtained w.r.t the parameter bounds. With these parameters, OPTICS will be able to detect clusters with a minimum of 2 points and the low value for ξ means the algorithm will detect clusters with little significance – as explained in Section 3.2.2.

The mapping between the hyperparameters and the target values (median Silhouette) obtained during Bayesian optimization is depicted in Figure 10. While the maps show a convergence to the previously used hyperparameters, there are other combinations which offer similar results, especially for *None* – OPTICS.



(a) (b)

Figure 10. *Bayesian optimization maps for None – OPTICS (a) and PCA – OPTICS (b). The color range denotes the linearly interpolated median Silhouette values obtained from clustering.*

We can compare the results obtained for these two methods with Bayesian optimized parameters against the manually chosen ones (Table 2). In the latter case, the number

of clusters on average is considerably reduced for *None* (3.2) and PCA (2.4). In addition, their classification performance is lower (-0.06 for *None* and 0.23 for PCA) and their standard deviation increased considerably: 0.27 and 0.42, respectively. However, regardless of the lower classification performance and higher standard deviation, clusters can be identified with more ease because their number is lower, and they contain more data points.

In Figure 11, we can observe the PDFs of a sample clustering set obtained with PCA – OPTICS in which the clusters -1 and 0 present PDFs that are very similar. This is confirmed by the causes identified by experts, since we can annotate both cluster -1 and 0 to *Low power / Performance degradation*.

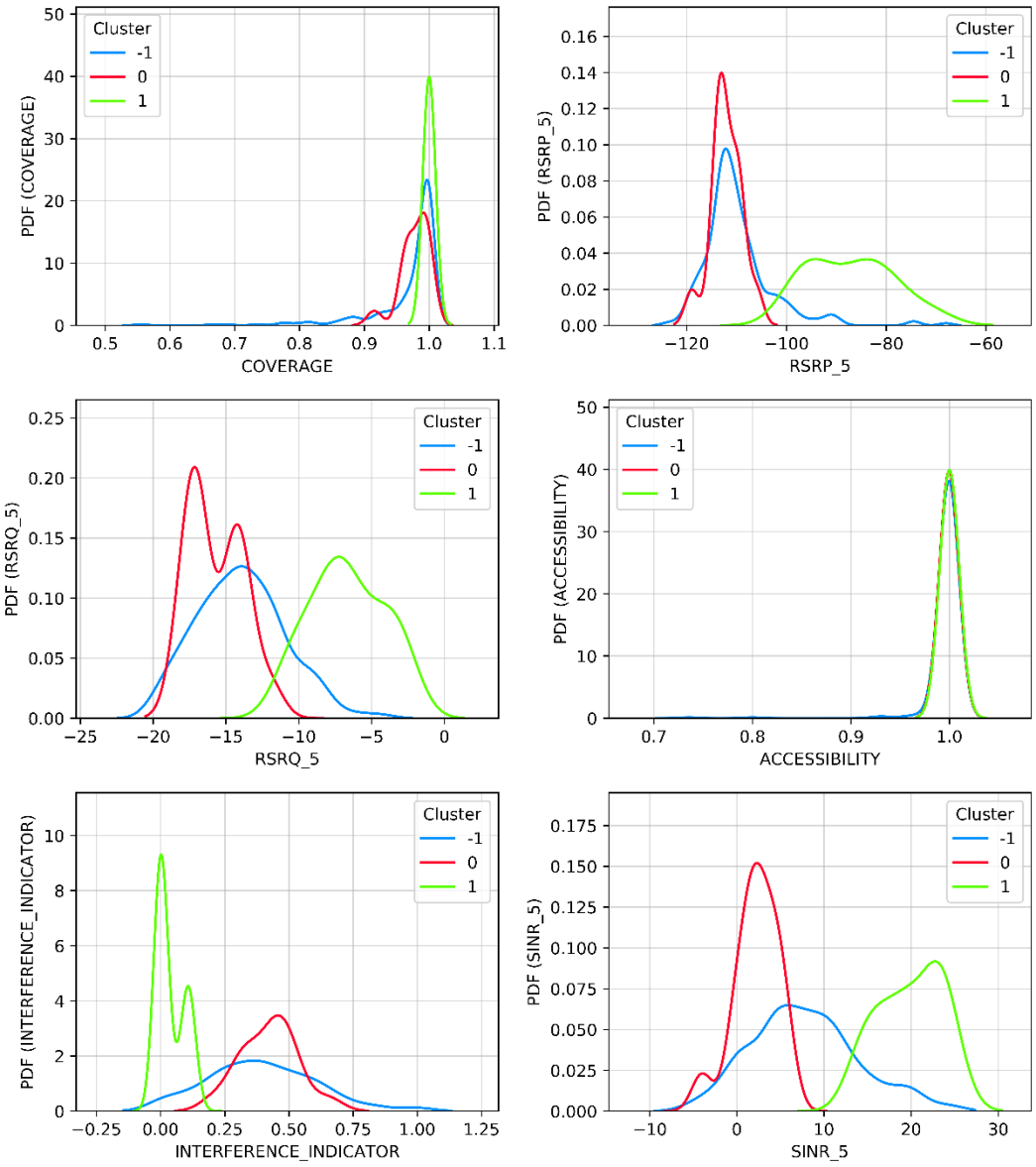


Figure 11. *Sample PDF plots for each KPI obtained using PCA – OPTICS with manual hyperparameter selection.*

Then, for the same sample, the classification performance is -0.12, which means data points are evaluated as belonging to the wrong cluster, according to the definition of Silhouette [51]. This can be observed in the data points scatter plot (Figure 12) for the same sample as in Figure 11, where the cluster 0 is surrounded with data points belonging to the cluster -1 and the distance between the centroids of clusters 0 and -1 is small.

In addition, choosing the hyperparameters for OPTICS is not trivial due to the high sensitivity of the method (Table 6). Therefore, a compromise must be made between the number of clusters detected and their relevance. In the case of the latter sample, changing the parameters may increase the number of irrelevant clusters, or prevent the detection of interesting clusters, like the cluster 1 which can be identified as *Good / Normal*.

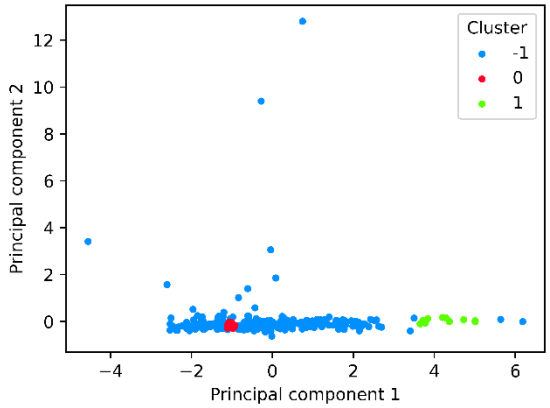


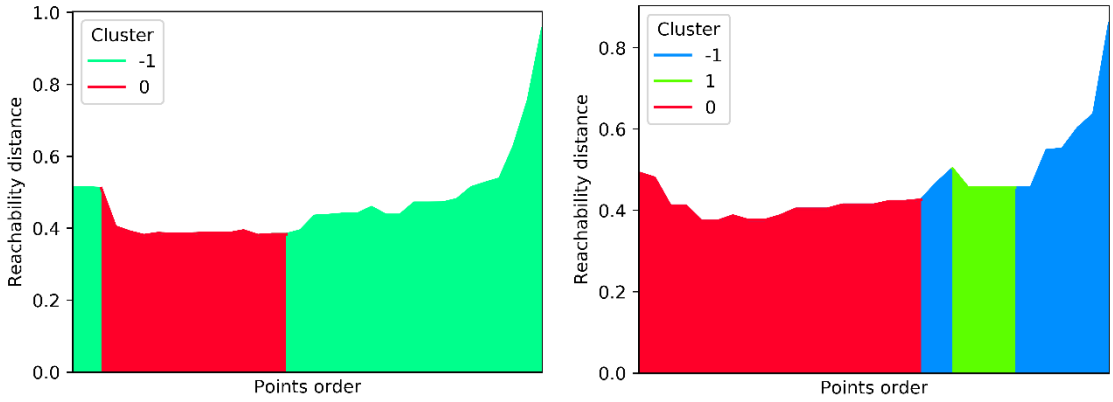
Figure 12. *Sample clusters plot using PCA – OPTICS with manual hyperparameter selection.*

Therefore, the latter observations highlight a limitation of the classification performance metric as it is not necessarily reflecting the capability of the method for recognizing relevant patterns. Indeed, internal evaluation metrics such as the Silhouette and the DB index do not consider pattern relevance as a factor. However, other evaluation methods usually involve the requirement for labeled data, which is considered unavailable in this study. This is one of the reasons why expert annotation is needed for a better evaluation of the method.

Another aspect to consider is the capacity for OPTICS clustering not to detect any cluster for a given dataset in contrast to K-means, for which we force the minimum number of clusters to be 2. While such cases are not considered for the calculation of the classification performance and consistency metrics, their interpretation is ambiguous. A high proportion of failed detection can either mean the clustering method is not adapted, or the dataset does not contain abnormal behaviors. However, comparing these results with those for other clustering methods reveals that different causes can be identified, which means OPTICS-based methods can fail to detect them.

In the case of SOM – OPTICS, it is also possible to distinguish the two *Good / Normal* and *Low power / Performance degradation* clusters in most cases. In contrast to the methods using *None* and PCA, using SOM with Bayesian optimization provided hyperparameters that produce clusters that can be identified distinctively. This can be explained by the vector quantization performed by SOM which reduces the number of data points and therefore the distance between the ones that are close to each other.

However, the main issue with SOM – OPTICS is that it detects very few clusters: 25% of the resulting cluster sets only contain one cluster, while 67% contain 2 clusters. This difficulty can be observed in the reachability plots in Figure 13 in which valleys are very flat, thus making potential clusters difficult to be detected.



(a) (b)

Figure 13. *Sample reachability plots for SOM – OPTICS. (a) shows a detected cluster which forms an almost-flat valley (Silhouette value = -0.05). (b) shows small valleys are detected (Silhouette value = 0.27).*

Further, we observe in Table 6 that OPTICS clustering is the fastest clustering method compared to K-means and GNG-PP (regardless of the dimensionality reduction process). However, if this is the case for a small dataset like the ones used in this study, the computation time will increase exponentially as the complexity for this algorithm is $O(n^2)$ if the distance ε is infinite, which is the case in our experiment. Therefore, for larger datasets, it is recommended to choose a smaller value for ε to reduce the computation time.

To conclude, the results from this study shows OPTICS-based methods offer limited results for finding patters of mobile network faults. While these methods can sometimes detect two main patterns, they have not been able to detect other patterns driven by a smaller number of data points. Instead, a larger number of clusters often result in finding variants of main clusters that are not enough significantly different. The outcome for OPTICS is likely to be similar with other density-based clustering algorithms, as the data

points do not visually appear to form several clusters based on denser areas, in most cases.

5.3 GNG-PP-based methods

GNG-PP-based methods offer one of the best classification performances on average, but also the highest standard deviation (hence the lowest consistency), highest sensitivity, and highest computation time without using dimensionality reduction (Table 6). GNG-PP is also the method that requires the most hyperparameters to select (6), which increases the method setup difficulty.

In comparison to the two other clustering algorithms, clustering with GNG-PP provides slightly more clusters on average (3.4 for *None* and 3.8 for PCA). In addition, clusters are usually easier to identify because their variance is smaller. As an example, Figure 14 shows the PDF plots for a sample clustering set obtained with *None* – GNG-PP. In this case, the four clusters have distinctively different patterns and, in contrast to K-means, overlapping clusters are rare for GNG-PP.

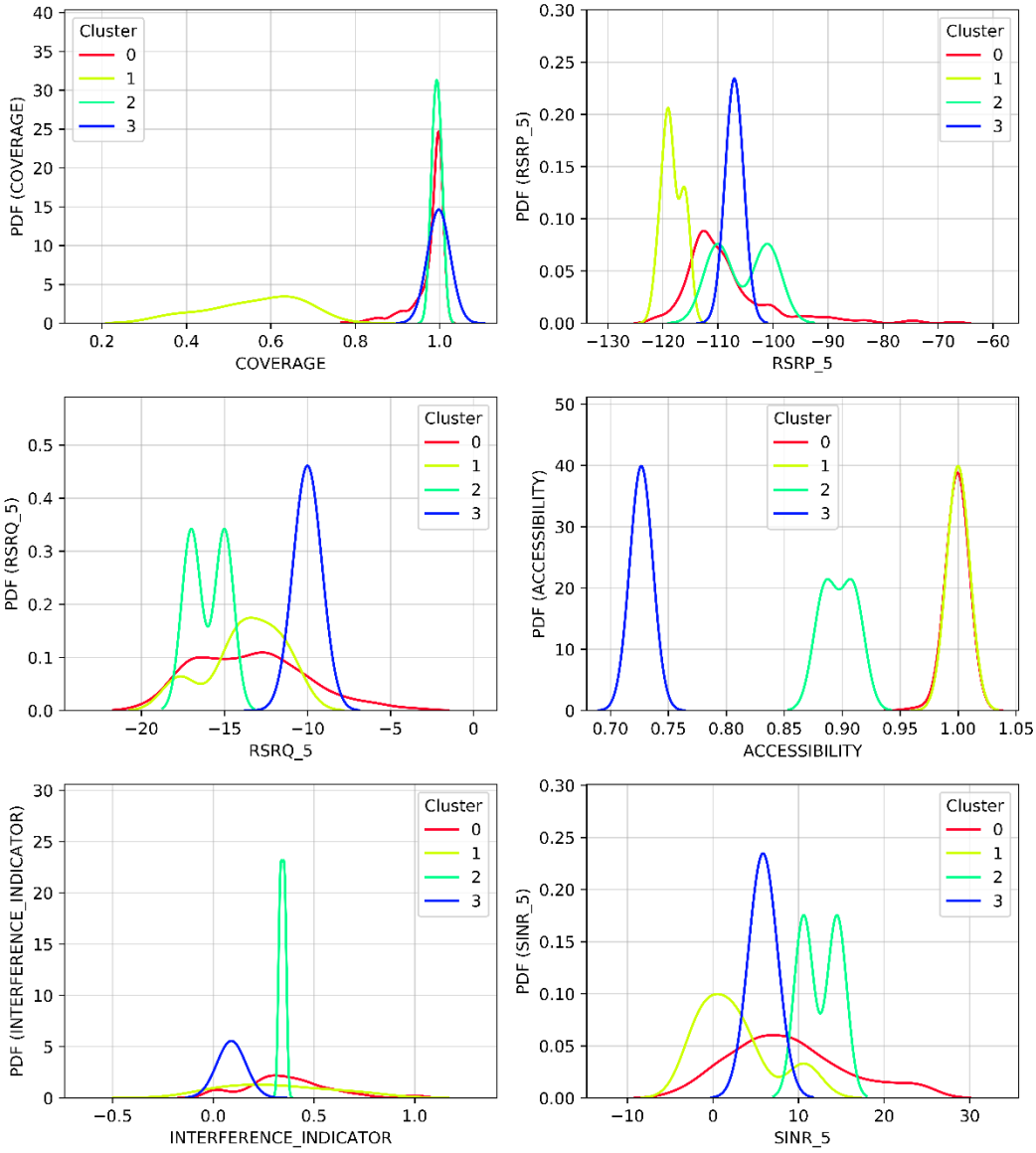


Figure 14. *Sample PDF plots for each KPI obtained using None – GNG-PP.*

We observed that distinctive patterns are also visible in most cluster sets obtained with GNG-PP. Therefore, the proportion of presence of the different identified causes is reported in Table 9. We can observe *None* – GNG-PP can detect *Good / Normal* clusters in only 11% of cases, when the latter cause is widely identified with K-means. On the other hand, PCA – GNG-PP detects the latter cause more often (in 74% of cases) as well as more diverse causes.

Table 9. *Presence of identified clusters in all time-window subsets for GNG-PP-based methods. Unreliable clusters (with less than three data points) are excluded.*

	Low power / Performance degradation	Low coverage	Good / Normal	Inter-site interference	Medium downlink / high uplink
***None* – GNG-PP**	100%	7%	11%	0%	0%
PCA – GNG-PP	100%	19%	74%	7%	4%

If GNG-PP clustering appears to detect clusters more accurately, we can observe it can sometimes provide clusters with a very small number of data points. In this study, we consider that clusters with three or less data points are not reflecting a recurrent pattern and are unreliable. Therefore, only 41% for *None* and 63% for PCA, of clusters are reliable. This problem is mainly causing the lack of diverse clusters identified as shown in Table 9, because many of the unreliable clusters tend to show patterns that could be identified.

Further, this problem can be explained by the training process of GNG-PP, which stops after a given number of epochs and try to approach a vector picked from the input distribution. Therefore, the algorithm may terminate in a state in which nodes do not map to the correct patterns and consequently explain the low consistency. An example of this phenomenon can be observed in Figure 15, where the cluster 2 only maps to three data points.

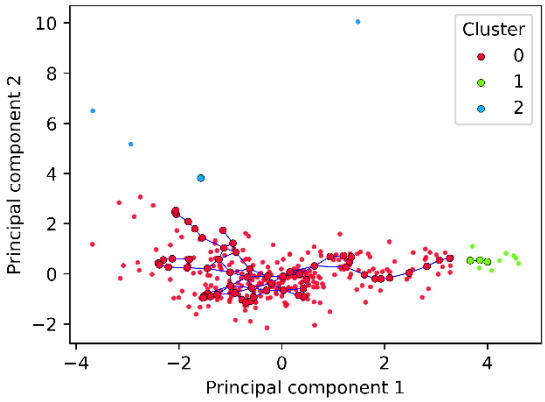


Figure 15. *Sample clusters plot using PCA – GNG-PP. The graph is generated by GNG-PP and the smaller points represent data points.*

The latter issue can be mitigated by adjusting the different hyperparameters for GNG-PP. For example, reducing the winner and neighbor learning rates would reduce the probability to have nodes that are far away from the denser part of the graph, because node updates would be less significant. However, the algorithm will detect less clusters

and may be less efficient. Finding the right compromise to obtain the most accurate results without having irrelevant clusters is not trivial, which can explain the high sensitivity of the method.

Another possible solution to improve the results would be to have a larger dataset, in particular with a greater number of network cells. This is expected to increase the number of faulty cells and therefore increase the reliability of rarer patterns.

Then, it is also possible for GNG-PP-based methods not to detect any clusters, like for OPTICS-based methods. Again, this phenomenon can be expected if all the cells are behaving normally, but it cannot be verified with unlabeled data.

Regarding the long computation time, we observed it is mainly dependent on the number of epochs, the maximum number of nodes and the input vector size. Therefore, one can adjust these parameters depending on the importance of computation time for their usage.

To conclude, GNG-PP-based methods appear to perform similarly to K-means, but trades performance for a lower consistency, higher sensitivity and longer computation time. In contrast to K-means, GNG-PP provides more accurate clusters (with less variance) but also more unreliable ones. However, with datasets containing a larger number of cells, GNG-PP may perform significantly better than K-means.

6. CONCLUSION AND RECOMMENDATIONS

As the mobile network industry grows, mobile network systems become more complex and costly to maintain. As an effort for automating and reducing troubleshooting costs, Self-Organized Networks, and in particular self-healing concepts, were introduced into 4G LTE technology. While a variety of studies were conducted in order to improve mobile network troubleshooting, no review was found on clustering algorithms, supposedly well suited for this task which generally involves unlabeled datasets. Therefore, the present thesis proposes a comparison of several clustering methods in order to offer a selected set of relevant data to experts.

The methodology first considers pre-processing real mobile network data that could be used by experts for troubleshooting. Then, several combinations of dimensionality reduction algorithms and clustering models are applied and evaluated with different metrics. Finally, the clusters obtained from the most performant methods are annotated by experts to evaluate their relevance.

Therefore, the results show that methods based on OPTICS, a density-based model, are not well suited for the task as they rarely detect more than two relevant mobile network causes.

Then, a popular centroid-based model – K-means – can detect various patterns with a high proportion of reliable clusters – those which contain more than 1% of the input dataset –. The algorithm appears to be especially efficient when preceded by the dimensionality reduction algorithm PCA. However, the clusters obtained with this method often overlap each other and have a large variety of data points making the cluster identification more difficult.

The last clustering algorithm addressed in this study was GNG-PP, a neural model. Like K-means, it is able to detect main patterns as well as more specific ones. On the one hand, GNG-PP is slower, more unstable, and more complex than K-means. It also provides more unreliable clusters that have insufficient data points to be significant. On the other hand, this method can provide more accurate clusters in which data points follow the same pattern and have a smaller variance, thus making the annotation easier.

In a nutshell, K-means-based methods would be especially recommended for systems which input dataset contains data for a small number of cells or requires fast processing. However, as larger systems are common in the industry, GNG-PP may perform better in such cases because more input data points are expected to improve the reliability of

clusters. Therefore, the clusters are more accurate than with K-means and contain network cells that are more relevant, consequently saving more time for experts.

Future research paths include a deeper comparison based on datasets of different sizes, especially larger ones, as well as labeled synthetic data. This would enable the possibility to compare clustering algorithms with the results of other studies using generated datasets.

Then, a system that automatically identifies root causes owing to a limited number of labeled clusters could be investigated. Such a system was developed in [26] with SOM – K-means, which can possibly be applied to our recommendations.

REFERENCES

[1] 3rd Generation Partnership Project (3GPP), “Telecommunication management; Self-Organizing Networks (SON); Concepts and requirements,” 2018.

[2] 3rd Generation Partnership Project (3GPP), “Telecommunication management; Self-Organizing Networks (SON); Self-healing concepts and requirements,” 2018.

[3] M. Ahmed, A. Naser Mahmood, and J. Hu, “A survey of network anomaly detection techniques,” *J. Netw. Comput. Appl.*, vol. 60, pp. 19–31, 2016, doi: https://doi.org/10.1016/j.jnca.2015.11.016.

[4] B. Andersen and T. N. Fagerhaug, *Root Cause Analysis : Simplified Tools and Techniques*. Milwaukee, UNITED STATES: ASQ Quality Press, 2006.

[5] A. Andreakis, N. v. Hoyningen-Huene, and M. Beetz, “Incremental Unsupervised Time Series Analysis Using Merge Growing Neural Gas,” 2009, pp. 10–18.

[6] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, “OPTICS: Ordering Points to Identify the Clustering Structure,” *SIGMOD Rec.*, vol. 28, no. 2, pp. 49–60, 1999, doi: 10.1145/304181.304187.

[7] R. Bellman, “Adaptive control processes a guided tour.” Princeton University Press, Princeton, N.J., 1961.

[8] J. Bergstra and Y. Bengio, “Random Search for Hyper-Parameter Optimization,” *J. Mach. Learn. Res.*, vol. 13, no. 1, pp. 281–305, 2012.

[9] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, “Algorithms for Hyper-Parameter Optimization,” in *Advances in Neural Information Processing Systems 24*, J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2011, pp. 2546–2554.

[10] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, “When is ‘nearest neighbor’ meaningful?,” *Lect. Notes Comput. Sci.*, vol. 1540, pp. 217–235, 1998, doi: 10.1007/3-540-49257-7_15.

[11] A. Bradai, T. Rasheed, T. Ahmed, and K. Singh, “Cellular Software Defined Network – a Framework,” *IEEE Commun. Mag.*, vol. 53, Jun. 2015, doi: 10.1109/MCOM.2015.7120043.

[12] L. Buitinck *et al.*, “API design for machine learning software: experiences from the scikit-learn project,” in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.

[13] F. Canales and M. Chacón, “Modification of the Growing Neural Gas Algorithm for Cluster Analysis,” in *Progress in Pattern Recognition, Image Analysis and Applications*, 2007, pp. 684–693.

[14] W. L. Chang, L. M. Pang, and K. M. Tay, “Application of self-organizing map to failure modes and effects analysis methodology,” *Neurocomputing*, vol. 249, pp. 314–320, Aug. 2017, doi: 10.1016/J.NEUCOM.2016.04.073.

[15] B. Charny, “World’s first 3G phone network goes live,” *ZDNet*, 2001.

[16] J. A. F. Costa and R. S. Oliveira, “Cluster analysis using growing neural gas and graph partitioning,” *IEEE Int. Conf. Neural Networks - Conf. Proc.*, no. October 2018, pp. 3051–3056, 2007, doi: 10.1109/IJCNN.2007.4371447.

[17] D. Davies and D. Bouldin, “A Cluster Separation Measure,” *Pattern Anal. Mach. Intell. IEEE Trans.*, vol. PAMI-1, pp. 224–227, 1979, doi: 10.1109/TPAMI.1979.4766909.

[18] H. E. Driver and A. L. Kroeber, *Quantitative expression of cultural relationships, by H.E. Driver and A.L. Kroeber.* Berkeley: University of California Press, 1932.

[19] Ericsson, “World’s first 4G/LTE network goes live today in Stockholm,” 2009.

[20] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A Density-Based Algorithm for Discovering Clusters 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*, 1996, pp. 226–231.

[21] V. Estivill-Castro, “Why so Many Clustering Algorithms: A Position Paper,” *SIGKDD Explor. Newsl.*, vol. 4, no. 1, pp. 65–75, 2002, doi: 10.1145/568574.568575.

[22] A. Flexer, “On the use of self-organizing maps for clustering and visualization,” *Lect. Notes Comput. Sci.*, vol. 1704, pp. 80–88, 1999, doi: 10.3233/ida-2001-5502.

[23] L. Flores-Martos, A. Gomez-Andrades, R. Barco, and I. Serrano, “Unsupervised system for diagnosis in LTE networks using Bayesian networks,” *IEEE Veh. Technol. Conf.*, vol. 2015, 2015, doi: 10.1109/VTCSpring.2015.7146146.

[24] B. Fritzke, “A growing neural gas network learns topologies,” *Adv. Neural Inf. Process. Syst. 7*, pp. 625–632, 1995.

[25] B. Gajic, S. Novaczki, and S. Mwanje, “An improved anomaly detection in mobile networks by using incremental time-aware clustering,” in *Proceedings of the 2015 IFIP/IEEE International Symposium on Integrated Network Management, IM 2015*, 2015, pp. 1286–1291, doi: 10.1109/INM.2015.7140483.

[26] A. Gomez-Andrades, P. Munoz, I. Serrano, and R. Barco, “Automatic root cause analysis for LTE networks based on unsupervised techniques,” *IEEE Trans. Veh. Technol.*, vol. 65, no. 4, pp. 2369–2386, 2016, doi: 10.1109/TVT.2015.2431742.

[27] GSMA Intelligence, P. Jarich, and T. Hatt, “Global Mobile Trends 2020,” 2019.

[28] H. Hotelling, “Analysis of a complex of statistical variables into principal components.,” *J. Educ. Psychol.*, vol. 24, no. 6, pp. 417–441, 1933, doi: 10.1037/h0071325.

[29] J. D. Hunter, “Matplotlib: A 2D graphics environment,” *Comput. Sci. Eng.*, vol. 9, no. 3, pp. 90–95, 2007, doi: 10.1109/MCSE.2007.55.

[30] E. J. Khatib, R. Barco, I. Serrano, and P. Muñoz, “LTE performance data reduction for knowledge acquisition,” in *2014 IEEE Globecom Workshops (GC Wkshps)*, 2014, pp. 270–274, doi: 10.1109/GLOCOMW.2014.7063443.

[31] T. Kohonen, “Essentials of the self-organizing map,” *Neural Networks*, vol. 37, pp. 52–65, 2013, doi: 10.1016/j.neunet.2012.09.018.

[32] T. Kohonen, *MATLAB Implementations and Applications of the Self-Organizing Map*. 2014.

[33] T. Kohonen, “Self-organized formation of topologically correct feature maps,” *Biol. Cybern.*, vol. 43, no. 1, pp. 59–69, 1982, doi: 10.1007/BF00337288.

[34] E. Kreyszig, H. Kreyszig, and E. J. Norminton, *Advanced Engineering Mathematics*, Tenth. Hoboken, NJ: Wiley, 2011.

[35] K. Kurniabudi *et al.*, “Network anomaly detection research: A survey,” *Indones. J. Electr. Eng. Informatics*, vol. 7, pp. 36–49, Mar. 2019, doi: 10.11591/ijeei.v7i1.773.

[36] N. Lavesson and P. Davidsson, “Quantifying the Impact of Learning Algorithm Parameter Tuning,” in *AAAI*, 2006, pp. 395–400.

[37] S. P. Lloyd, “Least Squares Quantization in PCM,” *IEEE Trans. Inf. Theory*, vol. 28, no. 2, pp. 129–137, 1982, doi: 10.1109/TIT.1982.1056489.

[38] S. López, “Anomaly Detection and Root Cause Analysis for LTE Radio Base Stations,” School of Electrical Engineering and Computer Science (EECS), KTH, Stockholm, 2018.

[39] J. MacQueen, “Some methods for classification and analysis of multivariate observations,” in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, 1967, pp. 281–297.

[40] G. Marsaglia, W. W. Tsang, and J. Wang, “Evaluating Kolmogorov’s distribution,” *J. Stat. Softw.*, vol. 8, pp. 1–4, 2003, doi: 10.18637/jss.v008.i18.

[41] F. J. Massey, “The Kolmogorov-Smirnov Test for Goodness of Fit,” *J. Am. Stat. Assoc.*, vol. 46, no. 253, pp. 68–78, Mar. 1951, doi: 10.1080/01621459.1951.10500769.

[42] W. McKinney, “Data Structures for Statistical Computing in Python,” in *Proceedings of the 9th Python in Science Conference*, 2010, pp. 51–56.

[43] Microsoft Corporation, “Most cited data mining articles according to Microsoft academic search,” 2010. [Online]. Available: https://web.archive.org/web/20100421170848/http://academic.research.microsoft.com/CSDirectory/paper_category_7.htm.

[44] V. Moosavi, S. Packmann, and I. Vallés, “SOMPY: A Python Library for Self Organizing Map (SOM).” 2014.

[45] F. Nogueira, “Bayesian Optimization: Open source constrained global optimization tool for Python.” .

[46] T. Oliphant, “A guide to NumPy.” USA: Trelgol Publishing, 2006.

[47] S. Papadopoulos, A. Drosou, N. Dimitriou, O. H. Abdelrahman, G. Gorbil, and D. Tzovaras, “A BRPCA based approach for anomaly detection in mobile networks,” *Lecture Notes in Electrical Engineering*, vol. 363. Imperial College London, London, United Kingdom, pp. 115–125, 2015, doi: 10.1007/978-3-319-22635-4_10.

[48] A. Paradisi, M. D. Yacoub, F. L. Figueiredo, and T. Tronco, *Long Term Evolution: 4G and Beyond*. Springer International Publishing, 2015.

[49] M. S. Parwez, D. B. Rawat, and M. Garuba, “Big Data Analytics for User-Activity Analysis and User-Anomaly Detection in Mobile Wireless Network,” *IEEE Trans. Ind. Informatics*, vol. 13, no. 4, pp. 2058–2065, 2017, doi: 10.1109/TII.2017.2650206.

[50] K. Pearson, “On lines and planes of closest fit to systems of points in space,” *London, Edinburgh, Dublin Philos. Mag. J. Sci.*, vol. 2, no. 11, pp. 559–572, 1901, doi: 10.1080/14786440109462720.

[51] P. Rousseeuw, “Rousseeuw, P.J.: Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. Comput. Appl. Math. 20, 53-65,” *J. Comput. Appl. Math.*, vol. 20, pp. 53–65, Nov. 1987, doi: 10.1016/0377-0427(87)90125-7.

[52] Y. Shevchuk, “NeuPy,” 2015. [Online]. Available: http://neupy.com.

[53] B. W. Silverman, *Density estimation for statistics and data analysis*. London; New York: Chapman and Hall, 1986.

[54] L. Sun-hee and L. Eun-joo, “S. Korea goes 5G wireless from Saturday,” *Maeil Business News Korea*, 2018.

[55] I. Syarif, A. Prugel-Bennett, and G. Wills, “Unsupervised Clustering Approach for Network Anomaly Detection,” 2012, pp. 135–145.

[56] M. T. Taner, J. D. Walls, M. Smith, G. Taylor, M. B. Carr, and D. Dumas, “Reservoir characterization by calibration of self-organized map clusters,” in *SEG Technical Program Expanded Abstracts 2001*, 2005, pp. 1552–1555.

[57] P. Virtanen *et al.*, “SciPy 1.0-Fundamental Algorithms for Scientific Computing in Python,” *CoRR*, vol. abs/1907.1, 2019.

[58] I. Vural and H. Venter, “Mobile Botnet detection using network forensics,” *Lecture Notes in Computer Science*, vol. 6369 LNCS. Department of Computer Science, University of Pretoria, Pretoria 0002, South Africa, pp. 57–67, 2010, doi: 10.1007/978-3-642-15877-3_7.

[59] X. Wang, Y. Jin, and Y. Yu, *A Mobile Network Performance Evaluation Method Based on Multivariate Time Series Clustering with Auto-Encoder*. 2018.

[60] P. F. Wilson, L. D. Dell, and G. F. Anderson, *Root Cause Analysis: A Tool for Total Quality Management*. ASQC Quality Press, 1993.

[61] D. H. Wolpert, “The Lack of A Priori Distinctions Between Learning Algorithms,” *Neural Comput.*, vol. 8, no. 7, pp. 1341–1390, 1996, doi: 10.1162/neco.1996.8.7.1341.

[62] D. H. Wolpert and W. G. Macready, “No Free Lunch Theorems for Optimization,” *Trans. Evol. Comp*, vol. 1, no. 1, pp. 67–82, 1997, doi: 10.1109/4235.585893.

[63] J. Zheng and V. Vaishnavi, “A Multidimensional Perceptual Map Approach to Project Prioritization and Selection,” *AIS Trans. Human-Computer Interact.*, vol. 3, pp. 82–103, 2011, doi: 10.17705/1thci.00028.