Unsupervised Learning: Anomaly Detection in Hypothyroidism dataset

University of Milan-Bicocca

Andrea Palmieri
a.palmieri13@campus.unimib.it
Matricula 921785

Andrea Yachaya
a.yachaya@campus.unimib.it
Matricula 913721

Abstract

This report aims to provide a comprehensive overview of Unsupervised Learning methodologies used for analyzing the Hypothyroidism dataset to detect anomalous observations. Hypothyroidism, characterized by an underactive thyroid gland, poses significant challenges in diagnosis and treatment. The dataset used in this study lacks labeled data, making anomaly detection with an unknown number of clusters the primary focus.
Unsupervised Learning plays a crucial role in scenarios where labeled data is scarce or unavailable. By leveraging algorithms such as hierarchical clustering, DBSCAN, k-medoids, and k-prototypes on the Hypothyroidism dataset, this study explores their effectiveness in uncovering anomalies.

1 Introduction

Anomaly detection is a crucial aspect of data analysis, especially in healthcare, where identifying rare or unusual cases can significantly impact patient outcomes. This report focuses on anomaly detection using Unsupervised Learning methods within the Hypothyroidism dataset, which consists of 7,200 observations with 15 binary columns and 6 continuous columns. The primary goal of this study was to identify abnormal patterns or anomalies that may indicate potential health issues or data quality problems and compute a probability score of an observation being anomaly.

This report does not contain any form of plagiarism, including content generated or suggested by AI tools such as ChatGPT or similar services. All sources used have been properly cited and referenced.

2 Dataset Preprocessing

To prepare the dataset for analysis, several preprocessing steps were undertaken. First, the continuous features were standardized to prevent any single feature from disproportionately influencing the results, using the following equation:

$$Z = \frac{x - \mu}{\sigma} \tag{1}$$

where Z is the standardized value, x is the original value, μ is the mean of the feature, and σ is the standard deviation of the feature.
Second, the binary features were converted to object data types. Given that these features are binary, one-hot encoding was not necessary, as each feature already represents a distinct categorical variable.
Finally, the original classes of each sample in the dataset are not provided. Consequently, our project focuses on unsupervised learning techniques to identify anomalies without relying on class labels.

3 Dimensionality Reduction

3.1 Anomaly Detection using PCA

We have excluded Principal Component Analysis (PCA) for anomaly detection and dimensionality reduction due to its inherent nature. PCA transforms the original dataset into a series of linearly uncorrelated components, ordered by the variance they explain. This process involves computations of the covariance matrix and the determination of eigenvalues and eigenvectors. These mathematical operations require numerical data, as concepts like covariance, eigenvalues, and eigenvectors do not directly apply to categorical data. Given that our dataset comprises

mixed data types, including categorical variables, PCA is not suitable for our purposes as it cannot effectively handle non-continuous features.

3.2 FAMD

To overcome the limitations of PCA, we employed Factor Analysis of Mixed Data (FAMD) as a dimensionality reduction technique. FAMD is a variant of PCA that can handle mixed data types, including categorical and continuous variables, making it particularly useful for our dataset. However, in this study, we did not use FAMD for anomaly detection because the original features cannot be straightforwardly reconstructed from the FAMD coordinates.
Instead, we used FAMD to reduce the dimensionality of the data, projecting the original feature space onto a smaller one. This enables us to visualize the data in a lower-dimensional space and to plot the clustering results of the algorithms discussed in the following sections.

4 Distance Based Clustering

The first category of approaches implemented were about distance-based clustering algorithms. These methods group similar data points into clusters based on some measure of distance or similarity. These algorithms rely on the concept of distance (or dissimilarity) between data points to form clusters where points within the same cluster are more similar (closer) to each other than to those in other clusters. Since we are dealing with a dataset containing mixed data-type, euclidean distance operates under the assumption of continuous, numerical data. It calculates the straight-line distance between points in a multi-dimensional space, which works well when dealing with numeric attributes that represent continuous quantities, such as temperature or length. However, in high-dimensional spaces, Euclidean distance can become less meaningful: this is often referred to as the "curse of dimensionality". As the number of dimensions increases, the distance between points becomes more similar, and it becomes difficult to differentiate between points. This phenomenon can lead to poor performance in clustering and classification tasks.
To address this problems, we employed the use of a different distance metric: the **Gower distance**.

4.1 Gower distance

The Gower distance is a metric used to calculate the dissimilarity or distance between two points in a dataset, particularly useful in situations where traditional distance metrics like Euclidean distance may not be appropriate due to the nature of the data. The

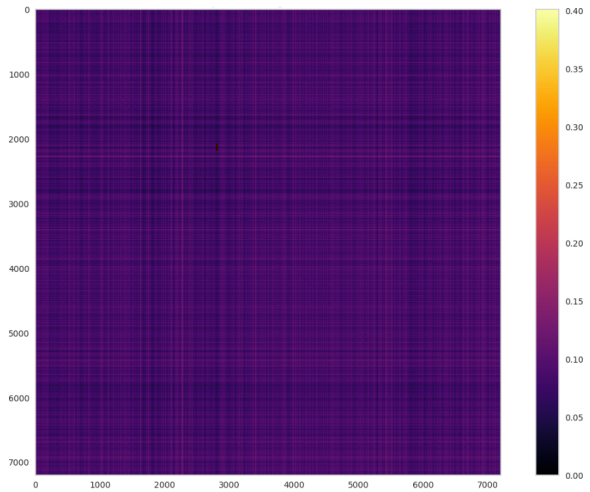

Figure 1: Proximity distance matrix using Gower distance.

Gower distance $d_{i,j}$ between two data points i and j in a dataset with p variables can be calculated using the following formula:

$$d_{i,j} = \frac{\sum_{k=1}^{p} w_k \cdot \delta_k(x_{ik}, x_{jk})}{\sum_{k=1}^{p} w_k} \tag{2}$$

where:

- w_k is the weight for variable k
- $\delta_k(x_{ik}, x_{jk})$ is the dissimilarity measure for variable k between data points i and j

For numerical variables, the dissimilarity $\delta_k(x_{ik}, x_{jk})$ is calculated using the absolute difference between the normalized values of the feature:

$$\delta_k(x_{ik}, x_{jk}) = \frac{|x_{ik} - x_{jk}|}{\max(x_k) - \min(x_k)} \tag{3}$$

where:

- x_{ik} and x_{jk} are the values of the k-th feature for observations i and j,
- $\max(x_k)$ and $\min(x_k)$ are the maximum and minimum values of the k-th feature across all observations.

For categorical variables, the dissimilarity $\delta_k(x_{ik}, x_{jk})$ is calculated using a simple matching approach:

$$\delta_k(x_{ik}, x_{jk}) = \begin{cases} 0 & \text{if } x_{ik} = x_{jk} \\ 1 & \text{if } x_{ik} \neq x_{jk} \end{cases} \tag{4}$$

where x_{ik} and x_{jk} are the values of the k-th feature for observations i and j.
In our specific case we imposed all the weights equals

to each other since we didn't have any domain-specific knowledge about the dataset to sensibly impose a different weight to one or more variables. The proximity matrix created using Gower distance, which will be used by the clustering methods we developed during the project, is shown in figure [1].

4.2 Hierarchical Clustering

Hierarchical clustering is a powerful technique that produces nested clusters, which can be visualized as a dendrogram. This method is particularly useful for understanding the hierarchical relationships between clusters. In agglomerative hierarchical clustering, there are several methods to compute inter-cluster proximity, each with its own advantages and disadvantages. These methods include minimum (single linkage), maximum (complete linkage), average, centroids-based methods, and Ward's method, which minimizes the sum of squared errors within clusters.
Ward's method tends to create globular clusters, making it a good candidate for initializing K-means clustering. However, Ward's method and centroid-based methods require the use of Euclidean distances and are not suitable for non-Euclidean distance metrics like the Gower distance.
For this reason hierarchical clustering was performed with the following linkage methods: *complete* and *average*. These methods do not require Euclidean distances and can work directly with the Gower distance matrix.

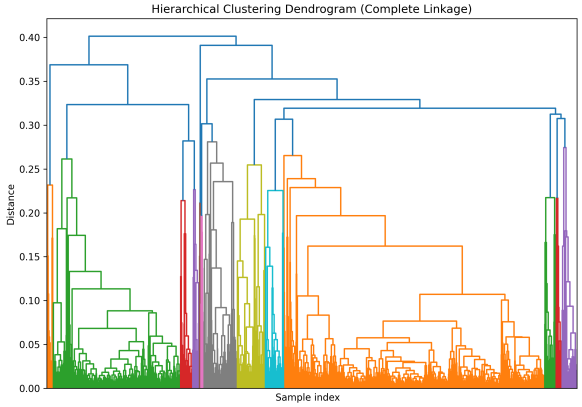

Figure 2: Dendrogram of *complete* linkage method.

4.2.1 Comparison of Linkage Methods and Analysis

The complete linkage method tries to minimize the maximum distance between points within each cluster, leading to the formation of compact clusters with small diameters. However, this method is susceptible to noise and anomalies, as a single anomaly can significantly increase the distance between clusters. The dendrogram in Figure [2] shows a highly branched structure with many small clusters, indicating that the complete linkage method has formed numerous small, tight clusters. On the other hand, the average linkage method defines the distance between two clusters as the average distance between all pairs of points, where one point is from each cluster. This method tends to produce clusters with roughly equal variance and is less susceptible to noise and anomalies compared to the complete linkage method. The dendrogram in Figure [3] shows a more balanced structure with fewer branches and larger clusters. The average linkage method has formed fewer but larger clusters, with a more gradual increase in the distance between clusters as the clusters merge.

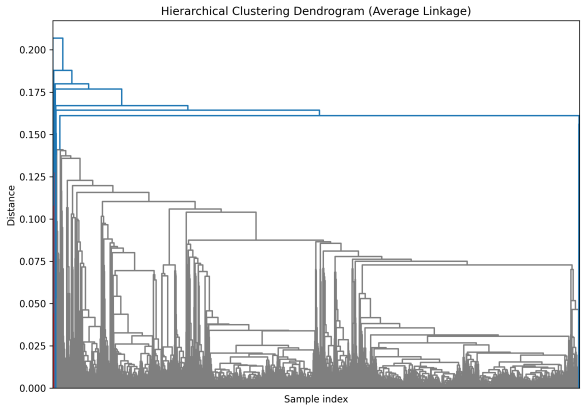

Figure 3: Dendrogram of *average* linkage method.

4.2.2 Determining the Optimal Threshold Distance

To determine the optimal threshold distance, we selected the one with the highest Silhouette score. The Silhouette score measures how similar a data point is to its own cluster compared to other clusters. It is defined for each point i as:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))} \tag{5}$$

where:

- $a(i)$ is the average distance between i and all other points in the same cluster.
- $b(i)$ is the minimum average distance between i and points in a different cluster.

The Silhouette score ranges from -1 to 1, where a higher score indicates better clustering performance, with points well matched to their own cluster and poorly matched to neighboring clusters.
The Silhouette score was computed for 100 times from 0.001 to the maximum height of the dendrogram, the

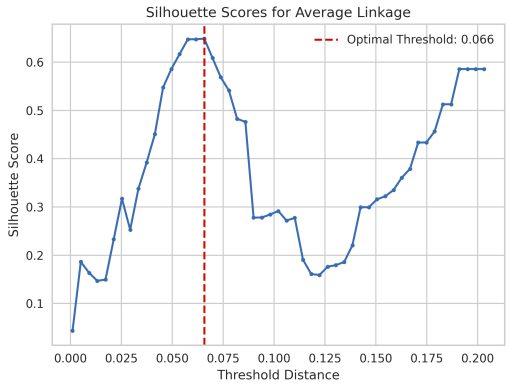

Figure 4: Silhouette score with *average* as linkage method at varying distance threshold. Optimal distance 0.065 with Silhouette 0.649.

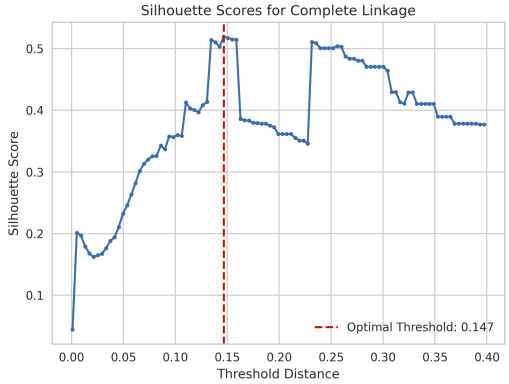

Figure 5: Silhouette score with *complete* as linkage method at varying distance threshold. Optimal distance 0.147 with Silhouette 0.519.

distance where all points have been hierarchically clustered. The maximum height varies for different linkage methods due to the different ways they compute distances between clusters: complete linkage had a maximum height of 0.401 and average linkage had a maximum height of 0.207.
The optimal threshold distance for average linkage corresponded to a distance of 0.065 with a Silhouette score of 0.649, while for complete linkage the optimal threshold distance was 0.147 with a Silhouette score of 0.519.

4.2.3 Identifying Anomalies and Computing Probabilities

To identify anomalies, clusters were ranked in descending order of size, and a threshold k was selected to exclude a certain number of the larger clusters from being counted as anomalies. Consequently, all elements within the remaining $n - k$ smaller clusters are considered anomalies. This process produced the curve of

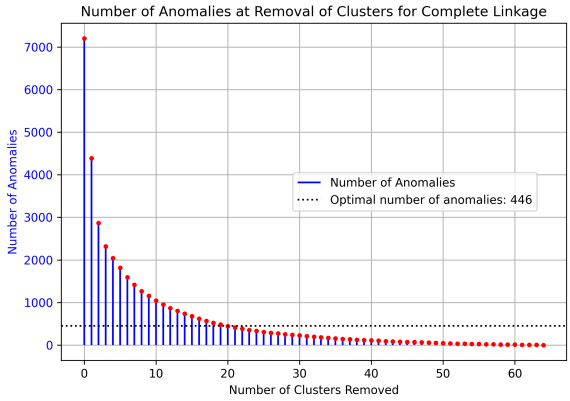

Figure 6: Number of anomalies with *complete* as linkage method at removal of clusters in decreasing order.

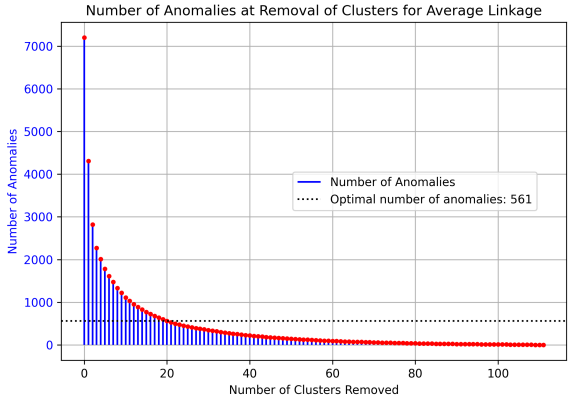

Figure 7: Number of anomalies with *average* as linkage method at removal of clusters in decreasing order.

Figure [6]. The optimal number of clusters to remove was determined visually by choosing a threshold after which change to the number of anomalies became insignificant. We opted to remove the 20 largest clusters, leading to 446 anomalies.
Next, we calculated the probability of each data point being an anomaly using an approach based on the 20th cluster as threshold. The increment in probability is computed as follows:

$$p^{(i)}_{\text{anomaly}} = \begin{cases} 0 & \text{if i=0} \\ \frac{0.5}{n_{\text{before threshold}}} + p^{(i-1)}_{\text{anomaly}} & \text{if i} < \text{threshold} \\ 0.5 & \text{if i} = \text{threshold} \\ \frac{0.5}{n_{\text{after threshold}}} + p^{(i-1)}_{\text{anomaly}} & \text{if i} > \text{threshold} \end{cases}$$

Each cluster was assigned a probability based on this approach, and each data point inherited the probability of its corresponding cluster. This method enabled a smooth transition in the anomaly probabilities.

The points identified as anomalies by the Linkage Methods are shown in Figure [8] and Figure [9], plotted

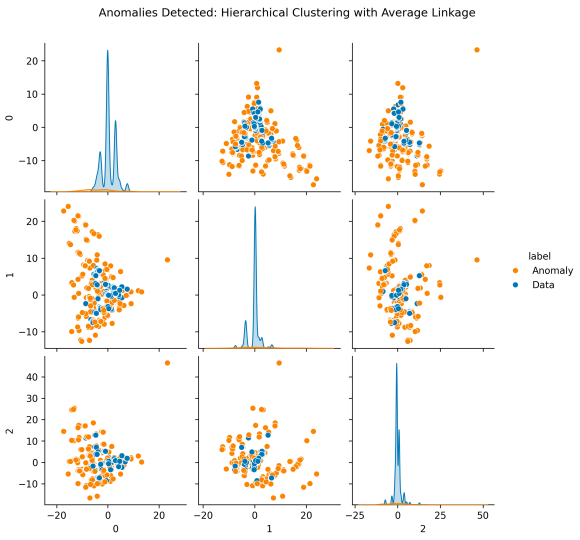

Figure 8: Hierarchical Clustering with Average Linkage: 428 anomalies visualized using FAMD.

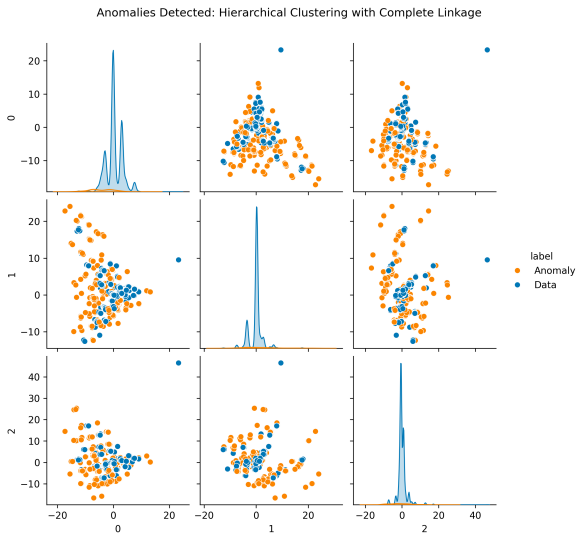

Figure 9: Hierarchical Clustering with Complete Linkage: 446 anomalies visualized using FAMD.

using FAMD for reducing the dimensionality from 21 features to 3 coordinates.

4.3 K-Means

K-Means is a clustering algorithm that partitions data into K clusters by minimizing variance within each cluster. It uses the Euclidean distance to compute cluster centroids as the mean of assigned points. The Euclidean distance metric does not handle categorical data effectively, leading to inaccurate cluster assignments. Additionally, calculating the mean of categorical variables is meaningless, causing unreliable centroids. Therefore, K-Means is not appropriate for our dataset of mixed data types.
Since we're using the Gower distance as a metric, we will use K-Medoids and K-Prototypes as clustering algorithms which can handle mixed data types dataset and precomputed metrics: the Gower distance in this study.

4.4 K-Medoids

K-Medoids is a variant of the popular K-Means clustering algorithm that is particularly well-suited for datasets containing mixed data types. K-Medoids uses actual data points within the cluster as representatives, known as medoids. The medoid is a data point that minimizes the sum of distances to all other points within the cluster.
The advantage of K-Medoids over K-Means for datasets with mixed data types is that it can use a precomputed metric - the Gower distance in our case - to find the medoids, making it suitable for our dataset.

4.4.1 Determining the optimal number of clusters

To determine the optimal number of clusters for the dataset, we employed the elbow method as shown in Figure [10]. This method identifies the point at which adding more clusters does not significantly improve the clustering performance, balancing model complexity and performance. The optimal number of clusters was 5 and we used the indices of the medoids to identify anomalies and compute their probabilities in the following section.

4.4.2 Identifying Anomalies and Computing Probabilities

Anomalies were identified by assessing the distance of each element from its nearest medoid: the closer an observation is to its nearest medoid, the lower the

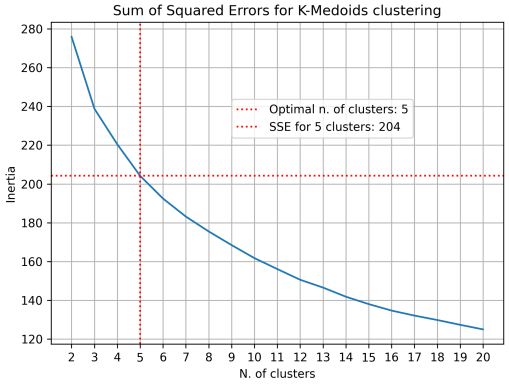

Figure 10: K-Medoids: SSE visualization with elbow.

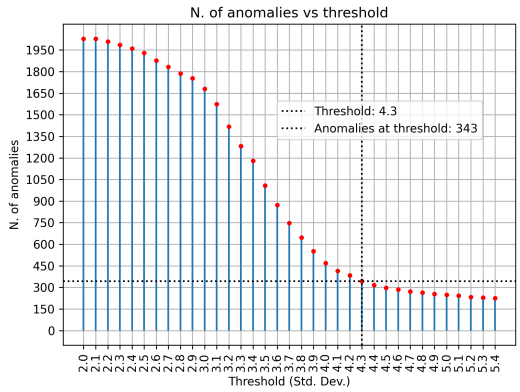

Figure 11: K-Medoids: number of anomalies for each threshold. For *threshold* = 4.3 a total of 343 anomalies was found.

probability that the observation is an anomaly. Elements deviating by more than a specified number of standard deviations from the mean distance within the same cluster were considered anomalies. Figure [11] illustrates the number of anomalies identified for each threshold. We opted to adopt a threshold value of 4.3, correlating to a count of 343 anomalies denoted by a red dotted line. This specific threshold was selected because beyond this value, the reduction in the number of anomalies detected exhibits a less pronounced decline, affecting fewer elements. Since anomalies are considered deviations from the norm, our objective is to identify a point beyond which further increases in the threshold value yield minimal changes in the count of anomalies detected.

The probability was computed as the distance of a point from its medoid divided by the threshold for that cluster and the result rounded either to 0 (data) or 1 (anomaly). Points at a distance equal or greater to the threshold of their cluster had a probability equal to 100%. The points identified as anomalies by K-Medoids are shown in Figure [12], plotted using FAMD for reducing the dimensionality from 21 features to 3 coordinates.

4.5 K-prototypes

Similarly with K-Medoids, we also employed the use of K-Prototype, which can also handle mixed data types but unlike K-Medoids, it uses prototypes, which are synthetic points representing the cluster. For numerical attributes, the prototype is the mean of the cluster. For categorical attributes, the prototype is the mode (most frequent category) of the cluster. Moreover, while in the case of K-Medoids, the medoid is an actual data point from the dataset that is part of the cluster, in K-Prototype the prototype is a synthetic point and does not necessarily correspond to an actual

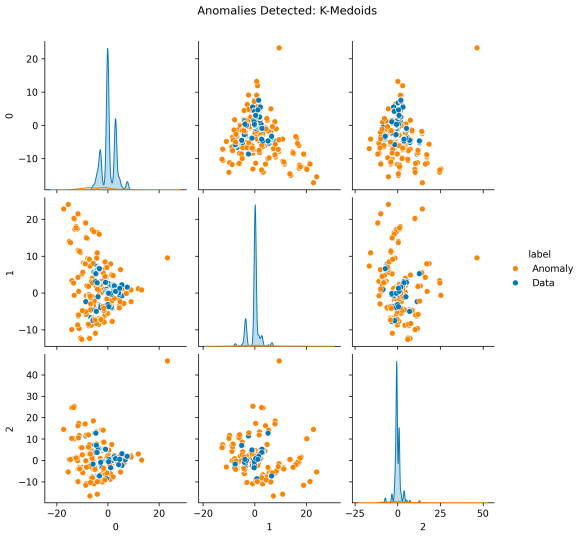

Figure 12: K-medoids: 343 anomalies visualized using FAMD.

data point in the dataset.

4.5.1 Determining the optimal number of clusters

Similar to K-Medoids clustering algorithm, for K-prototypes the optimal number of clusters is determined with the elbow method. The curve is shown in Figure [13] and the optimal number of clusters was found equal to 4.

4.5.2 Identifying Anomalies and Computing Probabilities

Similarly to K-Medoids algorithm, a point is identified as anomaly if its distance to the respective prototype is greater than a threshold. To compute the distance we followed closely the same method of the *KPrototypes* function. The curve is shown in Figure [14]

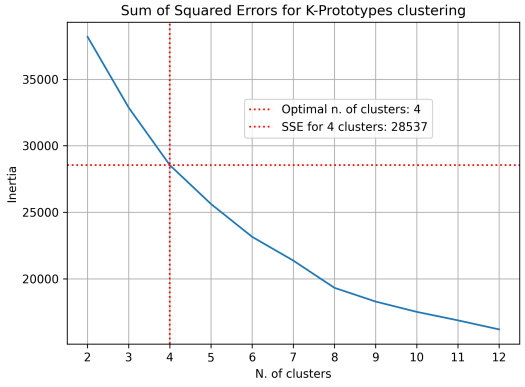

Figure 13: K-Prototypes: SSE visualization with elbow.

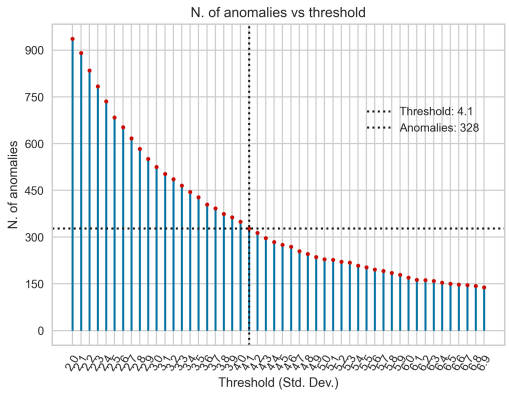

Figure 14: K-Prototypes: number of anomalies for each threshold. For threshold = 4.1 a total of 328 anomalies was found.

and the chosen threshold was 4.1 with 328 anomalies. The points identified as anomalies by K-Prototypes are shown in Figure [15], plotted using FAMD for reducing the dimensionality from 21 features to 3 coordinates.

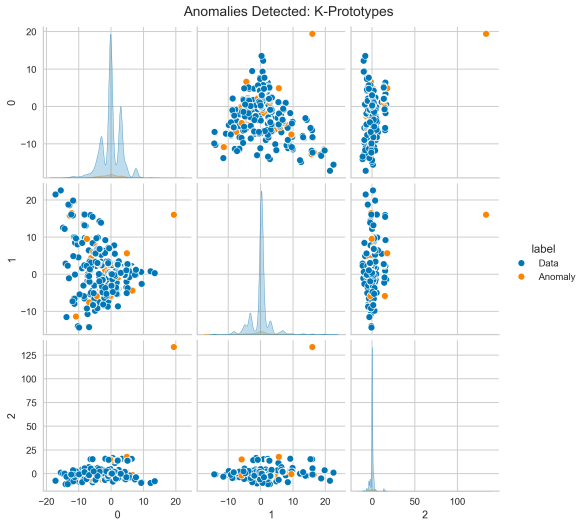

Figure 15: K-Prototypes: 328 anomalies visualized using FAMD

5 Density Based Clustering

Density-based clustering is a powerful technique for identifying clusters of arbitrary shape and separating noise or anomalies from the underlying clusters. Unlike distance-based methods like K-Means or K-Medoids, density-based clustering algorithms do not require a predefined number of clusters. Instead, they group data points based on their local density, considering both the density of the points themselves and their spatial proximity. One of the most widely used density-based clustering algorithms is DBSCAN (Density-Based Spatial Clustering of Applications with Noise), which was introduced by Sander et al. in 1998 [Sander et al. 1998]. DBSCAN is particularly effective in handling datasets with noise and anomalies, as it can identify and separate these points from the main clusters.

5.1 DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a density-based clustering algorithm that groups data points based on two key parameters: *epsilon* (ϵ) and *minimum points* (*minPts*). The algorithm operates by calculating the number of points within a radius ϵ around each point p. If the number of points is greater than or equal to minPts, p is classified as a core point. Points within the ϵ-neighborhood of a core point are assigned to the same cluster, and core points are connected if they are density-reachable. Border points, which lie within the ϵ-neighborhood of a core point but are not core points themselves, are also assigned to the clusters. Points that are neither core nor border points are considered noise or anomalies.

The main advantages of DBSCAN are that it can identify clusters of arbitrary shape and size and is robust to noise and anomalies. However, this technique is sensitive to varying densities and cluster sizes.

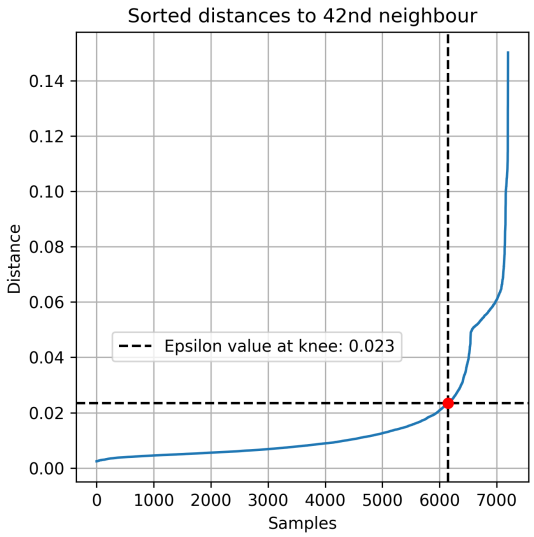

Figure 16: Sorted distances to 42^{nd} neighbour

5.1.1 Determining the Optimal ϵ and *minPts*

Choosing the appropriate parameters ϵ and *minPts* for DBSCAN is crucial for obtaining good clustering results. A common heuristic for determining *minPts* is to set it to twice the number of features in the dataset,

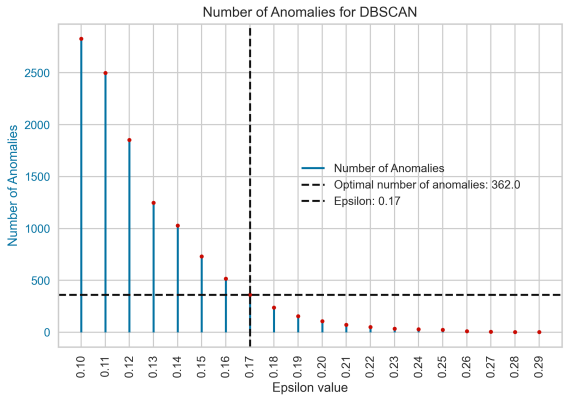

Figure 17: DBSCAN: number of anomalies for each ϵ. For $\epsilon = 0.17$ a total of 362 anomalies was found.

as suggested by Sander et al. [Sander et al. 1998]. In our case, with 21 features, *minPts* was set to 42.
To find a suitable starting value for the ϵ parameter, we computed the distance of each point to its k^{th} nearest neighbor, ordered these distances, and plotted them as shown in Figure [16].

The knee of this curve was identified as an initial estimate for ϵ and performed DBSCAN iteratively within the range $[0.5 \times \text{initial estimate}, 1.5 \times \text{initial estimate}]$. To select the ϵ value, similarly to the distance threshold for Hierarchical Clustering, K-Medoids and K-Prototypes, we chose an ϵ value after which the change in the number of anomalies showed a reduced decline. The chosen ϵ was 0.17 and the curve can be seen in Figure [17].

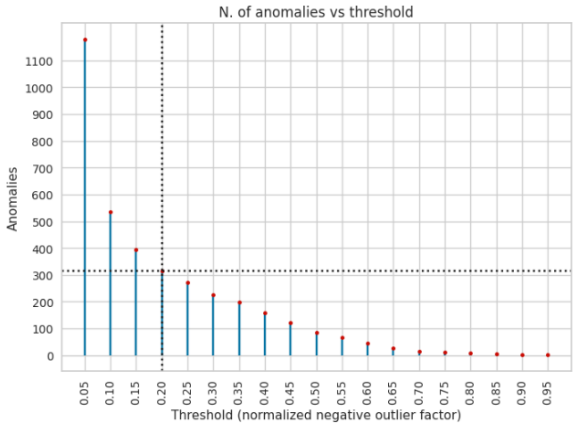

Figure 18: Choosing of the threshold value based on the number of anomalies for Local Outlier Factor algorithm

5.1.2 Identifying Anomalies

To compute the probability of each point being an anomaly using DBSCAN, we followed a systematic approach. For each point, the probability is a combination of the number of neighbors in the ϵ range and the mean distance of those neighbors from the point considered. This way we were able to express the probability of each data point in a continous range. The probability of an element to be anomalous is proportional to the mean distance of the neighbors point and inversely proportional to the number of neighbors in its ϵ range.
The points identified as anomalies by DBSCAN are shown in Figure [19], plotted using FAMD for reducing the dimensionality from 21 features to 3 coordinates.

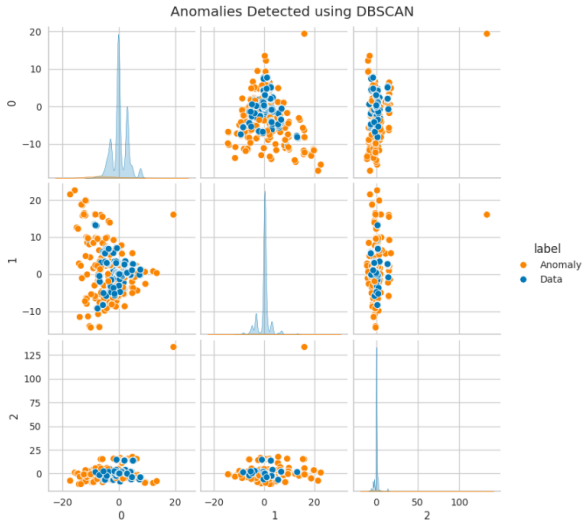

Figure 19: DBSCAN: 645 anomalies visualized using FAMD.

6 Local Outlier Factor

Local Outlier Factor (LOF) measures the local density deviation of a data point relative to its neighbors. Unlike global outlier detection methods, LOF considers the local context of each data point, making it robust to variations in data density. The algorithm assigns an outlier score to each data point, indicating its deviation from the local density of its neighbors. A high LOF score signifies a data point that is significantly less dense than its neighbors, thus classifying it as an outlier.
As with the previous approaches, we calculated the Local Outlier Factor (LOF) using a precomputed Gower distance matrix.
As previously done with other algorithms, we aimed to consider a continuous range of values to indicate the probability of an element being anomalous. The default implementation of the Local Outlier Factor (LOF) from the Python package scikit-learn returns an array containing the anomaly score for each element, expressed on a scale from -1 to 1, where a lower value indicates a higher likelihood of the element being an outlier.

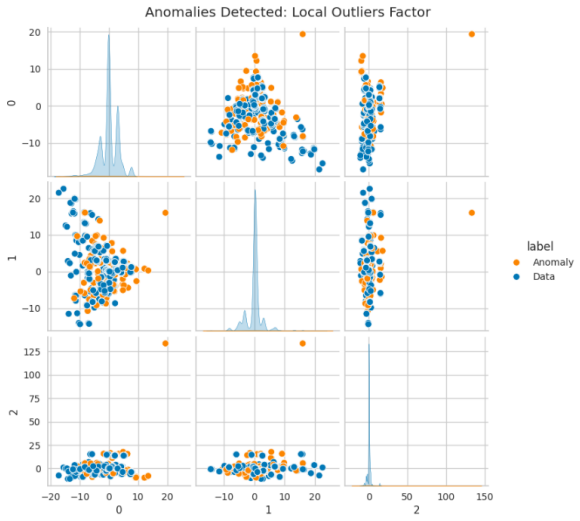

Figure 20: FAMD visualization for the Local Outlier Factor algorithm

We performed a normalization to adjust the probability range between 0 and 1, and subsequently determined a threshold to identify the most appropriate normalized score, based on the number of anomalies detected. Similar to other algorithms, we selected the threshold value where the rate of decrease in the number of anomalies significantly slowed down, as illustrated in Figure [18]. Specifically, we chose a threshold value of 0.2, which corresponds to the detection of 316 anomalies.

To better understand the algorithm's performance on real data, Figure [20] provides an FAMD visualization of the dataset. This visualization clearly indicates that, compared to other approaches we have studied, the performance of LOF is less effective, as evidenced by the more dispersed normal data points.

7 Comparison

Several methodologies have been devised to tackle the issue of outlier detection. This report offers a comprehensive assessment of the efficacy of each algorithm type utilized. The evaluation criteria encompass the detection rate of anomalies by each algorithm, alongside metrics employed to gauge their relative performance and divergence in solutions.

7.1 Adjusted Rand Index

The Adjusted Rand Index (ARI) is an improved version of the Rand Index, which measures the agreement between two partitions of the same dataset. The ARI adjusts this measure by accounting for the possibility of random assignments, thus providing a more accurate reflection of clustering performance. It ranges from -1 to 1, where:

- 1 indicates perfect agreement between the clusterings.
- 0 indicates that the clustering is no better than random.
- Negative values indicate less agreement than expected by chance.

The ARI is computed as follows:

- **Pairs Analysis**: Similar to the Rand Index, the ARI considers all pairs of items in the dataset and checks if they are in the same cluster or in different clusters in both the predicted and true clusterings.
- **Agreement and Disagreement**: For each pair of items, count how many pairs are in the same cluster in both clusterings, in different clusters in both clusterings, in the same cluster in one clustering but in different clusters in the other, and vice versa.
- **Expected Agreement**: Calculate the expected agreement by chance. This step adjusts for the fact that some agreement might happen just by random chance.
- **Calculation**: The ARI is calculated using the formula:

$$\text{ARI} = \frac{\text{Index} - \text{Expected Index}}{\text{Max Index} - \text{Expected Index}}$$

where:

  - **Index** is the number of agreements between the clusterings (i.e., the sum of pairs that are in the same or different clusters in both clusterings)
  - **Expected Index** is the number of agreements that would be expected by chance.
  - **Max Index** is the maximum number of agreements (which is the total number of pairs).

In our specific application context, we utilized the Adjusted Rand Index to assess the dissimilarity between solutions obtained from different algorithms. As illustrated in Table [1], the index score is notably influenced by the threshold criteria applied to determine the number of anomalies, despite keeping the same criteria across all algorithms. Nevertheless, it is evident that hierarchical clustering algorithms employing different linkage methods exhibit comparable scores, with dissimilarities probably attributable to variations in the number rather than the spatial distribution of anomalies.

Adjusted Rand Index	HC Average	HC Complete	DBSCAN	K-Medoids	K-Prototypes	LOF
HC Average	1.00	0.70	0.74	0.54	0.05	0.67
HC Complete	0.70	1.00	0.54	0.41	0.04	0.58
DBSCAN	0.74	0.54	1.00	0.57	0.07	0.47
K-Medoids	0.54	0.41	0.57	1.00	0.12	0.32
K-Prototypes	0.03	0.01	0.04	0.09	1.00	0.03
LOF	0.67	0.58	0.47	0.32	0.05	1.00

Table 1: Adjusted Rand Index for all the possible pairs of algorithms used during the report

Hierarchical clustering using average and complete as linkage method have an ARI of 0.70 with respect to each other. Considering they have a similar number of anomalies (428 and 446), this result show that there is some difference in the clustering solutions but a good level of agreement. It's worth noting that out of the two hierarchical clustering algorithms, average linkage achieves a higher ARI with respect to DBSCAN, K-Medoids and LOF, with values of 0.74, 0.54 and 0.67 respectively, while complete linkage results in 0.54, 0.41 and 0.58.
DBSCAN, despite being density based, shows a high ARI of 0.74 with average linkage method (which is distance based) but a lower ARI of 0.47 with LOF, which is also density based.
K-Prototypes has the lowest ARI scores with all other algorithms indicating a significant dissimilarity in its clustering results to the results of all other methods. This was unexpected due to the suitability of the algorithm for handling mixed data types and requires future work and research.

7.2 Comparing the number of anomalies

In this final subsection, we present a comparison of the number of anomalies detected by each algorithm. To ensure the results are comparable, the same criterion was applied in determining the threshold for each approach: keep as threshold the value before which the number of anomalies start decrease more slowly. As shown in Table [2], the number of anomalies detected was relatively consistent across all algorithms, with the total number ranging from 316 to 428.

8 Conclusions

This study evaluated several Unsupervised Learning methods for anomaly detection. To compute the final probabilities of anomaly, we opted on computing the average probability for each sample across their probabilities with the following algorithms:

	N. of Anomalies
HC Average	428
HC Complete	446
DBSCAN	362
K-Medoids	343
K-Prototypes	328
LOF	316

Table 2: Number of anomalies for each algorithm

- Hierarchical Clustering with Average linkage
- Hierarchical Clustering with Complete linkage
- DBSCAN

The choice of these algorithms is due to the highest ARI scores belonging to these algorithms pairs. The final number of outliers found is equals to 402, leading to the solution visualized in Figure[21].

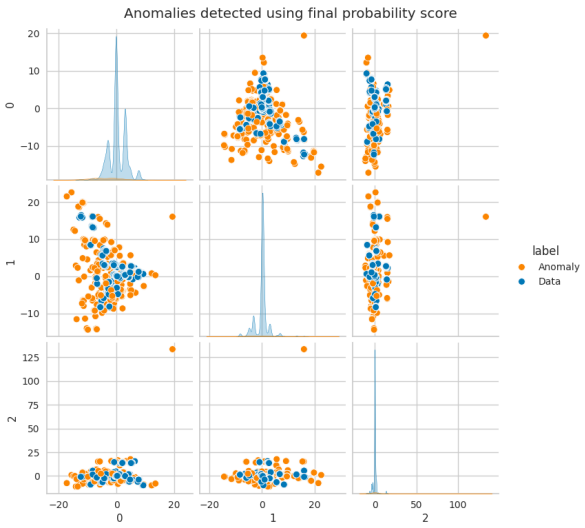

Figure 21: FAMD visualization for the final number of anomalies found

References

[Sander et al. 1998] Jörg Sander, Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu. *Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications. Data Mining and Knowledge Discovery*, 2:169–194, 1998.

[Hubert 1985] L. Hubert and P. Arabie, Comparing Partitions, Journal of Classification 1985

[Scikit-learn] Scikit-learn: Machine Learning in Python. Available at: `https://scikit-learn.org/stable/unsupervised_learning.html`