E.T.S. de Ingeniería Industrial,
Informática y de Telecomunicación

Federated Clustering

Grado en Ingeniería Informática

Trabajo Fin de Grado

Aitor Uranga Roldan

Asier Urio Larrea

Pamplona, 7 de Julio de 2024

upna
Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Contents

List of Figures

List of Tables

Abstract

Federated clustering is a innovative machine learning technique that conducts clustering analysis across multiple decentralized datasets while safeguarding data privacy. This method is crucial in situations where data decentralization is necessary due to privacy concerns, regulations, or limited bandwidth. Data remains localized in federated clustering, with only non-sensitive data being shared. The process involves combining local clustering models from various nodes into a global model using consensus-based methods, distributed optimization, or secure multi-party computation algorithms. By allowing collaborative analysis without exposing raw data, federated clustering enables insights from diverse datasets across organizations or locations, enhancing privacy and security while enabling data-driven decision-making in sectors like healthcare, finance, and telecommunications where confidentiality and compliance are vital. The goal is to see different implementations of federated clustering algorithms and see their performance under different situations

Keywords:

Federated clustering, Machine learning, Clustering analysis, Decentralized datasets, Data privacy, Data decentralization, Global model and Local model

Chapter 1

Introduction

By the use of clustering methods a lot of problems can be solved in different areas. To illustrate, in healthcare, clustering can assist in the identification of disease syndromes by analyzing patient data without disclosing the sensitive information, thus, ensuring the privacy. This method enables healthcare providers to uncover the trends and relations between patient symptoms and medical histories, thus, the diagnosis becomes more accurate and the treatment becomes more personalized. Clustering is also helpful in the management of hospital resources, the prediction of patient admissions, and the improvement of the healthcare services. Through the use of clustering algorithms, medical researchers can discover the hidden trends and relationships in large datasets which will lead to the medical research and public health initiatives being advanced. Principally, the possibility of data analysis while at the same time protecting the patient privacy solves the major ethical and legal issues thus leading to the compliance with the HIPAA, among others.

Clustering is one of the most valuable tools in the field of data science, machine learning, and statistical analysis. This term refers to the algorithms and techniques that are developed to find patterns of the given data.

The application of clustering extends beyond mere categorization but also numerous fields where it can be applied for different purposes, the examples are determining disease

syndromes in medical science, segmenting markets in business, structuring huge information reserves in digital technologies or reducing complex datasets in engineering. Clustering can be useful in efficient data handling, enhances data reduction, and facilitates anomaly detection and pattern recognition, giving crucial information of the distribution and characteristics of data.

However, there is one significant drawback of poor protection of users' privacy when it happens that the data is sourced from private sources. Traditional clustering algorithms neglect privacy since they consider all the obtained data as one set.

The question is how classical clustering algorithms can be adapted to a federated approach and how they will perform under different conditions. Moreover, determine if their performance matches that of the traditional approach for organizing various types of data.

1.1 Objectives

The main objective of this work is to test different federated clustering algorithms and evaluate them under various conditions to determine which is optimal in each case. For this goal, the following steps have been settled:

- Gain a theoretical comprehension of the fields.
- Implement the federated clustering algorithms.
- Test each algorithms for each dataset.
- Compare the results to the expected outcome from the traditional algorithm.
- Compare the results between algorithms.

1.2 Thesis outline

The second chapter consist on the revision of the information to understand the project, an introduction to clustering, including how it works, different uses for clustering and clustering types. In addition with how the Federated Learning works, its basis and how this frameworks is adapted to clustering.

The third chapter explains how each of the tested clustering algorithms work and how they have been adapted to a Federated framework. The algorithms that are included are Gaussian mixture model, K-means, Density-based spatial clustering of applications with noise (DBSCAN) and Spectral Clustering. This chapter include the description of the steps involved on both the traditional and federated approaches of each algorithm.

The forth chapter consist on the explanation of the used experimental framework. It describes the used datasets, how the algorithms are applied to these datasets, and finally, the comparison and analysis of the algorithms.

The fifth chapter contains the results obtained from each of the algorithms. In addition, it includes the analysis of the results and comparison between algorithms, highlighting their strengths and limitations in various scenarios.

The sixth and final chapter is the conclusions of the whole project. It has the key things that have been found, discusses the implication of these findings. Also, it contains possible future directions for future research based on the outcomes of the project and the technical limitations that this project had.

Chapter 2

Background

2.1 Clustering

Clustering is the technique that tries to find groups of data points in multidimensional space based on a similarity metrics [1]. These groups are defined as clusters, where each of them has a unique pattern between them [2]2.1. Clustering is based on the distance (similarity) between points. A common and used on this project distance measure is the Euclidean distance, which calculates the straight-line distance between two points in space. The Euclidean distance between two points $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ in n-dimensional space is given by:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}$$

This distance measure is straightforward and effective for clustering purposes, helping to identify groups of points that are close to each other in the multidimensional space. However, other distance measurements like Manhattan distance, cosine, and Mahalanobis distance can also be employed depending on the nature of the dataset. Manhattan distance is applicable when working with high-dimensional space where the change in individual dimensions matters. Cosine similarity is perfect for text data as well as in any other cases

where the direction of vectors rather than their size does matter. Mahalanobis distance is useful whenever working with correlated predictors as it takes into account the correlations and scales present in the data.

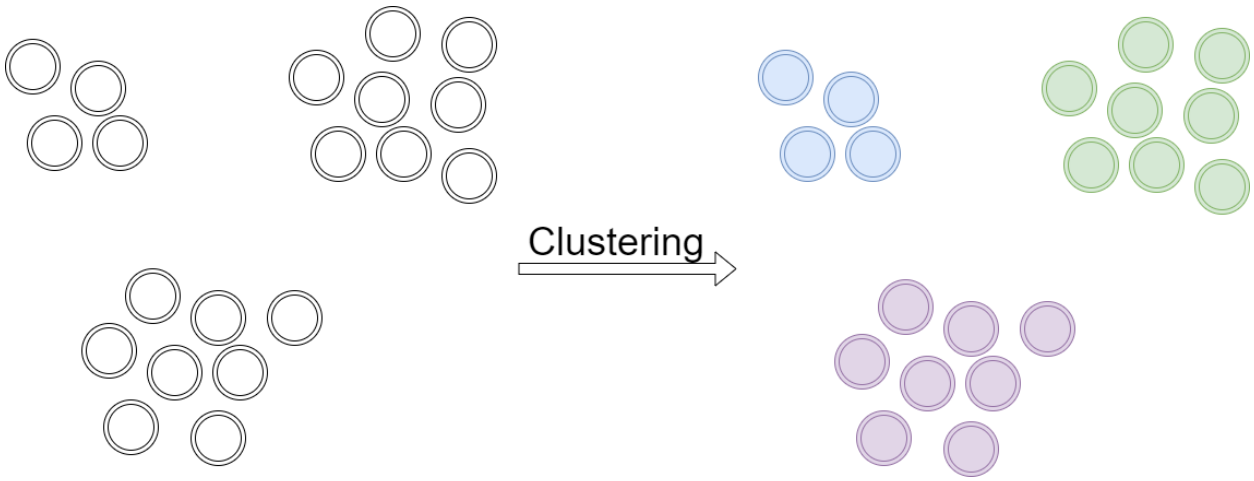

Figure 2.1: Clustering

Clustering is inside the category of unsupervised learning, where in the learning process the input data is not class labelled [3], and it is a difficult problem inside due this unsupervised nature and that every cluster may vary in shape and size[4].

Cluster analysis is a type of tool that can be employed in a number of fields such as business intelligence, image recognition, search engines, security, and others. In business intelligence clustering is communicated in person by segmenting customers into distinct groups according to the factors that they share in common, which makes the process of improving customer relationship management efficient. On the same way, cluster approach is applied in project management to facilitate auditing of projects at one time, and boost the successes of projects. In image recognition like in case of handwritten characters, clustering finds repetitions of the same letters and digits with certain variations in them which increase accuracy. clustering is also used in the web search to condense a big number of search results into a few groups searchable, and is taken up in information retrieval to cluster documents into thematic topics.

Clustering is also widely used as a preprocessing step for other processes, such as char-

acterization and classification. One of the most valuable feature of cluster analysis is its automatic nature. It categorizes groups or classes within the data and is sometimes referred to as auto-classification or data segmentation. This capability is particularly useful in outlier detection, which plays a significant role in areas where detecting this anomalies is crucial, for example in electronic commerce, where this outliers may indicate fraudulent or criminal activities. Given the vast amounts of data being collected and ongoing developments in related fields like machine learning and statistics, cluster analysis not only represents a recent area of research but also holds substantial potential in data mining[3].

The way of accomplishing this clustering can be quite different, resulting in several types of models for grouping data, each of which may be the most suitable for certain type of data and analysis. The aforesaid four types are the Density Model, Distribution Model, Centroid Model, Hierarchical Model. Each of these models approaches the task of clustering using different principles and algorithms to perform the task of division of data into meaningful clusters. These models are applicable to a wide variety of complexities and forms of distribution within data sets, spanning from the simple geometric kind to complex hierarchical cases.

2.1.1 Distribution-Based Method

Distribution-based clustering is a method that can be applied to data that naturally exhibits variability or uncertainty by transforming each point into a probability distribution as opposed to a fixed value. In figure 2.2 can be distinguished the different area probability of each cluster and how each point is clustered by that criteria. It is very helpful especially in areas like text mining where the documents can be modeled via distribution of word frequency or in bio-informatics, where the genetic expression levels of the organisms vary. Distribution-based clustering therefore offers a robust approach to understanding and categorizing data characterized by uncertainty and variability.[5]

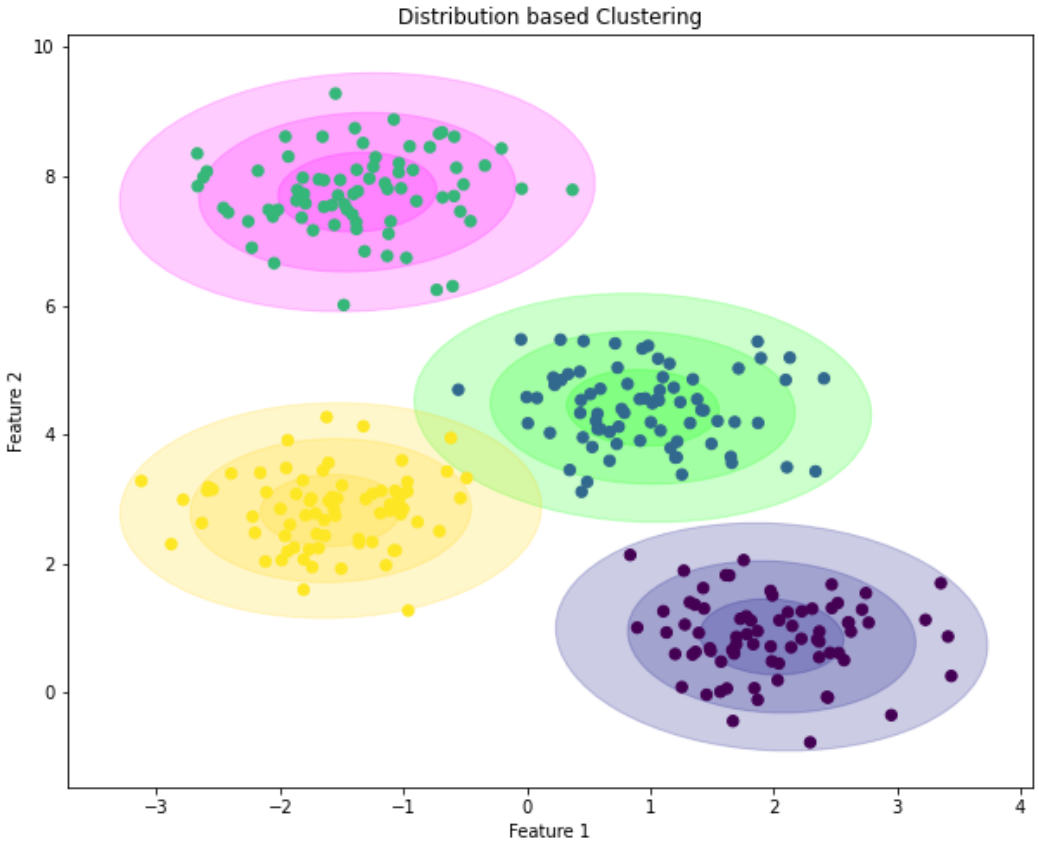

Figure 2.2: Density Based Clustering Distribution

2.1.2 Partitioning or Centroid Based method

Partitioning or Centroid based methods in cluster analysis involve dividing a dataset of 'n' objects into 'k' partitions or clusters, with each cluster containing at least one object. Typically adopting exclusive cluster separation where each object belongs to only one cluster, these methods are primarily distance-based. Is possible to see in figure 2.3 how each point is clustered in the cluster of it respective center. The process begins with an initial partitioning, followed by iterative relocation of objects between clusters to enhance cluster quality—aiming for intra-cluster closeness and inter-cluster separation. Traditional methods like k-means and k-medoids are commonly used, focusing on spherical clusters in not overly

large datasets, but for handling more complex shapes or larger datasets, these methods often extend into subspace clustering.[3]

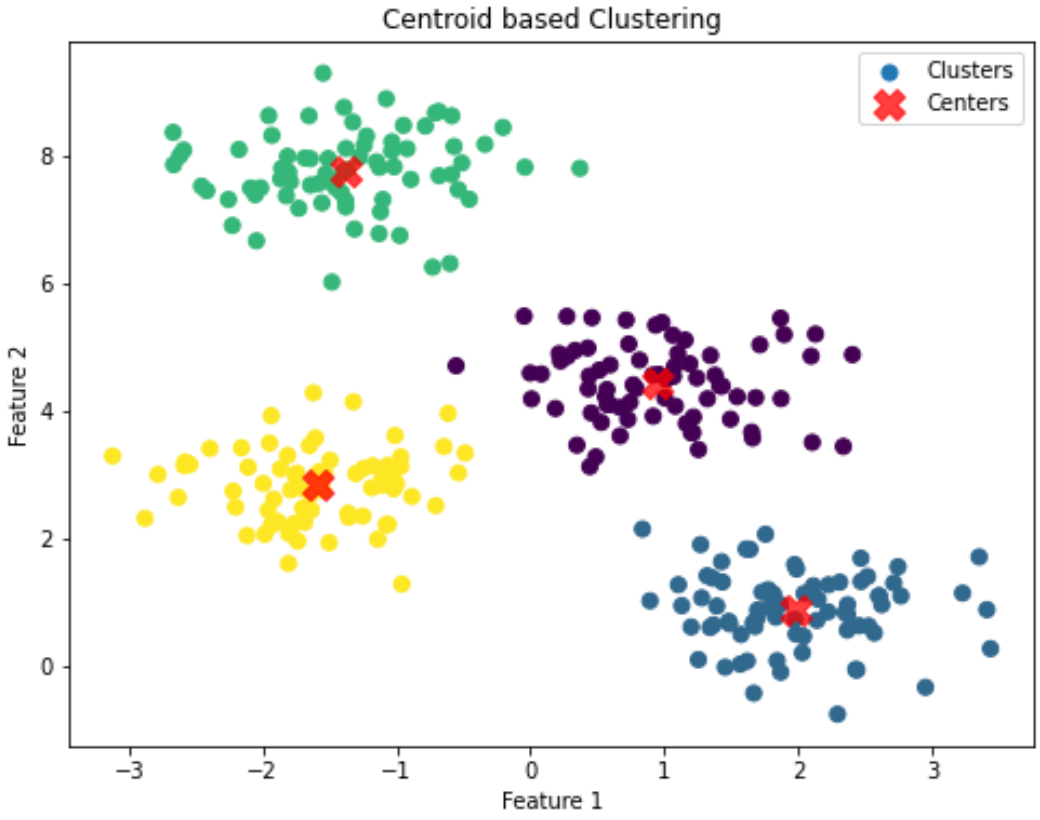

Figure 2.3: Centroid Based Clustering

2.1.3 Hierarchical Method

Hierarchical clustering methods systematically construct a cluster hierarchy either by a bottom-up approach (agglomerative), starting with each object as a separate group and progressively merging them, or by a top-down approach (divisive), starting with all objects in one cluster and iteratively dividing them, as is possible to see in figure 2.4. These methods can be based on distance, density, or continuity and might operate in full or subspace settings. A significant limitation is their irreversible nature; once a merge or split is made, it cannot be undone, leading to potentially irreversible decisions without reevaluation.[3]

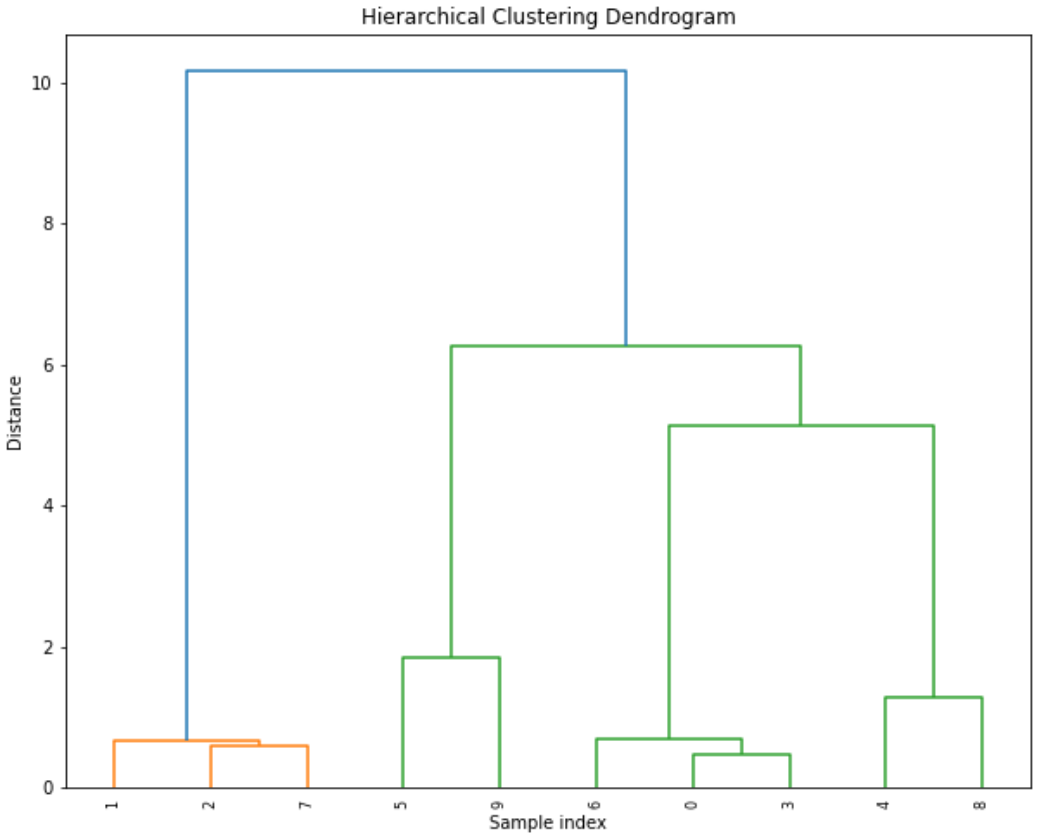

Figure 2.4: Hierarchical Based Clustering

2.1.4 Density-based Method

Density-based clustering utilizes local object density that exceeds a defined threshold to grow the clusters, ideal to identify clusters of arbitrary sizes and shapes and distinguish them from noise or outliers. Can be seen in figure 2.5 how this density of points form the different clusters. This approach tends to form exclusive clusters and is appropriate for scenarios where cluster density varies significantly. Therefore, it is helpful in cases where distance-based approaches may fail to identify non-spherical cluster shapes.[3]

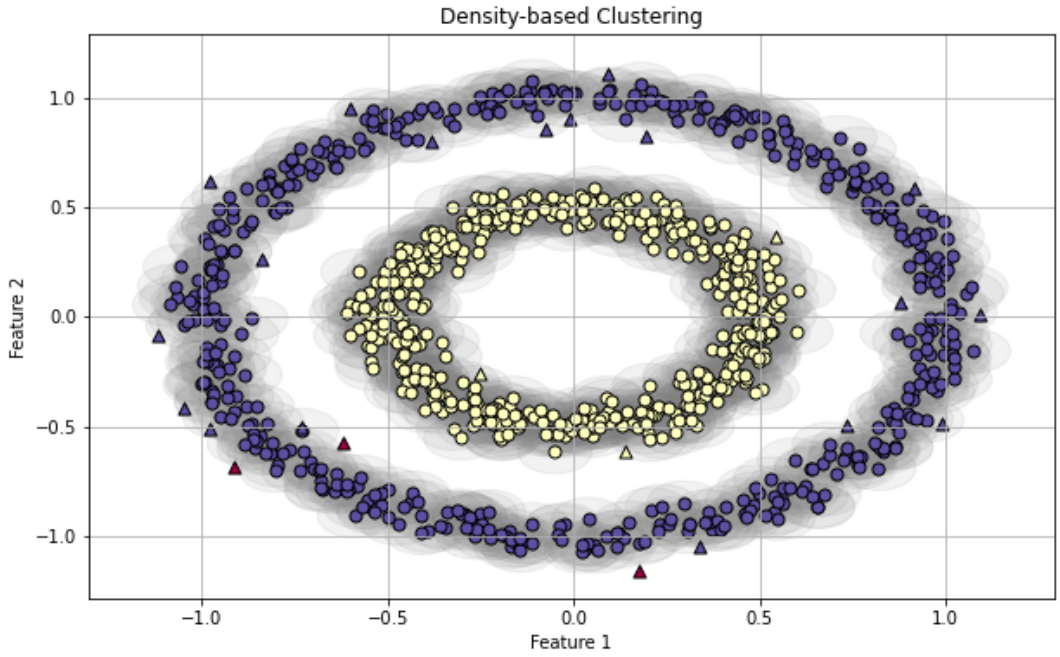

Figure 2.5: Density Based Clustering

2.2 Federated Clustering

In recent years, the usage of devices such as smartphones has increased[6]. At the same time, the amount of data that is being generated by this devices has also risen. However, this data has an important property: it is privacy-sensitive. The privacy-sensitive nature has the downside that the traditional approaches of machine learning fail maintaining the privacy when working with this type of data. In the case of clustering, one way to address this issue is through Federated Clustering.

2.2.1 Federated Learning

When the concept of Federated Learning is integrated with the traditional Clustering algorithms Federated Clustering appears. Federated Learning was mentioned the first time in 2016 by McMahan et al[7]: "We term our approach Federated Learning, since the learning

task is solved by a loose federation of participating devices (which we refer to as clients) which are coordinated by a central server".

Their suggested algorithm includes the application of neural networks and a fixed number of K devices/nodes with predetermined local datasets. At the start of a round, a fraction C of clients is chosen at random and the server sends the latest global neural network parameters to this subset of clients. The clients then process this global state of the server and their respective data and send diff updates back to the server. These updates are collected by the server and reflected into the global state of the server after averaging and the process goes on.

In summary, a global model is learned from the aggregation from the information of local clients as shown in this figure 2.6. The process follows this steps, first there is a central server that has a global model of the system. Additionally, there are clients with their respective dataset and local model. In each of the iterations, the clients download the model from the server, perform a local update of the model with their data, and after they upload it back to the server. Finally, the server does an aggregation of each of the model that the clients have uploaded[8].

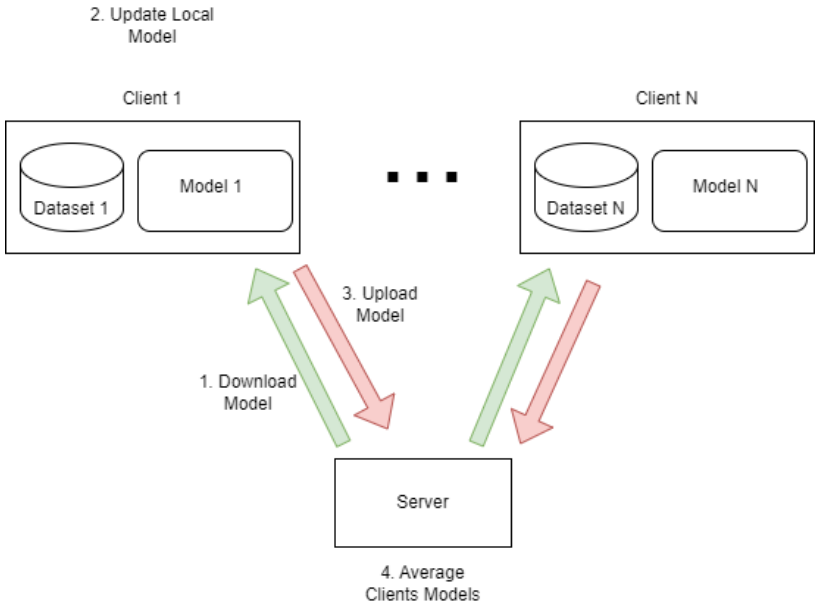

Figure 2.6: Federated Learning Framework

2.2.2 Data distribution in Federated Clustering

Almost all information on Federated Clustering focuses on its application to data that is not independent and identically distributed (non-IID). This characteristic is intrinsic to Federated Learning, as it involves receiving data from each client, which is rarely independent and identically distributed among them [9].

In federated learning, the non-IID data problems will mostly likely come from different data distribution (Pi and Pj) among different clients, which will probably affect the performance of model. Besides that, these distributions can be changed over time, giving rise to the changes defined by the distribution Q which makes matters more complex, changing the level of non-IID of the existing data. [10].

However, this work is centered in comparing Federated Clustering algorithms across different data distributions, where is mainly used IID data to simplify the comparison process and some test with non-IID data to test its performance alsp. In 2.7 and 2.8 there is a simplification to represent IID data and Non-IID data.

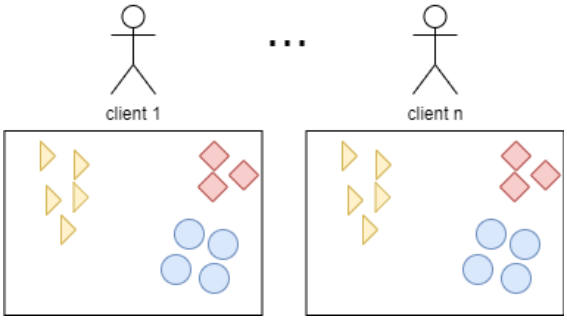

Figure 2.7: IID data

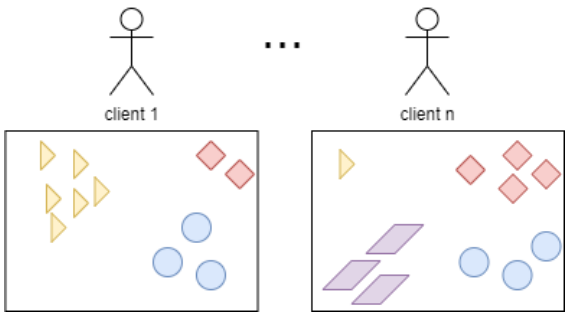

Figure 2.8: Non-IID data

Chapter 3

Federated Clustering Algorithm

For this work, one algorithm from each of the types of Clustering mentioned in Section 2.1 will be analyzed. For the centroid-based method, k-means; for density-based, DBSCAN; for hierarchical, spectral clustering; and for distribution-based, GMM.

3.1 K-means

K-means clustering is a popular unsupervised machine learning method that splits a particular data into a pre-specified amount of groups (referred to as K). Any split of the groups with similar traits and assign them into clusters is the aim of this method, as K-means is the vector quantization method. It is especially effective in the domain of data mining and pattern recognition, which ensures to recognize and proceed with unique categories in the data.[11]

How K-Means Works

K-means algorithm does this by the use of a straightforward and easy to implement iteration procedure that seeks to minimize the sum of distances between points and the centroid of the cluster they are in. At algorithm 1, there is breakdown of the standard steps involved in the K-means algorithm.

Algorithm 1 Standard K-means Clustering Algorithm

Input: Array $X = \{x_1, x_2, \ldots, x_n\}$ // Dataset to be clustered
Output: Clusters $C = \{c_1, c_2, \ldots, c_k\}$ // Number of required clusters and cluster centroids
Initialize k cluster centroids $C = \{c_1, c_2, \ldots, c_k\}$ randomly
repeat
    // Distance calculations
    for each i in 1 to n **do**
        for each j in 1 to k **do**
            Compute the Euclidean distance from data object x_i to centroid c_j
        end for
        // Data object assignment
        Assign data object x_i to the closest cluster c_j
    end for
    // Update cluster centroids
    for each j in 1 to k **do**
        Compute the new centroid c_j as the mean of all data objects assigned to cluster c_j
    end for
until the centroids do not change significantly between iterations or a maximum number of iterations is reached

Its outcome may often be found near a starting point, and this algorithm can have problems with the convergence to local minima.To counteract these issues, multiple runs with different initial centroids can be performed, along with techniques like the K-means++ initialization strategy for choosing smarter starting centroids.

How Federated K-Means Works

In the case of Federated K-means is done by sharing the center of the local models[12]2.2. The steps of Federated K-Means are shown on algorithm 2.

3.2 DBSCAN

DBSCAN is one of most prominent clustering techniques which identifies clusters in the dataset by means of density of data points. It is widely recognized as a good algorithm for sorting a mixed set of the clusters with both unusual shapes and sizes, as well as being more

Algorithm 2 Federated K-Means Clustering

Initialization:
Server initializes the global centroids $\{C_g\}$.
Server shares $\{C_g\}$ with selected clients.
for each round of communication **do**
    Local Learning:
    Choose a percentage of the clients K.
    for each client k in K in parallel **do**
        Client k performs local K-means clustering:
            Initialize local centroids with $\{C_g\}$.
            Perform several iterations of K-means on local data $\{D_k\}$.
            Obtain updated local centroids $\{C_k\}$.
        Client k sends $\{C_k\}$ to server.
    end for
    Centroid Aggregation:
    Server aggregates local centroids $\{C_k\}$:
        $\{C_g\} = \frac{1}{K} \sum_{k=1}^{K} \{C_k\}$.
    Iteration:
    Server sends updated global centroids $\{C_g\}$ back to clients.
end for
Convergence:
Repeat the process until convergence or for a predefined number of rounds.

stable in the presence of noise or outliers.[13]

How DBSCAN Works

DBSCAN operates on a few key principles and parameters:

- Epsilon (ϵ): Epsilon is the radius where points withing or close to this distance are considered neighbors.

- MinPts: This is the minimum number of points needed to make a condensed point area, a cluster of a dense region.

The algorithm classifies points into three types:

- Core Points: A point is a core point if at least it has MinPts points within distance (ϵ) of the point, including itself.

- Border Points: An edge is defined as having MinPoints within a distance of (ϵ) but still relatively close to a core point.

- Noise Points: Noise point is the point other than the core point and border point.

The steps of DBSCAN are shown in Algorithm 3.

How Federated DBSCAN Works

In the case of federated DBSCAN, the approach taken to fit the federated model involves dividing the data space into cells and calculating the density in each cell. This information is then uploaded to the global model. This approach is based on [14] is described on algorithm 4.

Algorithm 3 DBSCAN Clustering Algorithm

Input: Dataset D, distance threshold ϵ, minimum points $MinPts$
Output: Clustered data and noise
for each point P in dataset D **do**
    if P is not visited **then**
        Mark P as visited
        Fetch all points within ϵ-distance proximity of P (the neighbors)
        if the number of neighbors $\geq MinPts$ **then**
            Mark P as a core point and start a new cluster
            Add P to the cluster
            for each neighbor N of P **do**
                if N is not yet assigned to a cluster **then**
                    Add N to the cluster
                    if N is a core point **then**
                        Fetch N's neighbors
                        Recursively expand the cluster
                    end if
                end if
            end for
        else
            Mark P as noise (this may change later)
        end if
    end if
end for
End Procedure: Every point is either assigned to a cluster or marked as noise

Algorithm 4 Federated DBSCAN Algorithm

Input: Dataset D, granularity level L, minimum points $MinPts$
Output: Clustered data and noise
Local Computation:
for each client **do**
    Create a dictionary to count the number of points in each grid cell based on granularity level L
    Determine which grid cells contain at least $MinPts$ points to be considered dense
end for
Global Aggregation:
The central server aggregates local updates from all clients
Determine which grid cells across all clients are dense
Cluster Formation:
Using aggregated dense cell information, the central server initiates clusters starting from globally recognized dense cells
Apply a variant of the DBSCAN algorithm to cluster these cells, considering both core and border cells across client boundaries
Assignment:
for each client **do**
    Assign cluster labels to individual points based on the cell they belong to
    Mark points in sparse cells as noise
end for

3.3 Gaussian Mixture Model

A Gaussian Mixture Model (GMM) is a probabilistic model which is used to explain the density function of the given dataset in terms of a weighted sum of a number of Gaussian distributions. GMMs fall under the mixture models and are useful in modeling data which is naturally organized in clusters. They are commonly employed in clustering, density estimation, and as a model for generating data from complex distributions. The operation of GMM is described by the Algorithm 5 in [15].

1. **Assumption of Distribution:**

    - A GMM assumes each cluster follows a normal distribution.
    - "Mix" in GMM refers to the use of multiple Gaussians to model the data, with each Gaussian distribution representing a cluster.

2. **Parameterization:**

   - Each mixture component (a Gaussian distribution) has a mean, which is the center of the cluster, and a covariance, which determines the shape and orientation of the cluster.
   - The mixing coefficient indicates the proportion of each Gaussian component in the model.
   - Covariance structures can be full, tied, diagonal, or spherical.

3. **Expectation-Maximization (EM) Algorithm:**

   - The parameters of the GMMs are typically fitted using the EM algorithm, which alternates between two steps:
     (a) **Expectation (E) Step:** Calculate the probability (responsibility) that each data point belongs to each cluster, based on the current parameters of the Gaussians.
     (b) **Maximization (M) Step:** Update the parameters (means, covariances, and mixing coefficients) of each Gaussian to maximize the likelihood of the data given these responsibilities.

4. **Convergence:**

   - The EM algorithm iterates between the E and M steps until the parameters of the Gaussians and the likelihood of the data stabilize, showing insignificant changes upon further iterations.

5. **Result:**

   - GMM provides not only the cluster memberships for each data point but also the probabilities of each data point belonging to each cluster, as determined by the associated Gaussian distributions.

- This characteristic classifies GMM as a soft clustering method, contrasting with hard clustering approaches like k-means.[16]

How GMM Works

Algorithm 5 Gaussian Mixture Model (GMM) Algorithm

Input: Dataset $X = \{x_1, x_2, \ldots, x_n\}$, number of components K
Output: Parameters of the Gaussian mixtures: π_k, μ_k, Σ_k for $k = 1, 2, \ldots, K$
Initialization:
Initialize the means μ_k, covariances Σ_k, and mixing coefficients π_k for each component k
repeat
    Expectation (E-step):
    for each data point x_i **do**
        Compute the responsibility $\gamma(z_{ik})$ for each component k using:

$$\gamma(z_{ik}) = \frac{\pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_i|\mu_j, \Sigma_j)}$$

    end for
    Maximization (M-step):
    for each component k **do**
        Update the parameters using the responsibilities:

$$N_k = \sum_{i=1}^{n} \gamma(z_{ik})$$

$$\mu_k = \frac{1}{N_k} \sum_{i=1}^{n} \gamma(z_{ik}) x_i$$

$$\Sigma_k = \frac{1}{N_k} \sum_{i=1}^{n} \gamma(z_{ik})(x_i - \mu_k)(x_i - \mu_k)^T$$

$$\pi_k = \frac{N_k}{n}$$

    end for
until convergence criteria are met (e.g., log-likelihood change is below a threshold)

How Federated GMM Works

This approach to Federated GMM works similarly to the federated version of K-means. Instead of sharing all the data with the server, only the means, covariances, and precision matrices are uploaded, thereby preserving the privacy of the users. The steps of this approach can be seen in algorithm 6.

Algorithm 6 Federated Gaussian Mixture Model (GMM) Algorithm

Input: Datasets D_i for each user $i = 1, 2, \ldots, M$, number of components K
Output: Global GMM parameters: π_k, μ_k, Σ_k for $k = 1, 2, \ldots, K$
Initialization:
Initialize global means μ_k, covariances Σ_k, and mixing coefficients π_k for each component k
Distribute initial parameters to each client
repeat
    Choose K clients at random from all the clients being K a percentage of the clients
    Local GMM Training:
    for each client i in K **do**
        Train a GMM on local dataset D_i using the EM algorithm with the received initial parameters
        Compute local parameters: π_k^i, μ_k^i, Σ_k^i for $k = 1, 2, \ldots, K$
        Send local parameters to the central server
    end for
    Parameter Aggregation:
    Aggregate the received local parameters to create global parameters
    for each component k **do**
        Aggregate means: $\mu_k = \frac{1}{M} \sum_{i=1}^{M} \mu_k^i$
        Aggregate covariances: $\Sigma_k = \frac{1}{M} \sum_{i=1}^{M} \Sigma_k^i$
        Aggregate mixing coefficients: $\pi_k = \frac{1}{M} \sum_{i=1}^{M} \pi_k^i$
    end for
    Global Model Update:
    Update the global parameters based on the aggregated values
    Send updated global parameters back to each client
    Distribution and Iteration:
    for each client i **do**
        Use the updated global parameters for further local GMM training
    end for
until convergence criteria are met (e.g., log-likelihood change is below a threshold)
Model Utilization:
Utilize the final global model for cluster analysis and density estimation

3.4 Federated Spectral Clustering

Unlike the conventional clustering methods that usually work on the original features directly, spectral clustering works by the analysis of the similarity graph of the data. This graph consists of nodes that are data points, and the edges show the similarities between these points. The primary steps in spectral clustering are building a similarity graph, a Laplacian matrix of the graph is computed, and using the eigenvalues and eigenvectors of this matrix to transform the data into a space where the clusters can be more easily separated[17]. The process is shown in algorithm 7.

How Spectral Clustering Works

Algorithm 7 Spectral Clustering Algorithm

1: **Input:** Dataset $X = \{x_1, x_2, \ldots, x_n\}$, number of clusters k, similarity metric
2: **Output:** Cluster assignments for data points
3: **Similarity Graph Construction:**
4: Construct a similarity graph $G = (V, E)$ where V is the set of data points and E represents the edges weighted by their likeness according to the chosen similarity metric
5: **Laplacian Matrix:**
6: Compute the Laplacian matrix $L = D - A$ where A is the adjacency matrix of G and D is the degree matrix
7: **Eigenvalue Decomposition:**
8: Compute the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ and corresponding eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ of the Laplacian matrix L
9: **Clustering in Lower Dimensions:**
10: Select the first k eigenvectors corresponding to the smallest non-zero eigenvalues
11: Form a matrix U with these eigenvectors as columns
12: Apply a conventional clustering technique (e.g., k-means) to the rows of U to obtain the final cluster assignments

Step 1: Similarity Graph Construction

Starting the Spectral Clustering Algorithm requires a construction of similarity graph $G = (V, E)$, where V is a set of the points of given data, and E is a set of the likenesses of the weights according to a chosen similarity metric. It involves computing the similarity between two data points and or features and plotting them into a graph in form of nodes.

The connections or the edges in the graph are then assigned certain values similar to the values calculated previously for the associations between data points, with the higher value of connection meaning that the two data points are more similar to each other.

Step 2: Laplacian Matrix

The next step involves the computation of the Laplacian matrix L which encapsulates the structural characteristics of the similarity graph G. The Laplacian matrix L is defined as $L = D - A$ where L is the Laplacian matrix of G, A is the adjacency matrix of G, which reflects the similarity of nodes. And D is the degree matrix of G, which keeps information about the number of neighbors each node has in the graph.

Step 3: Eigenvalue Decomposition

Once the Laplacian matrix L is obtained, the eigenvalue decomposition is done to find the eigenvalues $\lambda_1, \lambda_2, ..., \lambda_n$ and corresponding eigenvectors $v_1, v_2, ..., v_n$. The eigenvectors represent the directions of maximum variance in the dataset, while the eigenvalues provide information about the amount of variance captured by each eigenvector.

Step 4: Clustering in Lower Dimensions

The next step is to pick the first k eigenvectors with the lowest eigenvalues that are greater than zero and arrange them in a U matrix which is formed with these eigenvectors in its columns. By choosing a number of eigenvectors smaller than the total number of data points in the dataset reducing the dimensionality of it, while preserving the most relevant information.

Step 5: Apply Conventional Clustering Technique

In the last step, we perform any general clustering algorithm like k-means applied on the rows of the matrix U in order to get the final cluster partition of the given data. K–means clustering is employed in partitioning the data into k clusters depending on the reduced

feature space derived from the spectral decomposition. This step allows the efficiency of clustering relevant clusters in the given data set and spectral information that the low-dimensional space reflects.

How Federated Spectral Clustering Works

For the Federated approach of Spectral Clustering, the steps of this approach are shown in algorithm 8.

Algorithm 8 Federated Spectral Clustering

Input: Client list $clientList$, number of clients $numOfClients$, number of nodes $numOfNodes$, number of clusters $numOfClusters$, power iteration per client $iters$, global aggregation rounds $globalRounds$
Output: List of labels S of N nodes after global clustering
Client Code:
function GETPOWERITERATIONCLIENT($client$, $iters$, $eigenVectors$)
    Internal Variables: $numOfNodes$, L, I
    $L = I -$ Normalized Laplacian matrix
    for $iter$ in range($iters$) **do**
        $eigenVectors = L \times eigenVectors$
    end for
    Return: $eigenVectors$
end function
Server Code:
function FEDSPECTRALSERVER($clientList$, $numOfClients$, $numOfNodes$, $numOfClusters$, $iters$, $globalRounds$)
    Internal Variables: v
    v: initialize a $numOfNodes \times numOfClusters$ matrix randomly
    for $round$ in range($globalRounds$) **do**
        $p = numOfClients$ copies of v
        for $client$ in range($numOfClients$) **do**
            $clientId = clientList[client]$
            $p[client] =$ getPowerIterationClient($clientId, numOfClusters, iters, p[client]$)
        end for
        $v =$ average of all matrices in p
        $v, r = qrDecomposition(v)$
    end for
    $globalLabels =$ KMeansClustering(v)
    Return: $globalLabels$
end function

Client side:

1. **Initialization**:

   - Each client initializes with their local normalized Laplacian matrix L, which is derived from their data's similarity graph. This matrix represents the local topology or the structure of the data.
   - An identity matrix I of the same dimension as L is also prepared.

2. **Matrix Modification**:

   - Clients modify the Laplacian matrix to $I - L$, preparing it for the power iteration process.

3. **Power Iteration**:

   - The client performs a series of power iterations on the eigenvectors. This involves repeatedly multiplying the modified Laplacian matrix with the eigenvectors to refine them. The number of iterations ($iters$) is predetermined by the server.

4. **Sending Results to Server**:

   - After completing the specified iterations, each client sends their updated eigenvectors back to the server. These vectors are crucial as they encapsulate the structural insights of the local data.

Server-Side:

1. **Initialization**:

   - The server begins by initializing a random matrix v. This matrix has dimensions corresponding to the total number of nodes and the number of clusters. It serves as the starting point for aggregating the eigenvectors from all clients.

2. **Global Rounds of Aggregation**:

   (a) *Distribution*:

   - At the start of each global round, the server distributes the current version of matrix v to each client.

   (b) *Local Updates*:

   - Clients receive the matrix, run their local `getPowerIterationClient` function to update it, and then send it back to the server.

   (c) *Aggregation*:

   - Once all clients have returned their matrices, the server averages these matrices. This step consolidates the updates from all the clients, integrating local information into a global context.

   (d) *Orthonormalization*:

   - After averaging, the server performs a QR decomposition on the resulting matrix to ensure the columns are orthonormal. This process prepares the matrix for effective clustering by removing any linear dependencies and standardizing the vectors.

3. **Clustering**:

   - Using the aggregated and processed matrix v, which now represents the global eigenspace, the server applies a k-means clustering algorithm. This step assigns a cluster label to each node based on its position in the eigenspace.

4. **Output of Global Labels**:

   - Finally, the server outputs the global labels, which effectively categorize each node into a cluster. These labels are the result of collaborative spectral clustering, reflecting both local structures and global coordination.[18]

Chapter 4

Experimental Framework

The primary objective of this experimental framework is to assess and compare the performance of various federated clustering algorithms across multiple datasets. This comparison aims to identify which algorithms are more robust and perform better across different types of data distributions.

4.1 Datasets

In all the datasets, the data was split into five clients to simulate a federated environment. This distribution models real-world scenarios where data is decentralized, allowing the performance of federated clustering algorithms to be evaluated under practical conditions.

4.1.1 IID datasets

Dataset	**Number of Classes**	**Number of Attributes**	**Number of Instances**
Blobs	4	2	3000 per client
Moons	2	2	3000 per client
Circles	2	2	3000 per client
Complex	2	6	2237 per client

Table 4.1: Datasets Information

- **Blobs Dataset:** A standard test pattern, consisting of randomly distributed blobs or clusters, ideal for evaluating basic clustering capabilities.

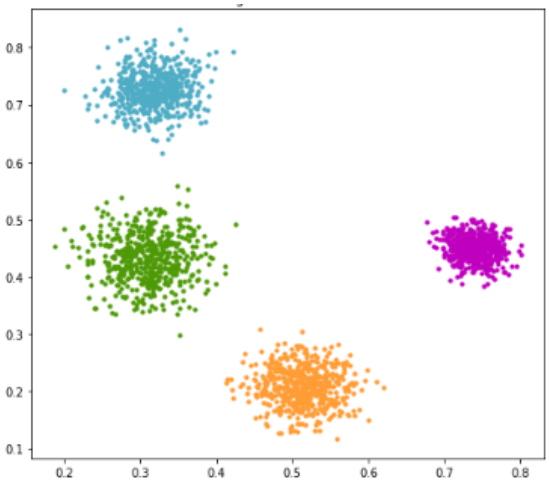

Figure 4.1: Blobs train dataset of a client

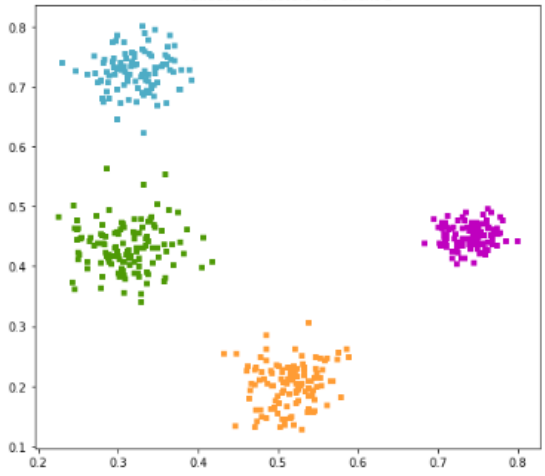

Figure 4.2: Blobs validation dataset of a client

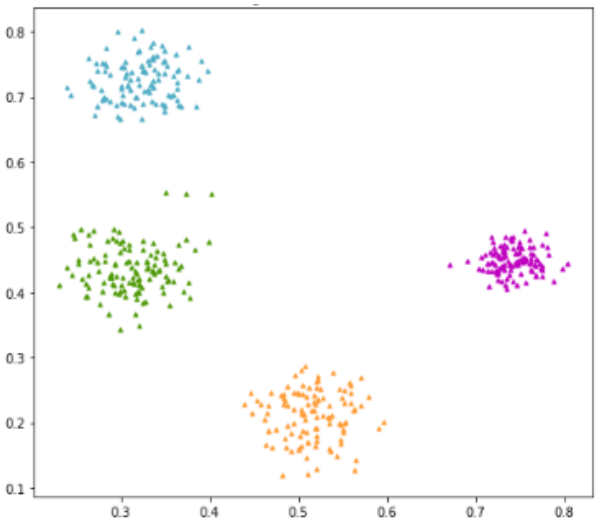

Figure 4.3: Blobs test dataset of a client

- **Circular Dataset:** Features points arranged in a circular fashion, challenging the algorithms' ability to differentiate circular boundaries.

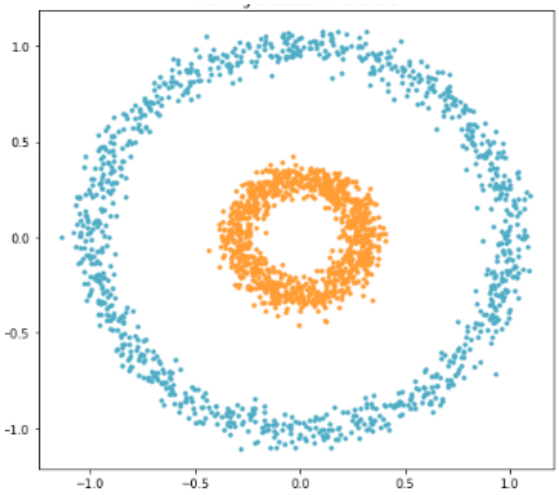

Figure 4.4: Circle train dataset of a client

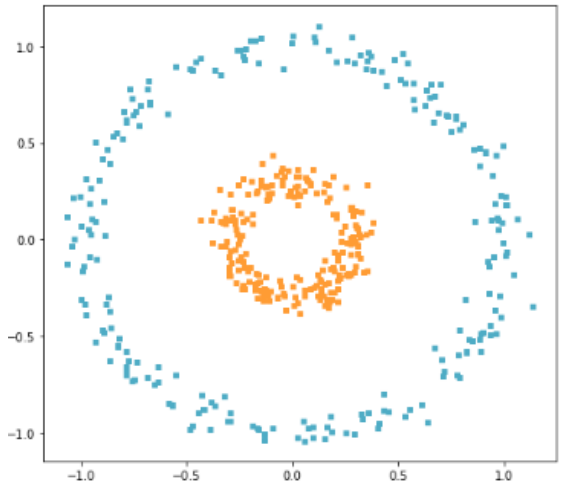

Figure 4.5: Circle validation dataset of a client

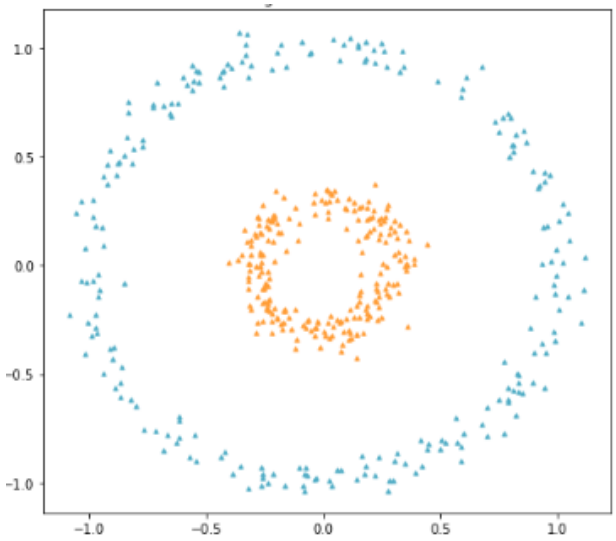

Figure 4.6: Circle test dataset of a client

- **Moons Dataset:** Comprises two interleaving half circles, offering a non-linear separation challenge.

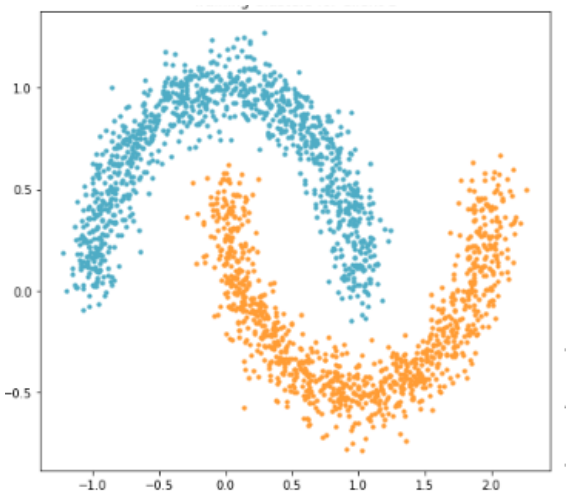

Figure 4.7: Moons train dataset of a client

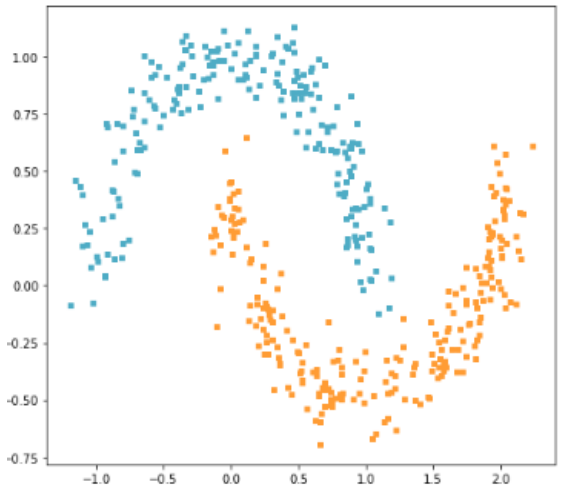

Figure 4.8: Moons validation dataset of a client

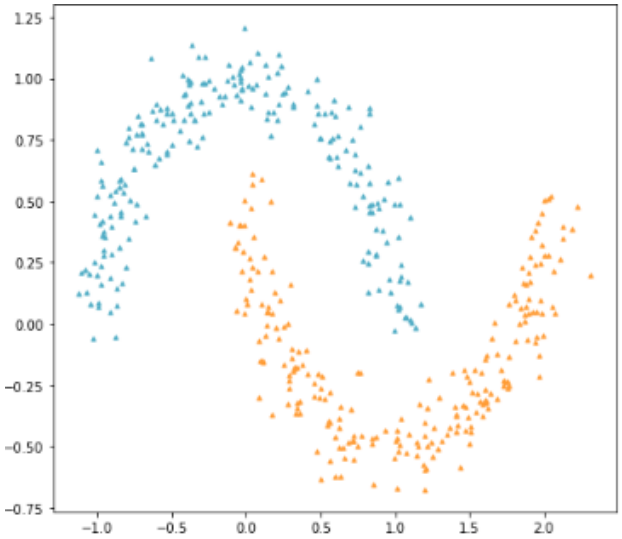

Figure 4.9: Moons test dataset of a client

- **Complex Dataset:** This is real dataset taken from https://data.world/datasets/clustering, that is used to be more challenging for the algorithms having more complex structure.

4.1.2 Non-IID dataset

In this dataset generation for federated learning, we further strived to make the given data non-IID (non-Independent and Identically Distributed) to imitate real-life scenario. First, we created the first random centers and standard deviation of three classes with two attributes.

These initial parameters were then varied slightly for each of the five clients, meaning that both distributions and values changed per client. This variation included making small independent, unpredictable changes to the means and standard deviations with that each client has a different number of samples for each class, contributing to the non-IID nature, so that the distribution of clients' data was unique.

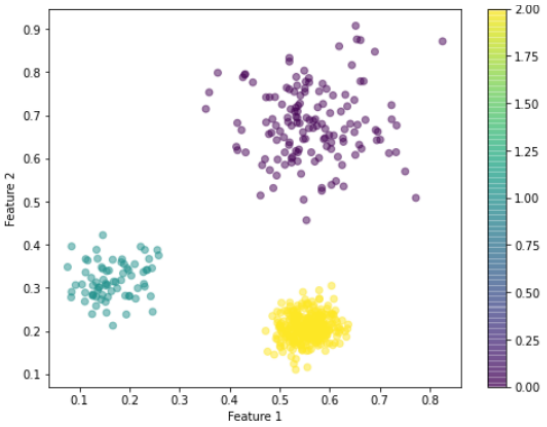

Figure 4.10: Non-IID data distribution of client 1

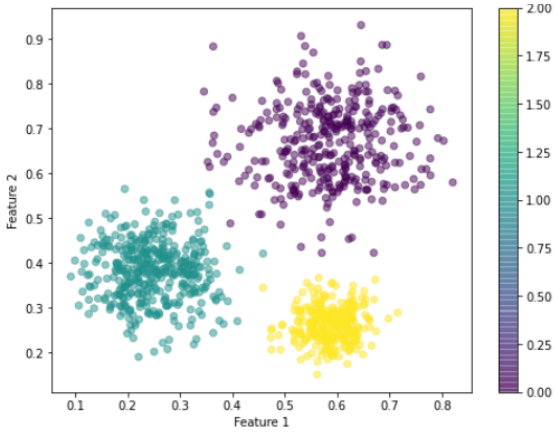

Figure 4.11: Non-IID data distribution of client 2

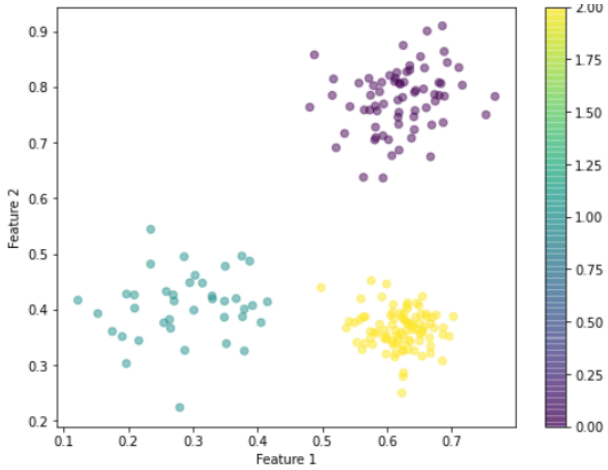

Figure 4.12: Non-IID data distribution of client 3

4.2 Federated Clustering Algorithms

The algorithms that are going to be tested are the ones mentioned on chapter 3: Federated K-means, Federated DBSCAN, Federated GMM and Federated Spectral Clustering.

4.3 Methodology

Each IID dataset is distributed in a simulation federation system which consists of multiple clients, where each client holds a part of the total amount of data. This data is equally distributed among the nodes.

Each federated clustering algorithm is applied independently to all the datasets. Algorithms operate under the federated learning scheme commented in section 2.2.1.This means that each model is trained locally on each client's data and updates the corresponding information to the server; the information depends on the algorithm used, as described in section 3.

In the realm of clustering evaluation, two primary types of scores are employed: internal and external ratings. The external scores employ the ground-truth labels as a criterion for measuring the success of clustering methods, which are compared to predefined labels. Typical external metrics may be the Adjusted Rand Index (ARI) or the Normalized Mutual Information (NMI). The ARI quantifies the dissimilarity between two data clusterings by adjusting for the random grouping of elements, while NMI determines the amount of shared information between the clustering and the ground truth, scaled accordingly to guarantee their comparability. On the other hand, the internal index does not involve tags from the outside; on the contrary, it measures the quality of the grouping from the data itself. Exemplary internal metrics are the Davies-Bouldin Index and the Dunn Index. The Davies-Bouldin Index focuses on the separation and compactness of the clusters, looking to get smaller value that indicates better clustering, while the Dunn Index aims at finding the smallest distance between clusters to the largest within-cluster distance, with higher values suggesting clearer delineation of clusters. This is an important distinction when the true labels are either missing or too costly to obtain. As a result, internal scores become the highly reliable option for a wide range of real-life applications.[19]

The evaluation metric that is used for the most of the experiment is Silhouette Score.The silhouette score is an internal evaluation metric that is often favored in clustering tasks

because it has strong assessment abilities owing to the fact that it does not need ground truth labels. With regard to exploratory data analysis when one does not know in advance the patterns and groupings within the data, this feature provides a great benefit. Unlike the external metrics, which only judge the efficiency of clustering based on the pre-defined labels, silhouette score can be used to measure both the cohesion within clusters and the isolation between them, and that would be an intuitive and comprehensive evaluation of the clustering structure that purely comes from the data itself. This self-sufficiency gives the silhouette method an irreplaceable place among the cluster evaluation tools, especially in unsupervised learning situations where often we do not have labels.[20]

How the Silhouette Score is Calculated

The silhouette score for each data point involves a few steps:

1. **Calculate $a(i)$ for each data point i:**

    - $a(i)$ is the average distance between i and all other data points within the same cluster. This measures how well i is matched to its own cluster, known as cohesion.

2. **Calculate $b(i)$ for each data point i:**

    - $b(i)$ is the smallest average distance of i to all points in any other cluster, of which i is not a member. This value represents the distance between i and the nearest cluster, which measures how well i is separated from the nearest cluster, referred to as separation.

3. **Compute the silhouette score $s(i)$ for each data point i:**

    - The silhouette score for each data point i is calculated using the formula:

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

- This formula yields a value between -1 and +1:
  - A score close to +1 indicates that the data point is well-matched to its own cluster and poorly matched to neighboring clusters.
  - A score close to 0 indicates that the data point is on or very close to the decision boundary between two neighboring clusters.
  - A score close to -1 indicates that the data point might have been assigned to the wrong cluster.

4. Average the silhouette scores of all individual points to determine the overall silhouette score for the dataset.[20]

However, for the testing of non-IID data, in addition to silhouette score, ARI will be used. The Adjusted Rand Index (ARI) is the corrected version of the Rand Index that measures the similarity of two clusterings of the same dataset. The ARI goes further than RI that is defined as the number of pairs in two partitions that are in the same cluster in both partitions divided by the total number of pairs of objects in the two partitions, by correcting for the fact that random clusterings might produce some degree of similarity by chance.[21]

$$RI = \frac{a+b}{a+b+c+d}$$

where:

- a is the number of pairs of elements that are in the same cluster in both partitions.
- b is the number of pairs of elements that are in different clusters in both partitions.
- c is the number of pairs of elements that are in the same cluster in the first partition but in different clusters in the second partition.
- d is the number of pairs of elements that are in different clusters in the first partition but in the same cluster in the second partition.

How ARI Score is Calculated

The ARI adjusts the RI for the chance grouping of elements. The ARI is given by:

$$ARI = \frac{RI - E[RI]}{\max(RI) - E[RI]}$$

where:

- RI is the Rand Index.
- $E[RI]$ is the expected value of the Rand Index (the RI value we would expect if the clusterings were random).
- $\max(RI)$ is the maximum value of the Rand Index (which would be 1 for identical clusterings).

To compute the ARI, the following steps are typically followed:

1. **Construct a Contingency Table**: Construct a contingency table that cross-tabulates the counts of elements in the clusters of the two partitions. This table provides the basis for calculating the necessary pair counts (a, b, c, and d).
2. **Calculate Pair Counts**: Using the contingency table, calculate the values for a, b, c, and d.
3. **Compute Expected Index**: Compute the expected index $E[RI]$, which represents the expected similarity between two random clusterings. This involves combinatorial calculations to account for the number of ways pairs can be grouped.
4. **Compute ARI**: Finally, compute the ARI using the formula provided above.

The ARI ranges from -1 to 1, where:

- 1 indicates perfect agreement between the clusterings.

- 0 indicates that the similarity between the clusterings is what would be expected by chance.

- Negative values indicate less agreement than would be expected by chance.

The score is computed to find the best parameters for each of the algorithms performing a parameter tunning to find the optimal configuration and assess clustering performance. For this purpose, the dataset of each client is divided into three parts: train, which is used to train each model; validation, which tests the results of each combination of parameters, taking the greatest score as the best parameter combination; and test, which evaluates the final result of the trained model on that dataset using the optimized parameters.

In the experimental evaluation of our clustering algorithms, various parameters were tested to ensure robust performance across different scenarios. Specifically, the Federated K-means parameters were derived based on the number of clusters (K), the proportion of clients participating in each round (client percentage), the total number of rounds of communication between the server and clients (rounds), the number of local training iterations in each client (epochs), and the batch size used in the client's computations (batch). These parameters are very important in maintaining the right balance between local computations and global aggregation necessary for high end clustering performance. For the Federated DBSCAN, we highlighted the granularity level, which defines the size of the grid cell that is suitable for density estimation and the number of points required to form a dense cluster, that is, MinPts. These are important parameters that enable core points and clusters to be determined in a distributed manner. Similarly, for the Federated GMM, the hyperparameters considered was the number of components in the mixture model and the percentage of clients in each round. The number of components parameter directly regulates the model's fitness of approximating the distribution of data and separating the data points into meaningful classes.

4.4 Comparison and Analysis

The results of each clustering algorithm on each dataset are kept and evaluated together. The analysis is going to consider how effective the methods are in dealing with data distributions of various types as well as the geometries present in a federated environment and compare them to their traditional counterpart.

Visualization of results will be performed through a cluster plot or performance metrics charts, which will help in identifying the strengths and weaknesses of each algorithm across various scenarios.

This experimental framework aims to provide comprehensive insights into the capabilities and limitations of different federated clustering algorithms when faced with varied data structures and distributions in a federated learning context. This comparison will help in understanding which algorithms are more versatile and effective in a decentralized and privacy-preserving environment.

Chapter 5

Results and Analisys

The results are organized first by each algorithm and their performance in each dataset. After this there is the comparison between algorithms on each of them analysing the strong points and downsides of each of them.

5.1 Results Federated K-Means

This first table, 5.1, contains the Silhouette Scores obtained from Federated K-Means, while the second table, 5.2, shows the results from the traditional framework across different datasets. These results indicate that the use of the federated framework generally results in a slightly negative difference suggesting similar capabilities to the traditional approach.

Blobs	**Moons**	**Circle**
0.79	0.49	0.47

Table 5.1: Results Federated K-Means

Blobs	**Moons**	**Circle**
0.80	0.493	0.476

Table 5.2: Results K-Means

Apart from the score the best way to see how good the algorithm is working when working with clustering is visualizing it.

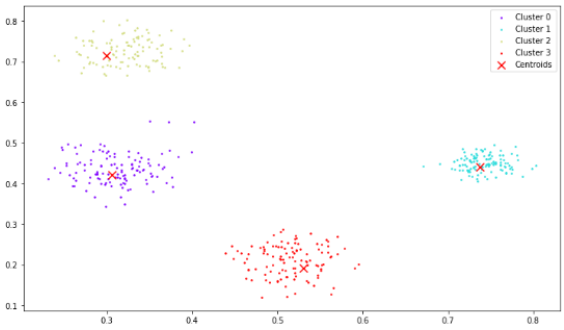

Figure 5.1: Federated K-Means in Blobs dataset

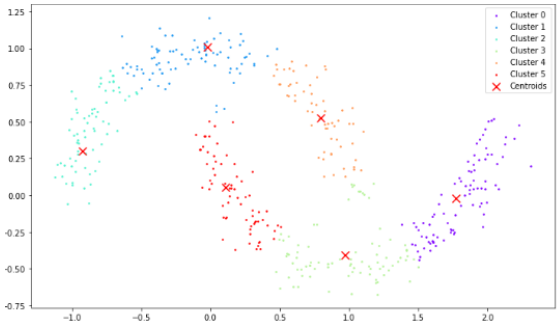

Figure 5.2: Federated K-Means in Moons dataset

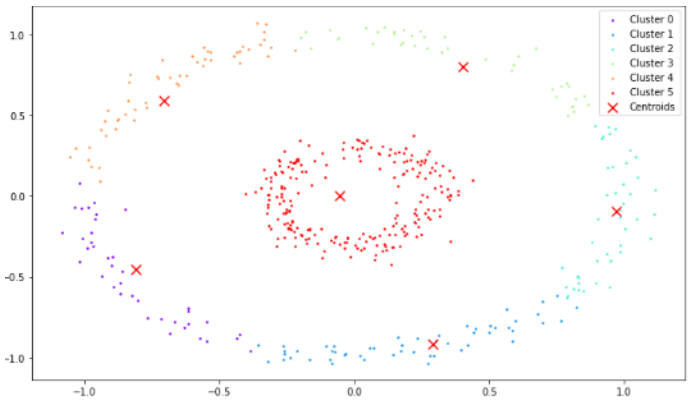

Figure 5.3: Federated K-Means in Circles dataset

In this second round of results, its possible to see that the results are very similar to the previous one, confirming that the federated approach does not have a significant difference in performance with the traditional one.

Blobs	**Moons**	**Circle**
0.762	0.471	0.60

Table 5.3: Results Federated K-Means second distribution

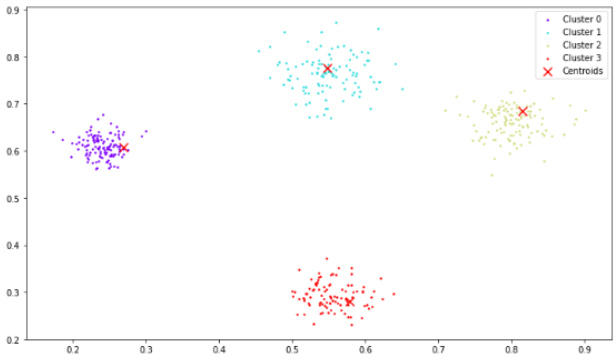

Figure 5.4: Federated K-Means in Blobs dataset second distribution

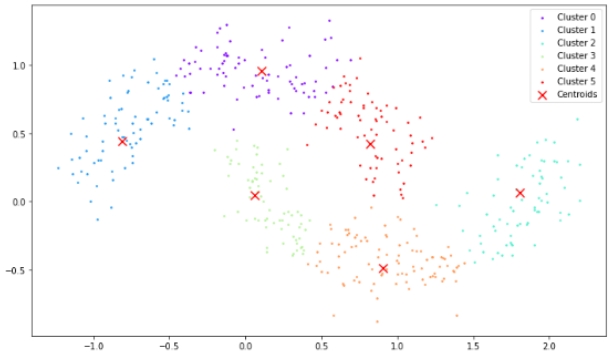

Figure 5.5: Federated K-Means in Moons dataset second distribution

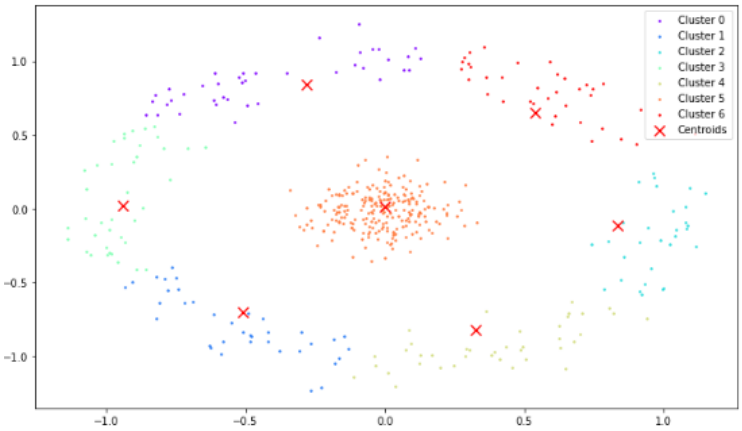

Figure 5.6: Federated K-Means in Circles dataset second distribution

Blobs	**Moons**	**Circle**
0.762	0.47	0.60

Table 5.4: Results K-Means second distribution

5.1.1 Complex Dataset results GMM

Regarding the complex dataset, it achieved a silhouette score of 0.75. At first glance, this might seem like a very good score, and indeed it is, but this score was reached with 5 clusters. Given that the dataset actually has 2 clusters, this indicates poor clustering performance despite the high silhouette score. The discrepancy suggests that the algorithm incorrectly identified the number of clusters, leading to an overestimation and thus misrepresenting the

true structure of the data.

5.1.2 Non-IID data results Federated K-means

Applying the K-means algorithm on non-IID data has shown good results and high accuracy of 0.73 in the case when clusters are well separated. However, because non-IID distributions holds and data instances may not exist identically over different clients, K-means clustering has been observed strong in partitioning data points into well-spaced clusters.

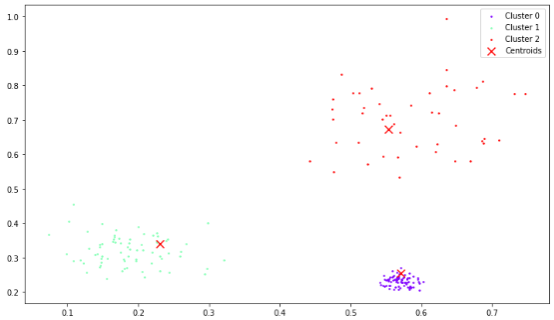

Figure 5.7: Federated K-Means in non-IID dataset client 1

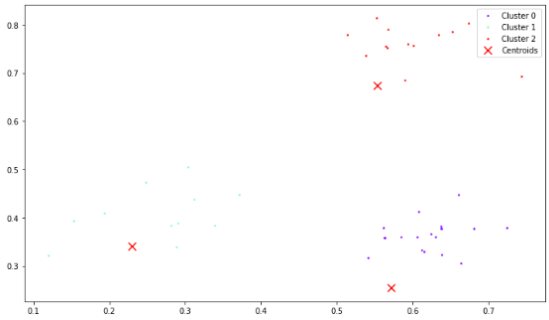

Figure 5.8: Federated K-Means in non-IID dataset client 2

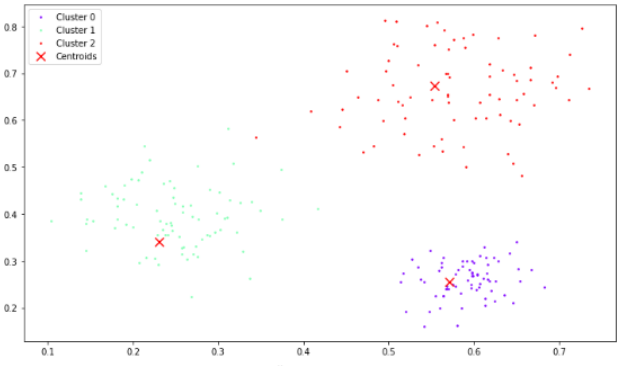

Figure 5.9: Federated K-Means in non-IID dataset client 3

However, despite of having a high silhouette score of 0.69 its possible to see that it encounters difficulties in accurately identifying clusters when they are more diffuse when using silhouette to validate. In scenarios where the boundaries between clusters are not well-defined, K-means struggles to correctly group the data points.

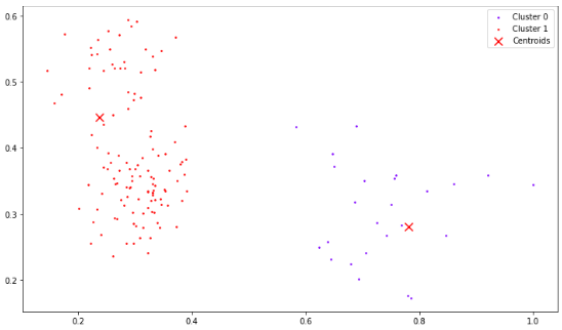

Figure 5.10: Federated K-Means in non-IID dataset client 1

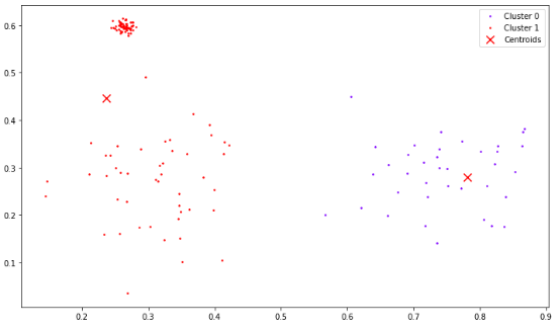

Figure 5.11: Federated K-Means in non-IID dataset client 2

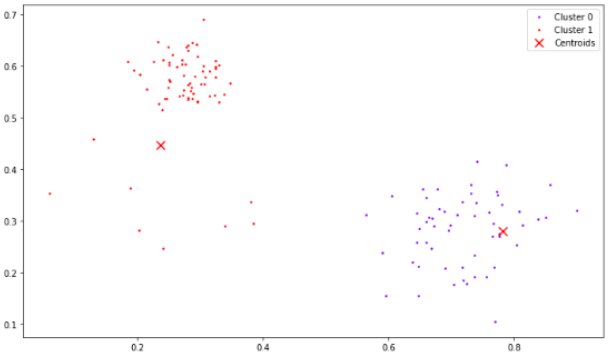

Figure 5.12: Federated K-Means in non-IID dataset client 3

However, if ARI is employed for searching the best combination of the hyperparameters, it is possible to receive a much clearer insight of the dataset distribution. This better insight helps in the identification of clusters and also improves the performance of the clustering algorithm. In the particular case of our application, this caused an improved ARI score of 0.97 which is remarkably high.

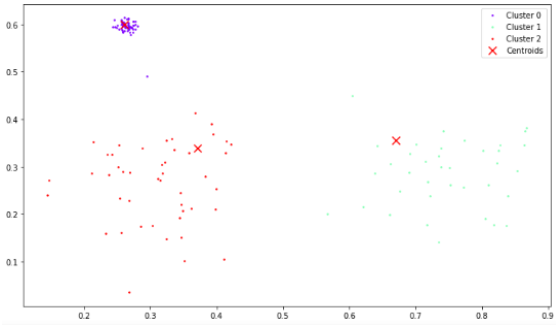

Figure 5.13: Federated K-Means in non-IID dataset client 1 using ARI

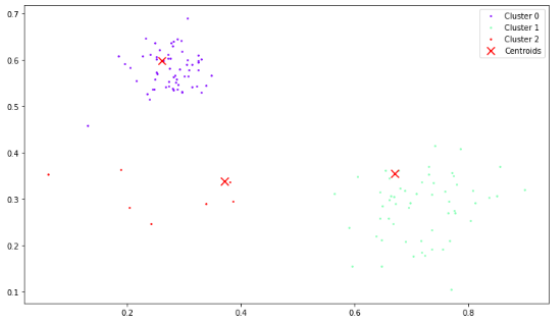

Figure 5.14: Federated K-Means in non-IID dataset client 2 using ARI

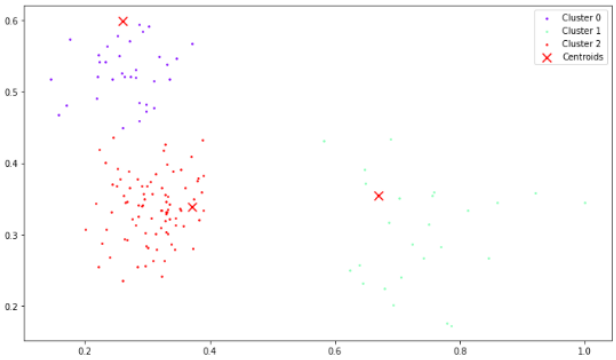

Figure 5.15: Federated K-Means in non-IID dataset client 3 using ARI

5.1.3 Analisys

As the classical approach Federated K-means clustering is a very good tool for blob-shaped clusters since its technique of finding the mean of the points in each cluster is used to update the centroids of the clusters. This method of calculation precisely reflects the center in isotropic clusters that have the same variance in all directions. The blob-shaped clusters are the ones that have these traits—they are well-defined, have globular shapes with little overlap—and therefore, K—means efficiently divides the data into different groups. Therefore, the algorithm's assumption that the data shape is convex clusters is a perfect fit for the raw data distribution of the clusters.

Nevertheless, also encounters the hardships with non-linear shapes like circles, moons or more complex forms. The fundamental rule of K-means that the clusters are convex is not

true for the circular or crescent shapes where the mean does not exactly define the center of the cluster. The problems are increased when the clusters have the same centers or have different densities, as K-means puts the points which are closer to the center of another cluster into the wrong cluster therefore, the wrong clustering is caused. This problem shows federated K-means' failure to deal with the intricate, non-linear data structures.

5.2 Results Federated DBSCAN

Same as in the section 5.1, the differences gotten in Federated DBSCAN, table:5.7, versus the results from the traditional, table:5.8 for a first distribution of the data, its clear that in almost all the cases the use of the Federated Framework has no big impact on the performance of the algorithm.

Blobs	**Moons**	**Circle**
0.768	0.252	0.196

Table 5.5: Results Federated DBSCAN first distribution

Blobs	**Moons**	**Circle**
0.768	0.274	0.161

Table 5.6: Results DBSCAN first distribution

Despite of the score of Federated DBSCAN 5.7, if the analysis of the visualization is done, its possible to see that DBSCAN works much better in non-linear data structures.

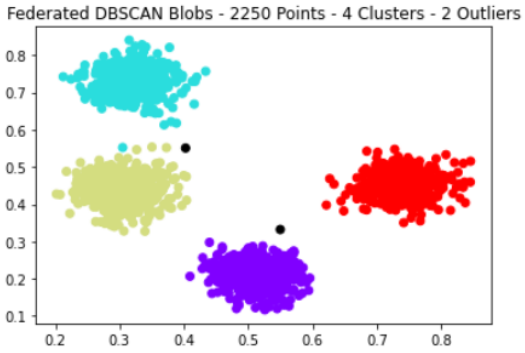

Figure 5.16: Federated DBSCAN in Blobs dataset first distribution

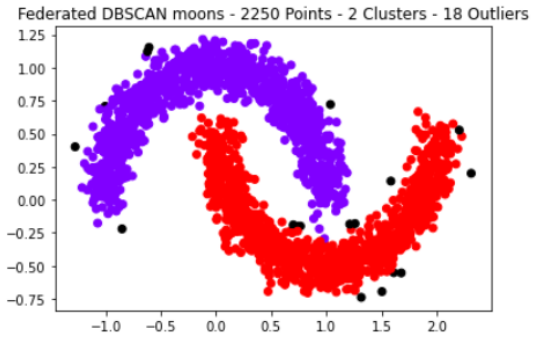

Figure 5.17: Federated DBSCAN in Moons dataset first distribution

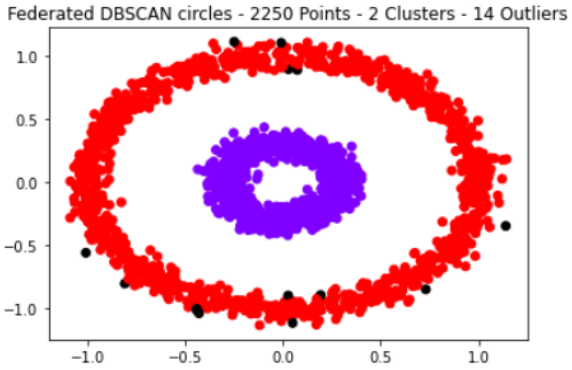

Figure 5.18: Federated DBSCAN Circles in Circles dataset first distribution

In this second round of results, its possible to see that when the cluster are more diffuse,the performance difference is greater than in the previous results. Showing that the performance of the traditional framework is better than the federated one in scenarios with less clearly defined clusters.

Blobs	**Moons**	**Circle**
0.64	0.224	0.196

Table 5.7: Results Federated DBSCAN second distribution

Blobs	Moons	Circle
0.751	0.242	0.275

Table 5.8: Results DBSCAN second distribution

Although, with the visualization, it is possible to see that the performance of Federated DBSCAN is still good. The main downside of this implementation is that the outlier detection is too severe, resulting in false outliers despite them being near the clusters, clear examples in the figures 5.19 5.21.

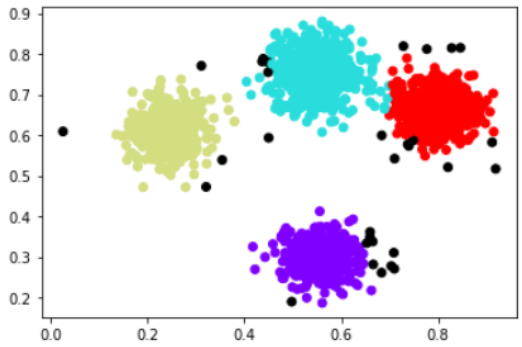

Figure 5.19: Federated DBSCAN in Blobs dataset second distribution

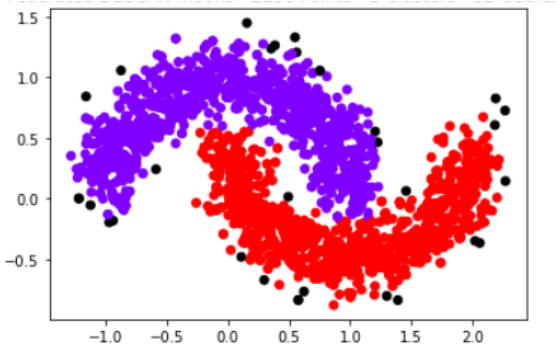

Figure 5.20: Federated DBSCAN in Moons dataset second distribution

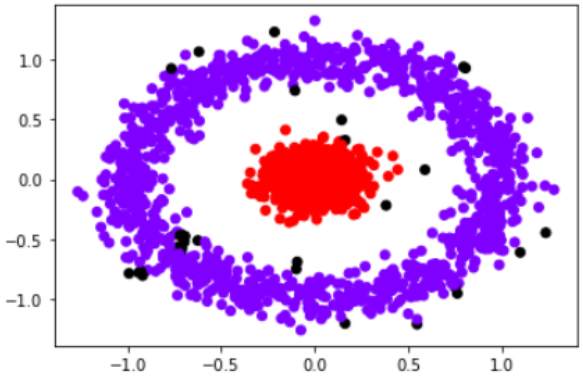

Figure 5.21: Federated DBSCAN Circles in Circles dataset second distribution

5.2.1 Complex Dataset results GMM

In the case of the complex dataset, Federated DBSCAN, despite having a not having a very high silhouette score, having 0.635, accurately identifies the correct number of clusters in the dataset, which is two. This demonstrates that the Federated DBSCAN approach is effective for datasets with complex structures. Federated DBSCAN's ability to correctly determine the number of clusters, even in challenging scenarios, underscores its utility for handling complex datasets with non-linear or irregular cluster shapes.

5.2.2 Non-IID data results DBSCAN

The Federated DBSCAN algorithm has its strengths when the clusters are evenly distributed in the space. The algorithm is efficient in the sense that it is able to correctly identify the clusters and separate them in the given scenarios thus providing efficient clustering results. This makes it ideal for data sets where clusters are well defined and separated with some equal distance between them for the algorithm reaching an score of 0.61 on the first non-IID dataset.

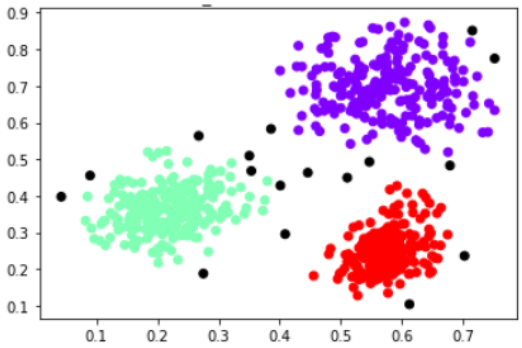

Figure 5.22: Federated DBSCAN in non-IID dataset 1

In the other hand, when encountering narrow, blurred cluster boundaries in a non-IID distribution has an issues of not being able to differentiate between clusters with the use of silhouette score to search the best hyperparameters, it reaches an score of 0.60, This is so because variations are inherently present alongside absence of clear boundaries; the latter

complicates the definition of the core points and the possibility to differentiate clusters within the DBSCAN, we can see this in figure5.23.

However, using the ARI solves this problem, enabling Federated DBSCAN to correctly identify the clusters while also increasing the number of detected outliers, resulting in an improved score of 0.82. This is possible to see in figure 5.24.

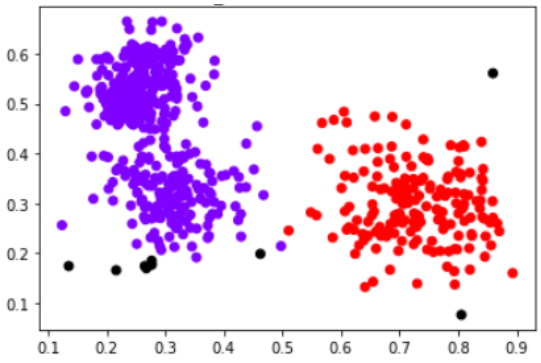

Figure 5.23: Federated DBSCAN in non-IID dataset 2 with silhouette

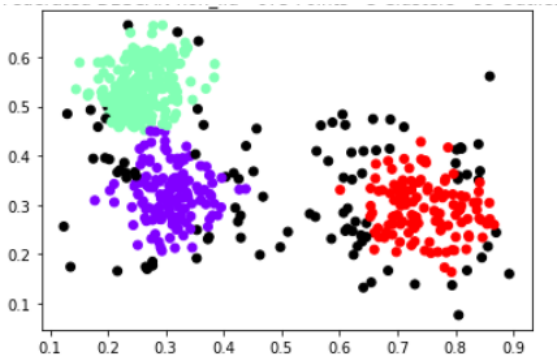

Figure 5.24: Federated DBSCAN in non-IID dataset 2 with ARI

5.2.3 Analisys

Just like the classic approach, Federated DBSCAN is very useful for detecting dense clusters in complicated data structures, as it assigns clusters to core and border points, and neglects the noise. This method that is based on density is suitable for data sets that have different densities and irregular shapes, therefore, it is good at treating non-linear data such as circles and crescents. In crowded areas, DBSCAN effectively clusters the closely packed points together and at the same time, it separates them from the less dense areas hence, it can form the clusters accurately without presuming any specific shape of the clusters.

5.3 Results Federated GMM

Blobs	Moons	Circle
0.80	0.49	0.47

Table 5.9: Results Federated GMM

Blobs	Moons	Circle
0.80	0.49	0.39

Table 5.10: Results GMM

Like in the previus results the best way to ensure the results is trought the visualization of clusters:

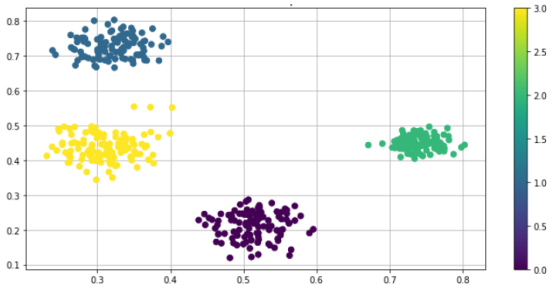

Figure 5.25: Federated GMM in Blobs dataset

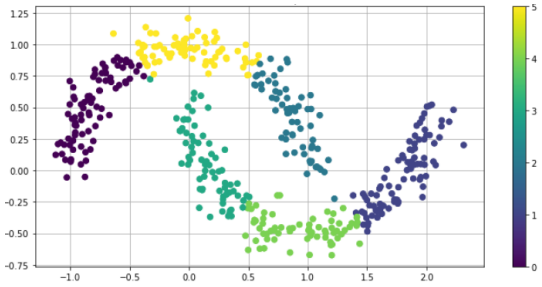

Figure 5.26: Federated GMM in Moons dataset

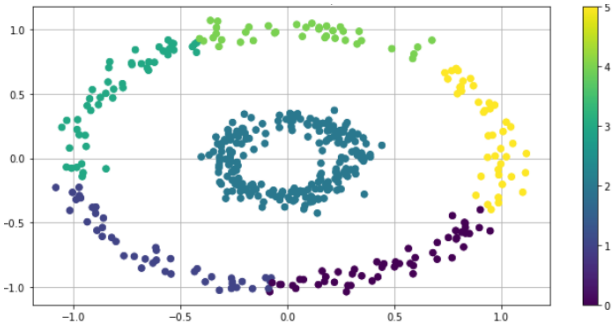

Figure 5.27: Federated GMM Circles in Circles dataset

Its possible to notice that its general performance its very similar to the results obtained in the Federated K-means with a lowering in the complex dataset in both of the distributions.

Blobs	Moons	Circle	Complex Dataset
0.76	0.49	0.60	0.67

Table 5.11: Results Federated GMM second distribution

Blobs	Moons	Circle	Complex Dataset
0.76	0.50	0.60	0.719

Table 5.12: Results GMM second distribution

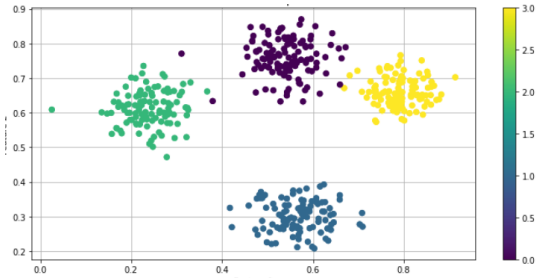

Figure 5.28: Federated GMM in Blobs dataset second distribution

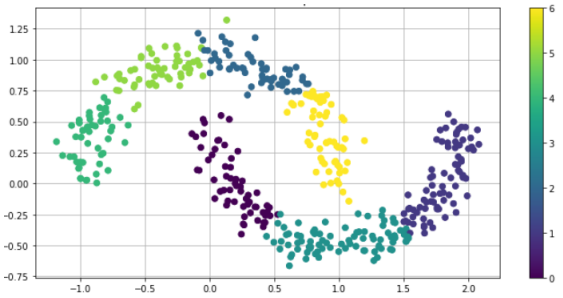

Figure 5.29: Federated GMM in Moons dataset second distribution

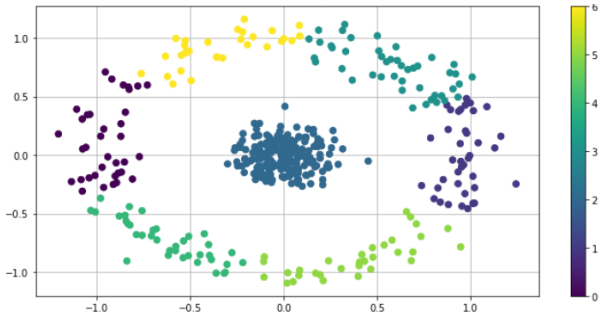

Figure 5.30: Federated GMM Circles in Circles dataset second distribution

5.3.1 Complex Dataset results GMM

In the case of the complex dataset, Federated GMM stands out as a viable alternative among clustering algorithms. It achieves a silhouette score of 0.68, surpassing Federated DBSCAN, while also identifying a more appropriate number of clusters compared to K-means. With

3 clusters maximizing the silhouette score, Federated GMM strikes a balance between clustering accuracy and model complexity. This demonstrates its effectiveness in providing a nuanced understanding of complex datasets, making it a valuable tool in federated learning scenarios where data privacy is a priority.

In the case of the complex dataset, Federated GMM stands out as a viable alternative among clustering algorithms. It achieves a silhouette score of 0.68 and identifying almost correctly the amount of cluster. The number of clusters that maximizes the score for Federated GMM is 3, which, although not perfectly matching the true number of clusters, provides a good insight of the data. This demonstrates that Federated GMM offers a balanced approach, delivering a reasonable trade-off between clustering accuracy and the silhouette score for complex datasets.

5.3.2 Non-IID data results GMM

The Federated GMM model has been able to accurately recognize the proper set of clusters in the given dataset, despite some unclear borders between clients. It achieved a silhouette score of 0.70.

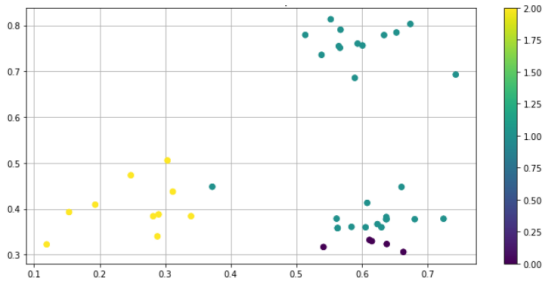

Figure 5.31: Federated GMM client 1 in non-IID dataset 1

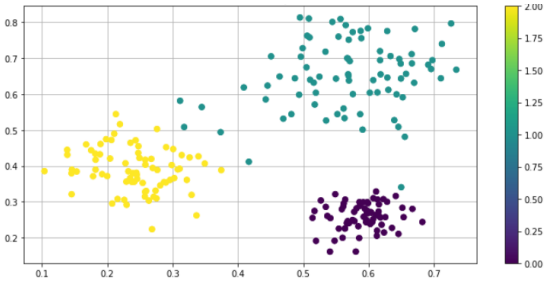

Figure 5.32: Federated GMM client 2 in non-IID dataset 1

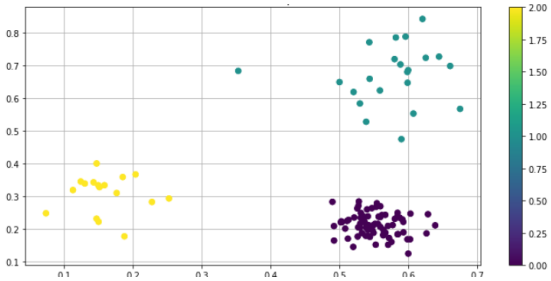

Figure 5.33: Federated GMM client 3 in non-IID dataset 1

In the case of the second non-IID dataset, the performance of the algorithm has been good with an score of 0.64 in silhouette. Furthermore, similar parameters has been reached with both metrics as validation, having an score of 0.987 with ARI.

In the case of the second non-IID dataset, the algorithm performed well, achieving a silhouette score of 0.64. Furthermore, similar parameter settings yielded a high validation score of 0.987 using the ARI.

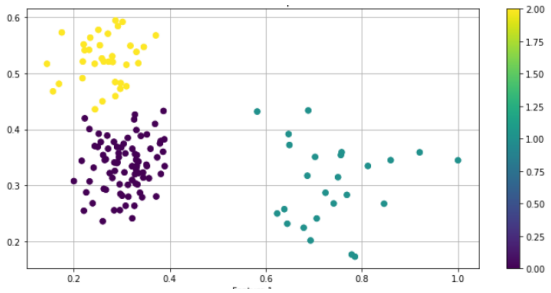

Figure 5.34: Federated GMM client 1 in non-IID dataset 2 with silhouette

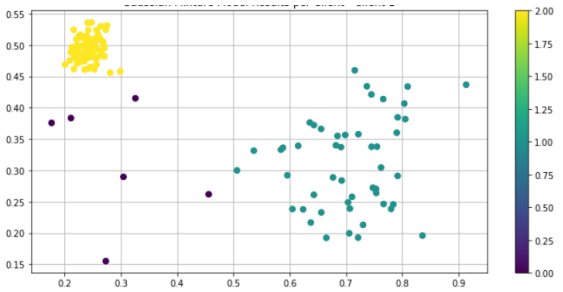

Figure 5.35: Federated GMM client 2 in non-IID dataset 2 with silhouette

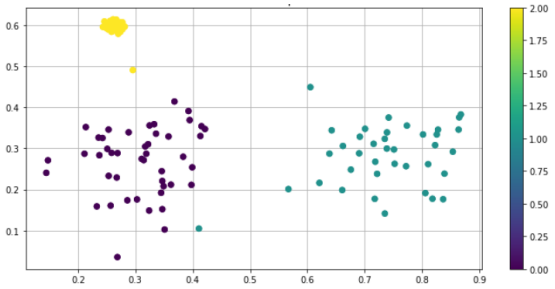

Figure 5.36: Federated GMM client 3 in non-IID dataset 3 with silhouette

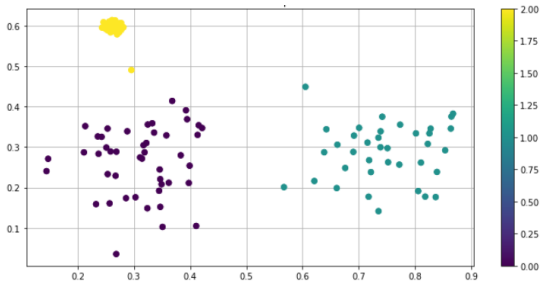

Figure 5.37: Federated GMM client 1 in non-IID dataset 2 with ARI

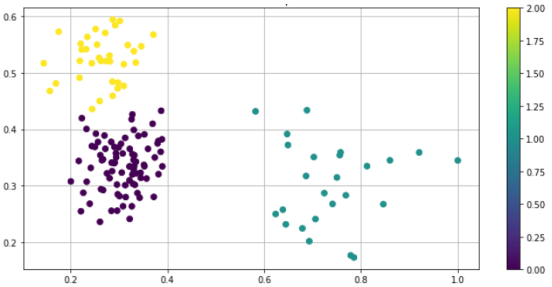

Figure 5.38: Federated GMM client 2 in non-IID dataset 2 with ARI

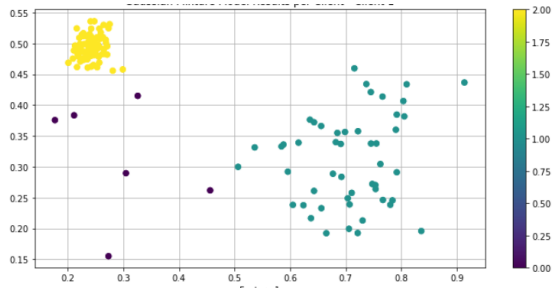

Figure 5.39: Federated GMM client 3 in non-IID dataset 3 with ARI

5.3.3 Analisys

The Federated Gaussian Mixture Model (GMM) proves to have an almost same performance as the traditional GMM approach, as shown by the comparative analysis of the Silhouette Score from different datasets, tables 5.10 and 5.9. Both federated and traditional frameworks are equally effective in their results, especially in datasets that have blob-like or linear structures where the data points cluster around Gaussian distributions. The same performance as GMM implies that the federated approach still has the statistical power of GMM to estimate the parameters of the Gaussian distributions even when the data are spread on different nodes without the central aggregation thereby the user privacy and the data security are still maintained.

Nevertheless, GMM has a difficulty to deal with non-linear data structures, a limitation that is based on its basic assumptions. GMM innately implies that each cluster can be described as a Gaussian distribution, which means a generally ellipsoidal shape. In these cases, the data are in the form of more complex shapes like intertwined spirals or crescent moons, and the assumption is not true. The expectation-maximization algorithm used to optimize the parameters of these Gaussian distributions may not converge thoroughly on non-linear configurations, Therefore, the clusters will be poorly identified and the clustering accuracy will be reduced. This restriction shows a significant problem area where the federated GMM, although it possesses the qualities of managing distributed data, resembles the traditional GMM in its inability to handle complex, non-linear data geometries.

5.4 Results Federated Spectral Clustering

This work has follow the implementation proposed in the work of Thakkar et al.[18], and also tried a self development following the article[8]. But was not possible to develop a fully functional version of Federated Spectral Clustering. The main reasons for the unsuccessful completion of the project of Federated Spectral Clustering are the complexities of

aggregating local spectral properties and the difficulties in the preservation of the order and the integrity of the eigenvectors across multiple nodes. These special challenges of keeping the global eigenstructure and aligning the dimensions of eigenvectors during aggregation have caused the huge deterioration of clustering performance and therefore the algorithm became useless in the federated learning environment. Hence, even though the attempts to make spectral clustering compatible with the federated settings, the special requirements of the process of aggregation still block the way for a practical Federated Spectral Clustering solution.

Problems appear when labels are created on a central server using aggregated spectral features, without the possibility of getting to individual data points' detailed characteristics. In the mentioned cases, the misalignment and distortion in the global eigenstructure caused by the bad aggregation process result in labels that do not accurately represent the real-life relationships and distributions of the data points across the various nodes. The server-generated labels are reapplied to the individual datasets, and the results are usually markedly misclassified, Hence, the clustering outcomes are largely inaccurate. This separation the labels derived from the compromised global model from the true essence of the local datasets. Therefore, the data's natural structure is distorted and the analysis later on based on these clusters is unreliable, making the entire federated clustering approach practically useless.

5.5 Comparison between algorithms

Dataset	**Federated K-means**	**Federated DBSCAN**	**Federated GMM**
Blobs	0.79	0.768	0.80
Moons	0.49	0.252	0.49
Circles	0.47	0.196	0.47

Table 5.13: Algorithm Results for Each Dataset first distribution

Dataset	Federated K-means	Federated DBSCAN	Federated GMM
Blobs	0.762	0.751	0.76
Moons	0.471	0.224	0.49
Circles	0.60	0.196	0.60

Table 5.14: Algorithm Results for Each Dataset second distribution

In the environment of federated clustering algorithms, the flexibility each algorithm shows is vital since the different data complexities are the reason for the varying data clustering. DBSCAN turns out to be the most versatile choice, which is capable of dealing with datasets that are characterized by non-linear shapes and different densities successfully. This flexibility is the key factor that makes it really good for the real-world data which is not necessarily uniform. The contrary is that GMM is flexible but its usefulness depends on the assumption of the statistical distributions of the data, which might be restrictive for the datasets that clearly conform to these assumptions. On the one hand, K-Means, even though it is less flexible, it is very simple and efficient, especially when the data has spherical clusters. It, therefore, is a perfect choice for datasets which have the clear and separable groups, but its usefulness decreases with the increasing complexity of data structures.

Dataset	Federated K-means	Federated DBSCAN	Federated GMM
non-IID Dataset	0.73	0.61	0.70

Table 5.15: Algorithm Results for non-IID Dataset 1

Metric	Federated K-means	Federated DBSCAN	Federated GMM
Silhouette	0.69	0.50	0.64
ARI	0.97	0.82	0.987

Table 5.16: Algorithm Results for non-IID Datasets

The performance of all algorithm is good when the clusters are well distributed across

the space. All of them being able to recognize the clusters without problems and having good scores how as reflected in the table 5.15.

However, Similar to the federated K-means algorithm when the labels are not used as an score and silhouette score is used as an score to tune the hyperparameters, the Federated DBSCAN algorithm has issues differentiating between clusters when encountering narrow, blurred cluster boundaries in a non-IID distribution. Variations are inherently present alongside the absence of clear boundaries; the latter complicates the definition of core points and the possibility to differentiate clusters within DBSCAN.

Unlike the previous models that have not been effective in handling non-IID data, the Federated GMM model has been able to achieve accurate recognition of the proper set of clusters in the given dataset using silhouette score, this gives it an advantage in situation where the real labels are unavailable.

Dataset	**Federated K-means**	**Federated DBSCAN**	**Federated GMM**
Complex	0.75	0.676	0.55

Table 5.17: Algorithm Results for Complex Dataset

Taking into account the effectiveness of these algorithms, their accuracy and application suitability can greatly affect their selection in the real life instances. Both K-Means and GMM usually obtain higher silhouette scores on structured and uncomplicated datasets, which means a greater level of cluster purity. On the other hand, DBSCAN, although it may have a lower silhouette score, has a lot of practical use in processing complex, irregularly shaped, or noisy datasets due to its ability to find out the hidden data structures. This trait is especially precious in federated settings where data is both massive and diverse and is scattered unevenly across different nodes. Hence, the selection of a federated clustering algorithm is to be done based on the type of the dataset. For the simpler, well-defined datasets, K-Means or GMM would be the best choice but for the more complex cases with uncertainties and noise, DBSCAN is the most suitable, hence the challenges that the data

presents will be overcome.

Chapter 6

Conclusions and future lines

Federated clustering is nowadays a very important tool in the data landscape of the modern world, in which the data is decentralized very often and spreads across many domains or nodes. The main benefit of this method is to the fact that it is able to deal with the data privacy and security issues in a very effective way. Through local processing of data at each node and the only sharing of insights or the aggregate information instead of raw data, Federated Clustering presents a solution that is in line with the strict legal regulations dealing with data privacy.

The reason why the federated clustering is so useful is not only the privacy, but it also has another advantage. It allows organizations to have a better understanding of data from a wider database than they could if data silos were kept, without affecting the privacy of the different sources of individual data. The need for decentralized data is especially great in the fields of healthcare, finance, and telecommunications, where the data is sensitive and cannot be centralized because of the reasons that are related to privacy issues or logistical constraints.

The analyses and discussions on the differences and similarities of the performance of federated clustering algorithms DBSCAN, GMM and K-Means clustering algorithm. DBSCAN is appropriate for irregular and noisy environments and as such it is very useful when

heterogeneity in a dataset is likely to be very high in real world applications. On the other hand, K-Means is very suitable for environments where clusters have more rounded shapes and their solutions are comparatively easier. As for GMM, it is suitable for analyzing data with a Gaussian distribution under a convex solution for an interaction between the modes. Due to this, the choice of a given algorithm is often based directly on the features of the available data and the peculiarities of a particular application.

There are numerous challenges associated with non-IID data in federated learning since the data is distributed across multiple devices and is heterogeneous and incoherent, making the local models of the participated devices hard to compile into a unified global model. However, by using appropriate models for the specific situations, these challenges can be effectively addressed.

Therefore, federated clustering is a perfect solution for the current situation of data privacy and the enhancement of the organizations ability to carry out data analysis on a large scale while keeping the data controlled. The data is expanding both in quantity and type, and the privacy issues are still the major issue, so federated clustering is a approach that will be used to protect the data and at the same time will maximize the use of the data. Hence, it is a key methodology that is widely used by the data scientists and analysts of today who are dealing with the complex data ecosystems.

6.1 Future Lines

As future line for this project, one of the most promising areas of focus is the testing of these algorithms on real non-iid data using actual devices. Testing federated clustering under these conditions is a must since it can mimic the numerous challenges in the real existence where data is derived from multiple sources with dissimilar features. This step will play a key role in ruling out theoretical models and making the algorithms capable of handling real data diversity and complexity.

In addition to doing research and experimenting with more complex algorithms, another considerable path towards improvement in federated learning is checking different cluster algorithms. Among the current topics that are have been researched on this project are DBSCAN, GMM, Spectral Clustering and K-Means. But there may be many other techniques which could prove to be advantageous with respect to some specific situations.For example, algorithms designed for a wider scalability, those that can adapt to dynamic environments, or those that solve problems related to sparse data could address gaps left by current methods. Not only will the broadening of the diversity of the algorithms tested help to enhance their robustness and adaptability, but, also, it will go a long way to increasing their applicability beyond different sectors and data challenges.

With a broader team that includes specialists in telecommunications, another key area of future research in federated clustering is the communication problems between client-server. Having an efficient communication between the distributed nodes and the central server is very essential for the practical implementation of federated learning algorithms. Limitations of bandwidth, latencies varying, and privacy of data issues may require innovative solutions to allow for efficient data synchronization without consuming excessive resources or compromising user privacy. It will be crucial to come up with light communication protocols or incorporate new technologies like differential privacy or data compression to solve these issues and increase the efficiency and capacity of the federated learning systems.

Bibliography

[1] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM Comput. Surv.*, vol. 31, p. 264–323, sep 1999.

[2] B. Everitt, "Cluster analysis," *Quality and Quantity*, vol. 14, no. 1, pp. 75–100, 1980.

[3] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*. San Francisco, CA, USA: Morgan Kaufmann, 3 ed., 2011.

[4] A. Jain, R. Duin, and J. Mao, "Statistical pattern recognition: A review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, pp. 4–37, 01 2000.

[5] B. Jiang, J. Pei, Y. Tao, and X. Lin, "Clustering uncertain data based on probability distribution similarity," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 4, pp. 751–763, 2013.

[6] Pew Research Center, "Smartphone ownership and internet usage continues to climb in emerging economies," 2016.

[7] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Federated learning of deep networks using model averaging," *CoRR*, vol. abs/1602.05629, 2016.

[8] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran, "An efficient framework for clustered federated learning," in *Advances in Neural Information Processing Systems* (H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds.), vol. 33, pp. 19586–19597, Curran Associates, Inc., 2020.

[9] X. Ma, J. Zhu, Z. Lin, S. Chen, and Y. Qin, "A state-of-the-art survey on solving non-iid data in federated learning," *Future Generation Computer Systems*, vol. 135, pp. 244–258, 2022.

[10] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. A. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. L. D'Oliveira, S. E. Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konečný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, M. Raykova, H. Qi, D. Ramage, R. Raskar, D. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, and S. Zhao, "Advances and open problems in federated learning," *CoRR*, vol. abs/1912.04977, 2019.

[11] A. M. Ikotun, A. E. Ezugwu, L. Abualigah, B. Abuhaija, and J. Heming, "K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data," *Information Sciences*, vol. 622, pp. 178–210, 2023.

[12] H. H. Kumar, K. V R, and M. K. Nair, "Federated k-means clustering: A novel edge ai based approach for privacy preservation," pp. 52–56, 2020.

[13] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," p. 226–231, 1996.

[14] CodiceSpaghetti, "Federateddbscan." `https://github.com/codiceSpaghetti/FederatedDBSCAN/tree/main`, 2024. Accessed: 14/05/2024.

[15] D. A. Reynolds *et al.*, "Gaussian mixture models.," *Encyclopedia of biometrics*, vol. 741, no. 659-663, 2009.

[16] M.-S. Yang, C.-Y. Lai, and C.-Y. Lin, "A robust em clustering algorithm for gaussian mixture models," *Pattern Recognition*, vol. 45, no. 11, pp. 3950–3961, 2012.

[17] A. Ng, M. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," vol. 14, 2001.

[18] J. Thakkar and D. Joshi, "Fedspectral+: Spectral clustering using federated learning," 02 2023.

[19] E. Rendón, I. Abundez, A. Arizmendi, and E. Quiroz, "Internal versus external cluster validation indexes," *International Journal of Computers and Communications*, vol. 5, pp. 27–34, 01 2011.

[20] M. K.-P. N. Yalchin Efendiev, Taketomo Mitsui and F. Tank, *Journal of Computational and Applied Mathematics*. Elsevier, 1975.

[21] L. Hubert and P. Arabie, "Comparing partitions," *Journal of Classification*, vol. 2, no. 1, pp. 193–218, 1985.