IE494 Big Data Processing

Final Project Report

Ishita Rathod (202101516)

Bansri Patel (202101097)

(Git repository: Link)

1. Problem and Solution

1.1 Introduction to Clustering in Big Data

Clustering is one of the most important techniques in data analysis and machine learning, especially in scenarios where labelled data is not available. The goal of clustering is simple yet powerful: to group similar data points into clusters based on their features, thereby uncovering hidden patterns in data. For instance:

- In business, clustering can help identify customer segments for targeted marketing.
- In healthcare, clustering can group patients based on medical profiles for better diagnosis and treatment strategies.
- In ecology, clustering can analyze environmental data to identify regions with similar climatic conditions.

Clustering is widely used in many fields, but it often runs into challenges when dealing with large-scale datasets. In today's world of big data, datasets aren't just huge in size—they're also complex, with high dimensions and diverse features. This complexity demands algorithms that are both robust and scalable.

1.2 Challenges with K-Means

The **k-means algorithm** is one of the most popular clustering methods due to its simplicity and efficiency for small to medium-sized datasets. However, k-means has several limitations when dealing with large or complex datasets:

1.2.1 Sensitivity to Outliers

K-means represents each cluster using a centroid, which is the average of all points in the cluster. While this works well for clean datasets, it becomes problematic when outliers are present. For example, in customer segmentation, a single customer with an unusually high spending pattern can disproportionately shift the centroid, which eventually leads to inaccurate clustering.

1.2.2 Assumption of Spherical Clusters

K-means assumes that all clusters are spherical and have similar sizes. This is rarely the case in real-world data, where clusters can be irregularly shaped, elongated, or vary significantly in density. This challenge becomes even bigger with big data, where the "variety"—one of the 3 V's of big data—brings in a mix of feature types and complicated structures. As a result, k-means often fails to capture the true structure of the data.

1.2.3 Scalability Issues

When applied to large datasets, the sequential implementation of k-means becomes computationally expensive. Each iteration of the algorithm requires:

1. Computing the distance between each data point and every centroid.
2. Updating centroids based on the assigned data points.

These steps are repeated until convergence, resulting in a time complexity of $O(n \cdot K \cdot d \cdot I)$, where:

- n : Number of data points.
- K : Number of clusters.
- d : Number of dimensions (features) in the data.
- I : Number of iterations.

For big data, where *n, d and I* can be very large, the computational cost becomes a significant bottleneck. Additionally, k-means requires loading the entire dataset into memory, which is impractical for datasets that exceed the system’s RAM.

1.3 K-Medoids

To address some of these limitations, **k-medoid** was introduced as an alternative clustering method. K-medoids is similar to k-means but differ in one key aspect: instead of using centroids (averages), it selects **medoids**, which are actual data points, as cluster representatives. This fundamental difference makes k-medoids:

- **More Robust to Outliers**: Medoids, as they are real data points, they are less influenced by extreme values compared to centroids.
- **Better for Irregular Clusters**: K-medoids does not assume spherical clusters, which allows it to handle diverse cluster shapes effectively.

However, k-medoids is computationally more expensive than k-means because evaluating potential medoids requires calculating the total distance between all data points and their respective medoid candidates. This cost grows significantly for large datasets.

1.4 The Need for Parallelization

Clustering algorithms like k-means and k-medoids operate through iterative processes, repeatedly calculating distances and updating clusters. While k-means performs well with small datasets, its limitations become evident as datasets grow larger and more complex. The computational cost rises significantly because the algorithm needs to calculate distances for every data point and cluster representative during each iteration. Memory usage also becomes a bottleneck, as traditional implementations require the entire dataset to fit into memory, which is not feasible for massive datasets. Additionally, these algorithms face challenges when dealing with sparse or highdimensional data, as dense vector calculations and centroid updates can slow down performance, particularly in the case of k-means.

Parallelization offers a way to address these challenges by distributing computations across multiple processing units. In both k-means and k-medoids, the two main operations—distance computations and centroid/medoid updates—are well-suited for parallel execution. By dividing the dataset into smaller partitions, multiple processing nodes can compute distances and assign clusters simultaneously.

- **Reduce Execution Time**: By dividing the dataset into partitions, multiple nodes can compute distances and assign clusters simultaneously.
- **Handle Large Datasets**: Distributed systems can process datasets that exceed the memory capacity of a single machine.
- **Improve Scalability**: Parallel implementations ensure that the algorithms scale well with increasing data size and computational resources.

1.5 Spark: A Platform for Big Data Clustering

To implement parallel k-medoids, we leverage **Apache Spark**, a distributed computing framework designed for big data processing. Spark's key features include:

- **Resilient Distributed Datasets (RDDs)**: RDDs allow data to be distributed across multiple nodes in a cluster, enabling parallel computations while ensuring fault tolerance.
- **In-Memory Processing**: Spark stores intermediate results in memory, reducing the overhead of repeatedly reading and writing to disk.
- **Scalable Architecture**: Spark can handle datasets ranging from gigabytes to petabytes by adding more nodes to the cluster.

By using Spark, we can parallelize the core steps clustering algorithms, including distance computations and centroid/medoid updates, to achieve scalability and efficiency.

1.6 Objectives of This Project

This project focuses on implementing and evaluating parallelized versions of both **k-means** and **k-medoids** clustering algorithms using Apache Spark.

1. **To Parallelize K-Means and K-Medoids**:
    - Implement the core iterative processes (distance computations and cluster updates) for k-means and k-medoids on Spark.
2. **To Compare Sequential and Parallel Implementations**:
    - Evaluate the performance of sequential and parallel versions of k-means and k-medoids in terms of execution time and clustering quality.
3. **To Analyze Two Datasets**:
    - Country Dataset
    - Iris Dataset

4. **To Highlight the Trade-Offs Between K-Means and K-Medoids**:
   - Demonstrate the robustness of k-medoids for irregular clusters and outlier handling.
   - Analyze the efficiency of k-means for datasets with spherical and well-separated clusters.

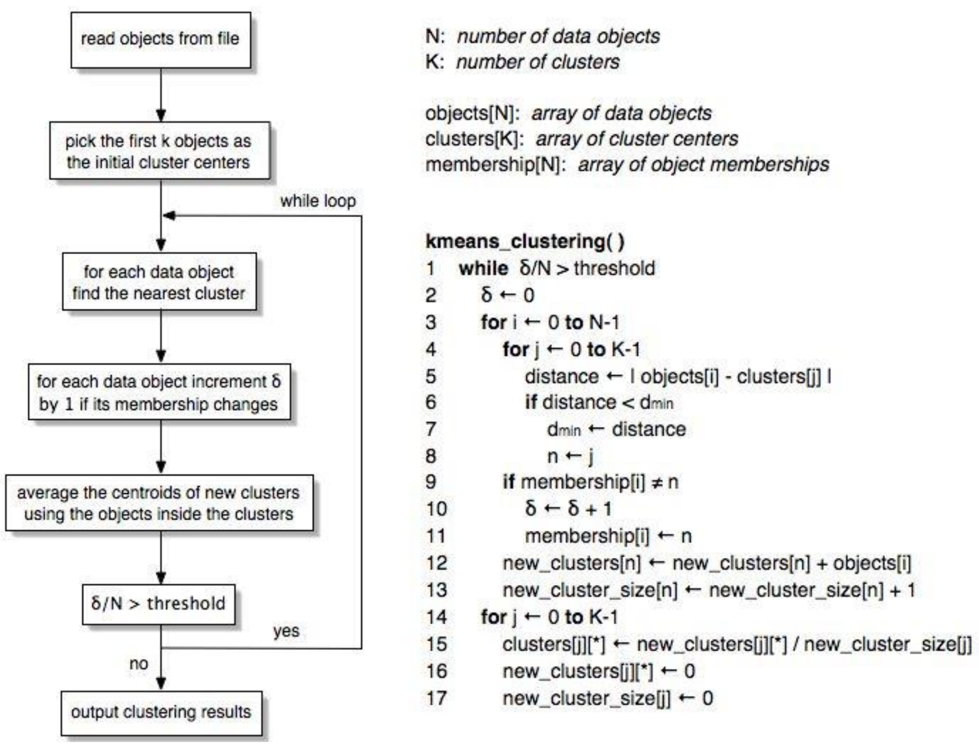
By achieving these objectives, this project aims to highlight the practical advantages of parallel k-medoids for big data applications.

2. Approach/Algorithm Implemented

This section details the k-means and k-medoid clustering algorithms, their key steps, and the methodology for parallelizing these algorithms using Apache Spark.

2.1 K-Means Clustering Algorithm

The **k-means algorithm** aims to partition a dataset into *k* clusters. This algorithm involves iterative refinement. Each iteration consists of two key steps: assigning data points to the nearest cluster centroid and recalculating centroids based on the assigned points. This continues until the centroids stabilize.


read objects from file
pick the first k objects as the initial cluster centers
while loop
for each data object find the nearest cluster
for each data object increment δ by 1 if its membership changes
average the centroids of new clusters using the objects inside the clusters
δ/N > threshold
yes
no
output clustering results

N: *number of data objects*
K: *number of clusters*

objects[N]: *array of data objects*
clusters[K]: *array of cluster centers*
membership[N]: *array of object memberships*

```
kmeans_clustering( )
while  δ/N > threshold
    δ ← 0
    for i ← 0 to N-1
        for j ← 0 to K-1
            distance ← | objects[i] - clusters[j] |
            if distance < d_min
                d_min ← distance
                n ← j
        if membership[i] ≠ n
            δ ← δ + 1
            membership[i] ← n
        new_clusters[n] ← new_clusters[n] + objects[i]
        new_cluster_size[n] ← new_cluster_size[n] + 1
    for j ← 0 to K-1
        clusters[j][*] ← new_clusters[j][*] / new_cluster_size[j]
        new_clusters[j][*] ← 0
        new_cluster_size[j] ← 0
```

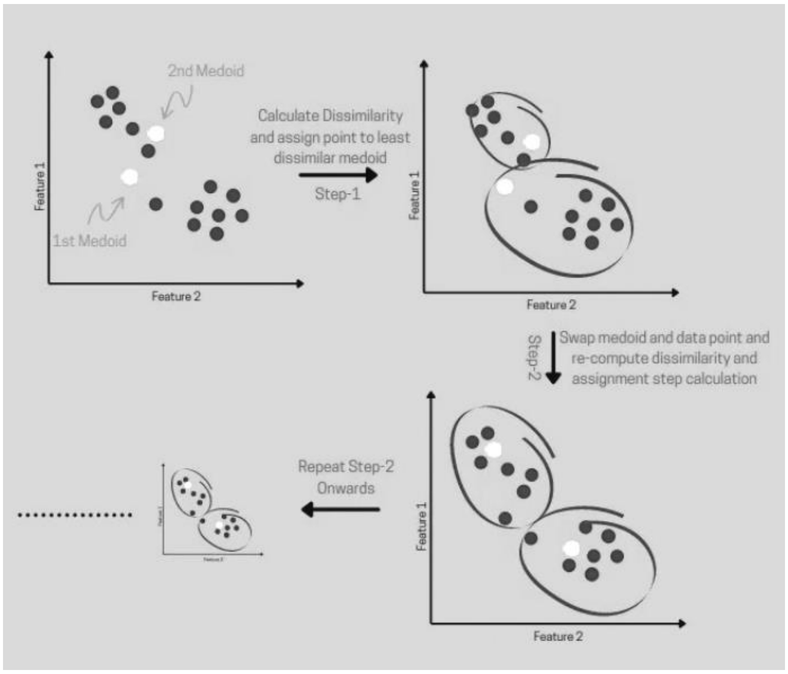
Source: Link

The algorithm works iteratively:

1. **Initialization**: Randomly select *k* data points as centroids.
2. **Cluster Assignment**: Assign each data point to the nearest centroid based on a distance metric (e.g., Euclidean distance).
3. **Centroid Update**: Compute the mean of points in each cluster and update the centroid.
4. **Convergence Check**: Repeat steps 2 and 3 until centroids stabilize or a maximum number of iterations is reached.

2.2 K-Medoids Clustering

The **k-medoids algorithm** improves upon k-means by using actual data points (medoids) as cluster representatives instead of centroids. This makes k-medoids more robust to outliers and effective for clusters with non-spherical shapes. However, it is computationally more intensive due to its iterative process of swapping medoids to minimize total clustering cost.


2nd Medoid
1st Medoid
Feature 1
Feature 2
Calculate Dissimilarity and assign point to least dissimilar medoid
Step-1
Step-2
Swap medoid and data point and re-compute dissimilarity and assignment step calculation
Repeat Step-2 Onwards

Source: Link

Steps of the K-Medoids Algorithm

1. **Initialization -** Randomly select *k* points from the dataset as the initial medoids.
2. **Cluster Assignment -** Calculate the distance of each data point to all medoids and assign each point to the cluster represented by the nearest medoid.
3. **Cost Calculation -** Compute the total cost as the sum of distances between each data point and its assigned medoid.

4. **Medoid Optimization -** Randomly select a new point as a potential medoid and reassign clusters and recalculate the total cost. If the new cost is lower, accept the change and use the new medoid, If the cost is higher, revert to the previous medoid and repeat this step.
5. **Convergence -** Continue swapping medoids and optimizing clusters until no further changes occur till convergence.

3. Testing: Dataset, Results, and Analysis

After implementing and parallelizing the k-means and k-medoids algorithms using Apache Spark, the next step was to test their performance on real datasets. For this, we used two datasets: the Country Dataset and the Iris Dataset, which are well-suited for evaluating clustering algorithms.

3.1 Dataset Details

Iris Dataset

The Iris Dataset is a classic benchmark dataset often used to evaluate clustering algorithms. It contains:

- **Features**: Sepal length, sepal width, petal length, and petal width.
- **Classes**: Three distinct species of Iris flowers: Setosa, Versicolor, and Virginica.

Although the Iris dataset is relatively small and well-structured, it serves as a good test case for verifying the clustering algorithm's ability to classify between well-defined groups.

Country Dataset

The Country Dataset contains socio-economic and health-related indicators for various countries. These features include:

- **Child Mortality**: Number of deaths per 1,000 live births.
- **Exports**: Percentage of GDP contributed by exports.
- **Health Expenditure**: Percentage of GDP spent on healthcare.
- **Income**: Average income per person.
- **Inflation**: Annual percentage increase in prices.
- **Life Expectancy**: Average lifespan of individuals in years.
- **GDP**: Total economic output of a country.

These attributes provide diverse set of features for clustering. The goal here is to group countries into clusters that reflect similarities in their socio-economic and health conditions.

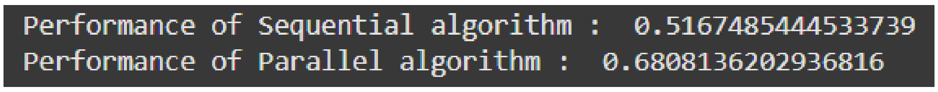
3.2 Results

To evaluate the performance of the clustering algorithms, we used the below key metric:

Silhouette Score: This score quantifies the quality of clustering by measuring how similar a data point is to its assigned cluster compared to other clusters. Higher scores indicate better-defined and more meaningful clusters.

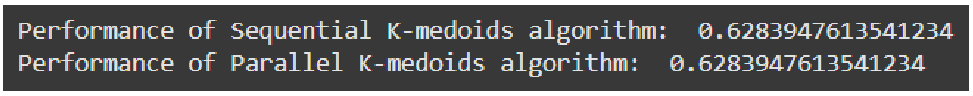
1. **Results for the Iris Dataset**

   - Comparison of the Silhouette Scores between sequential and parallel K-Means Clustering algorithm:



```
Performance of Sequential algorithm :  0.5167485444533739
Performance of Parallel algorithm :  0.6808136202936816
```

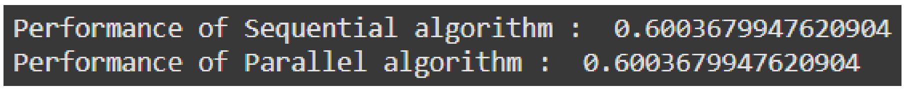
   - Comparison of the Silhouette Scores between sequential and parallel K-Medoids Clustering algorithm:



```
Performance of Sequential K-medoids algorithm:  0.6283947613541234
Performance of Parallel K-medoids algorithm:  0.6283947613541234
```

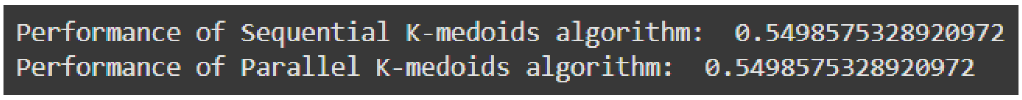
2. **Results for the Country Dataset**

   - Comparison of the Silhouette Scores between sequential and parallel K-Means Clustering algorithm:



```
Performance of Sequential algorithm :  0.6003679947620904
Performance of Parallel algorithm :  0.6003679947620904
```

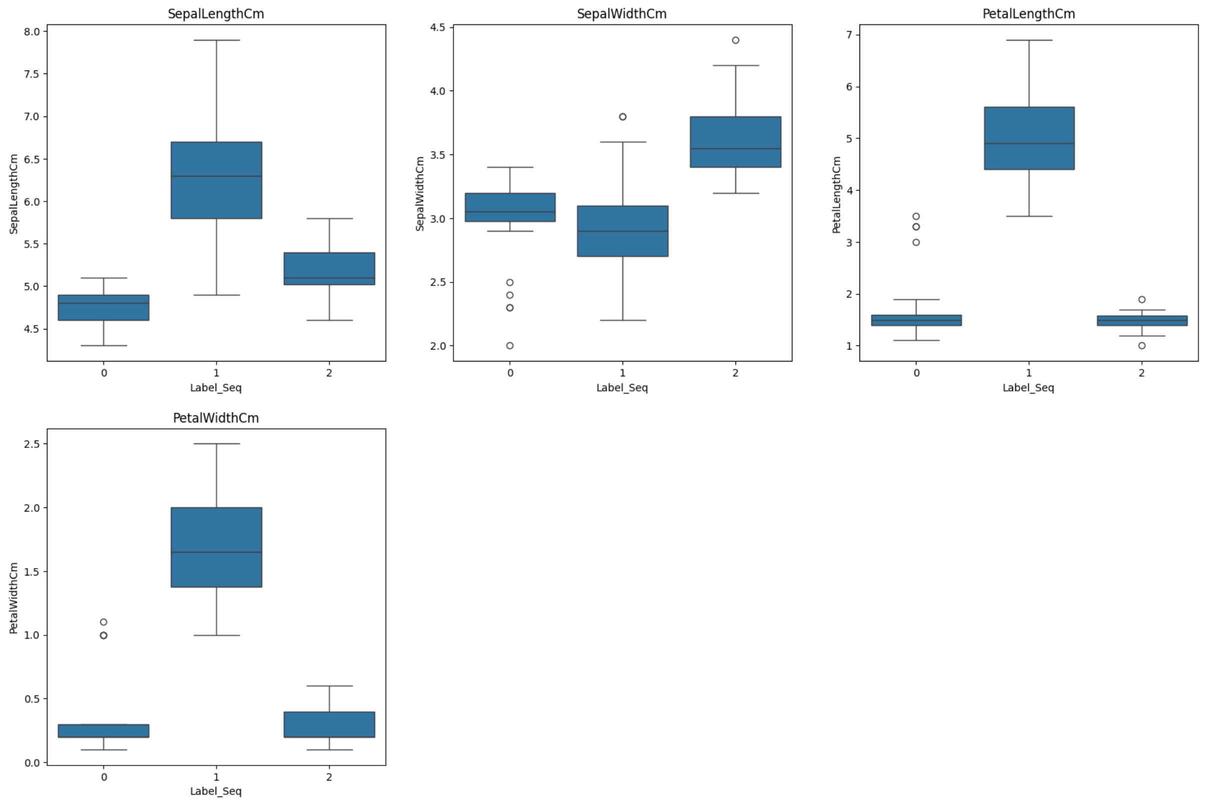
   - Comparison of the Silhouette Scores between sequential and parallel K-Medoids Clustering algorithm:

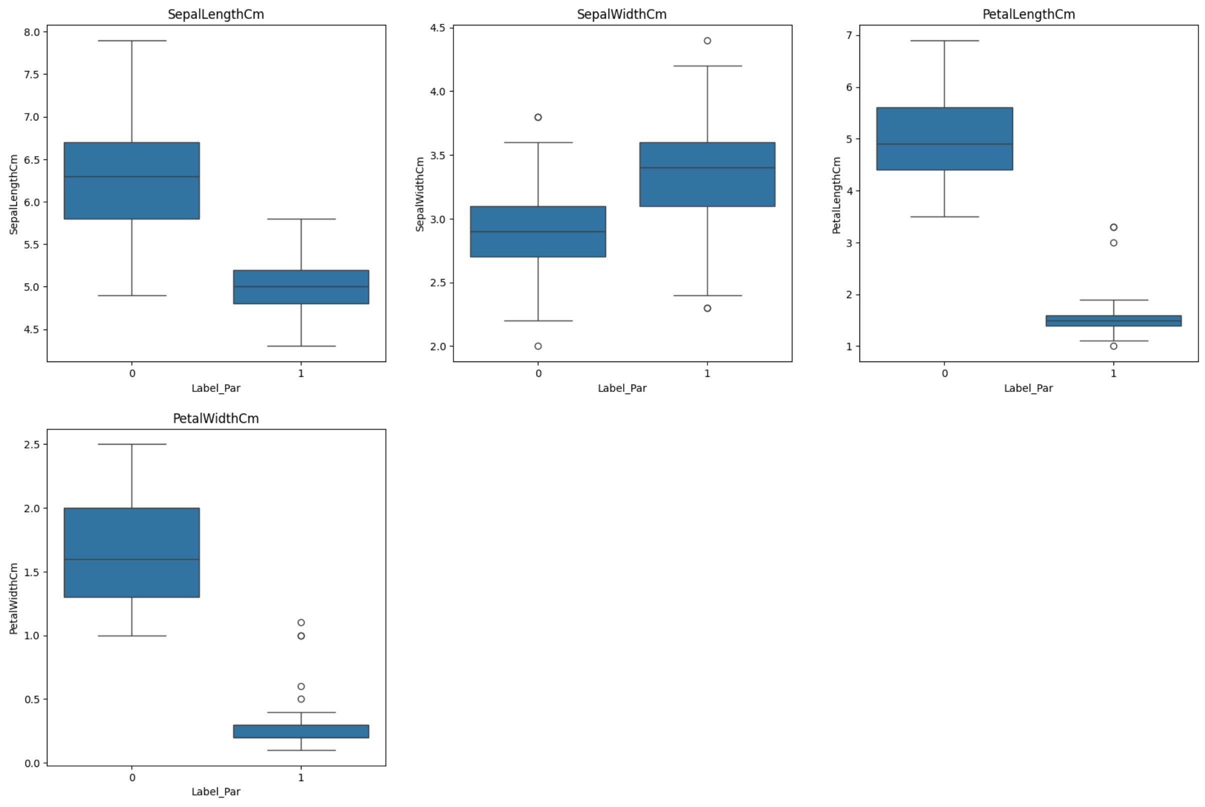


```
Performance of Sequential K-medoids algorithm:  0.5498575328920972
Performance of Parallel K-medoids algorithm:  0.5498575328920972
```

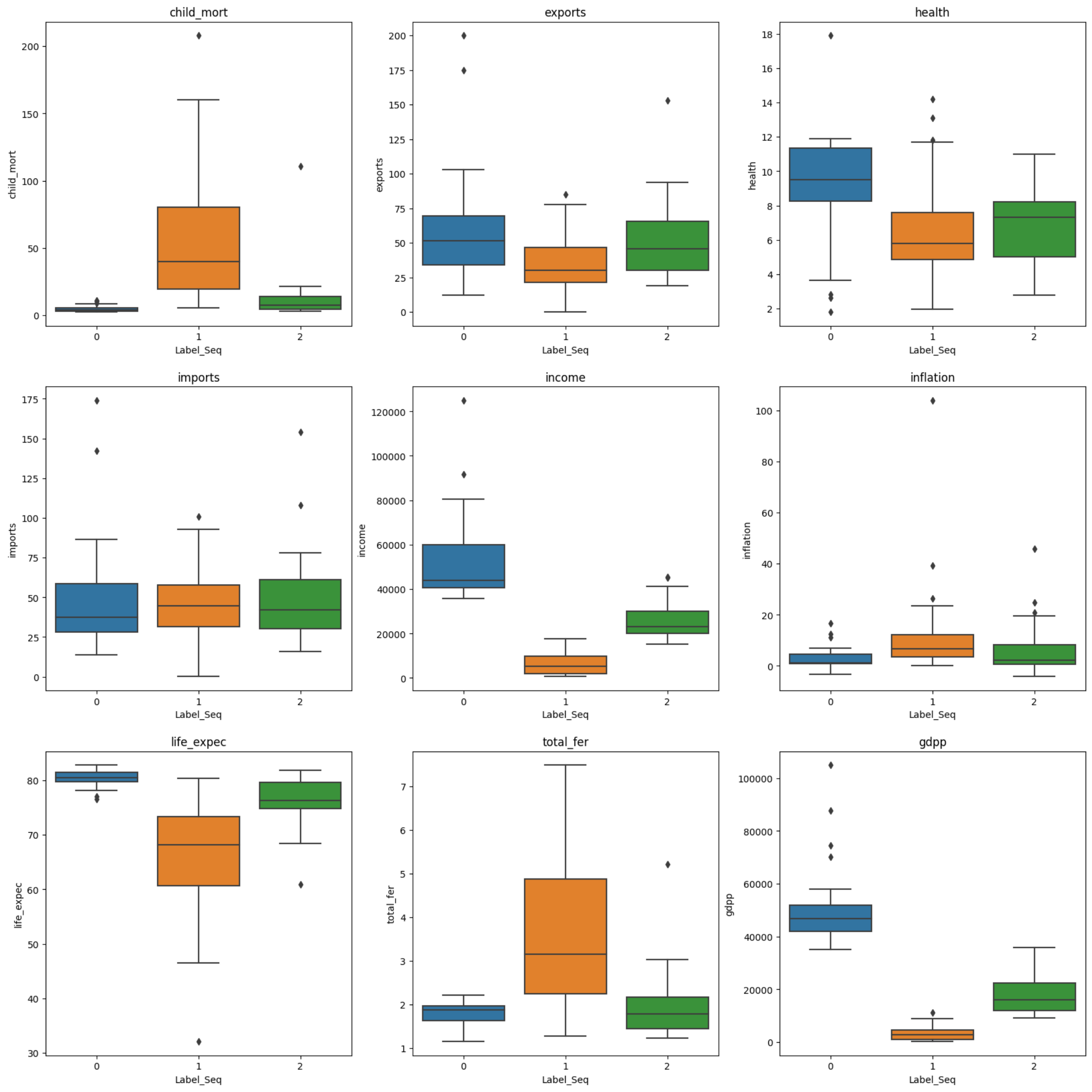
Results for k-means clustering:

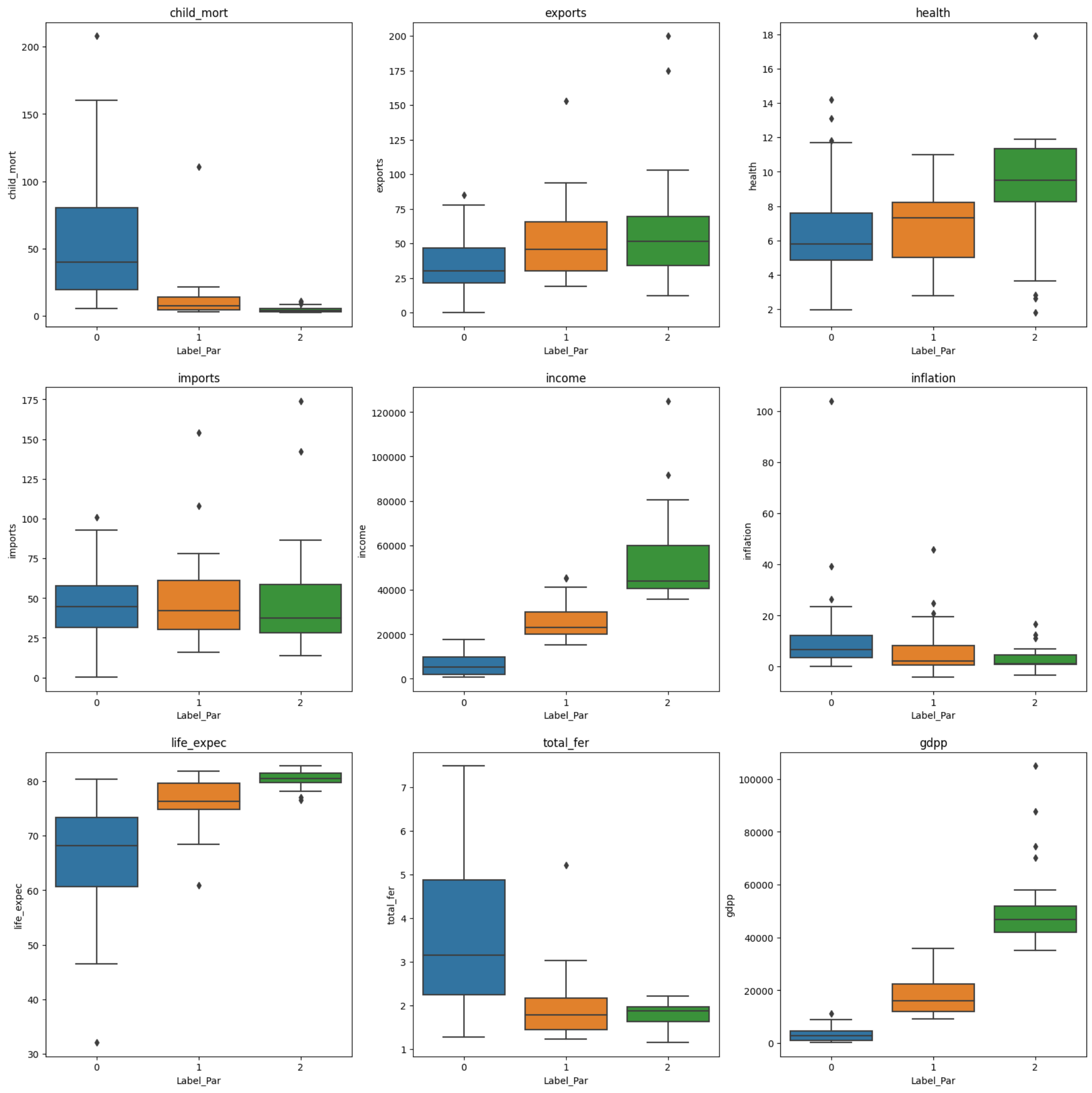
Iris dataset:





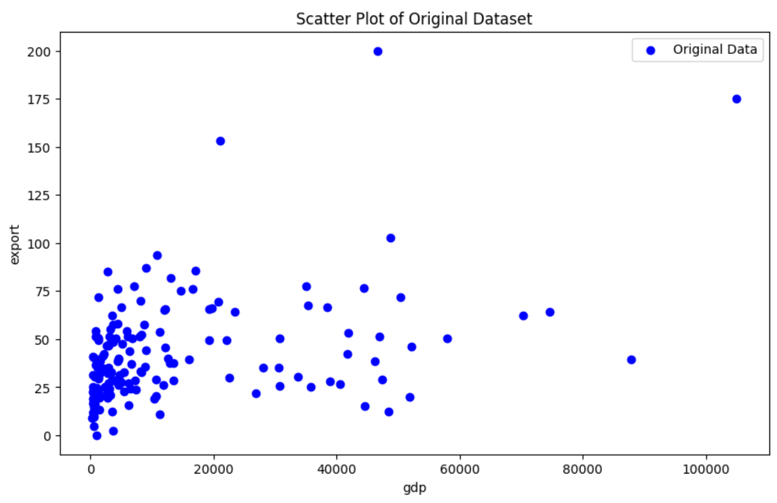
Country dataset:

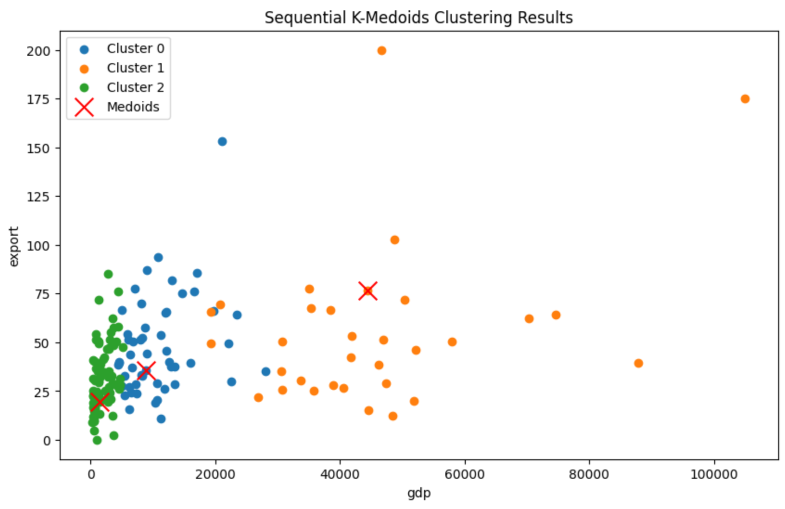


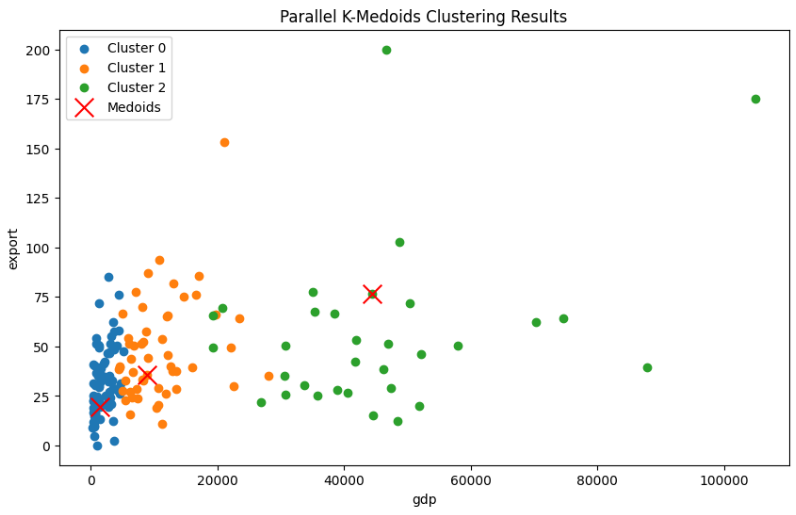

child_mort
exports
health
imports
income
inflation
life_expec
total_fer
gdpp
Label_Par
0
1
2

Results for k-medoids clustering:

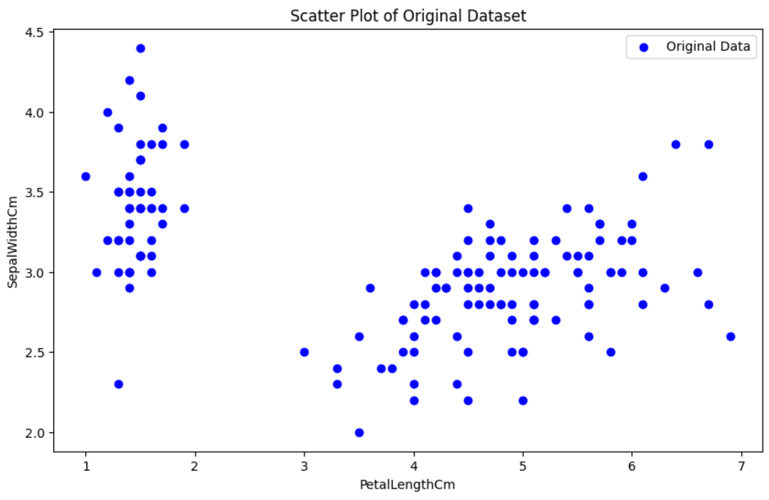
Country dataset:

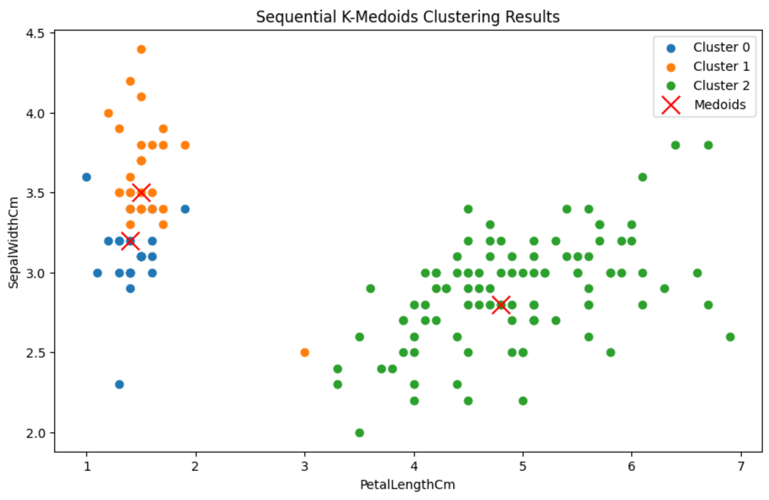


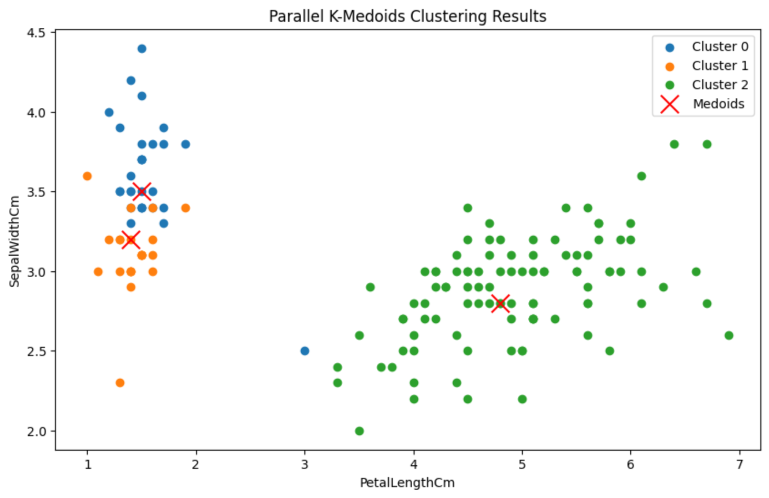




Iris dataset:







References:

[1] Wang, Bowen, et al. "Parallelizing k-means-based clustering on spark." 2016 International Conference on Advanced Cloud and Big Data (CBD). IEEE, 2016.