Final Report

Name	Student ID
Dharva Patel	202001135
Het Patel	202001434
Gaurang Parmar	202001444

Github: Link

Abstract

In this report we are going to discuss the two clustering algorithms: k-means & k-medoids. Problem with their sequential implementation and parallel implementation as solution.

Problem and Solution:

Conventional data analysis methods face difficulties in the big data era due to the growing volume and complexity of data. To successfully handle enormous datasets and extract useful insights, specialized algorithms capable of scalable and distributed processing are required.

The clustering algorithm came into play here. Clustering refers to the process of grouping similar data points or objects into distinct subsets, known as clusters, based on certain features or characteristics. The key aspects of clustering are similarity measurement, grouping similar entities, scalability and many more. K-means clustering is a kind of clustering algorithm. However, there's an issue with sequential implementation.

Problem with implementation of sequential K-means clustering algorithm:

Sequential implementation of the clustering k-means algorithm has various difficulties, particularly when working with huge datasets, leading to the creation of parallel k-means clustering techniques. Here are a few issues regarding sequential k-means clustering algorithm:

Computational complexity:

The computational complexity of the sequential algorithm is:

O(n * K * d * I)

where,

n : total points

K : total number of clusters

d : number of attributes

I : iterations count

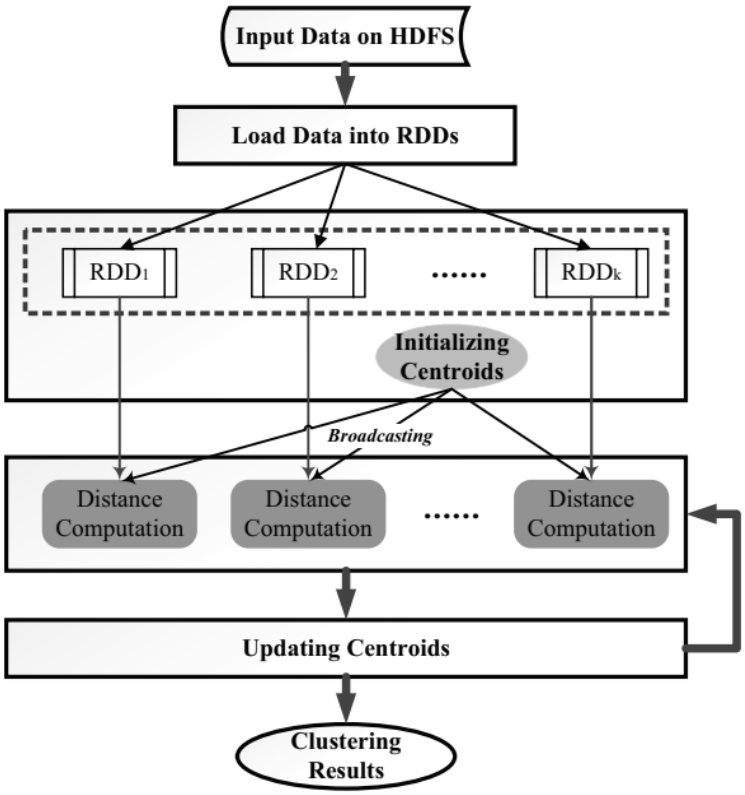
Challenges which arise from the iterative nature of k-means, where each iteration involves potentially extensive distance calculations and centroid updates. In the context of big data, where n, K, d and I can be large, these complexities highlight

the need for parallel and distributed implementations to enhance computational efficiency.

Memory Requirements:
Sequential k-means algorithm requires presence of complete dataset in memory. This might result in memory limits for huge datasets, especially when dealing with big data, where the dataset size exceeds the available RAM.

Inefficiency with Large Datasets:
The computing cost of distance calculations and centroid updates rises linearly as the dataset size (total data points(n)) grows. This can result in significant processing time for large datasets.

Implementing parallel versions of the clustering k-means algorithm is a key solution. Parallel approaches distribute the workload across multiple processing units, enabling concurrent updates to cluster assignments and centroids. This is particularly beneficial for handling large datasets and improving the algorithm's scalability.



But there are some limitations in the k-means clustering algorithm as below.

Spherical Shaped Clusters:
K-means clustering assumes that clusters are spherical and equally sized, as it uses the mean (centroid) to represent the cluster. This means that if the true clusters have a different shape (e.g., elongated or irregular), K-means may not perform well. It's not suitable for identifying clusters with complex shapes or clusters that vary in size and density.

Sensitivity to Outliers:
K-means is sensitive to outliers or data points that are significantly different from the majority of the data. Outliers can disproportionately affect the mean calculation during the centroid update step, leading to skewed and flawed cluster assignments.

This can be addressed by the k-medoids clustering algorithm by offering robustness to outliers. Its choice of medoids as representatives enhances its suitability for scenarios where outliers and irregular cluster shapes can pose challenges to traditional k-means clustering.

The problems in the sequential k-medoids clustering algorithm can also be addressed by parallelizing it.

The key difference between clustering k-means and k-medoids algorithm:
Centroid Representation:
K-means: Centroids in K-means are represented as the mean (average) of data points in the cluster.
K-medoids: Medoids in K-medoids are represented by the actual data points themselves, minimizing the sum of dissimilarities (often measured using distance metrics) to other points in the cluster.

Sensitivity to Outliers:
K-means: Sensitive to outliers because it minimizes the sum of squared distances.
K-medoids: More robust to outliers since it uses medians (or other representative points) that are less influenced by extreme values.

Assignment of Points to Clusters:

K-means: The cluster with the closest mean is given points.

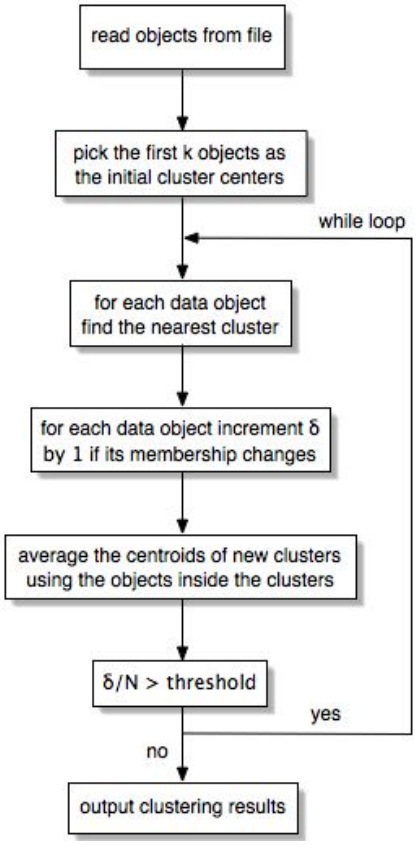
K-medoids: Assigns points to the cluster having closest medoid, where the medoid is the data point that minimizes the sum of dissimilarities to other points in the cluster.

Algorithm Convergence:

K-means: Converges when the centroids no longer change significantly.

K-medoids: Converges when the medoids no longer change significantly.

Algorithm Implemented:



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

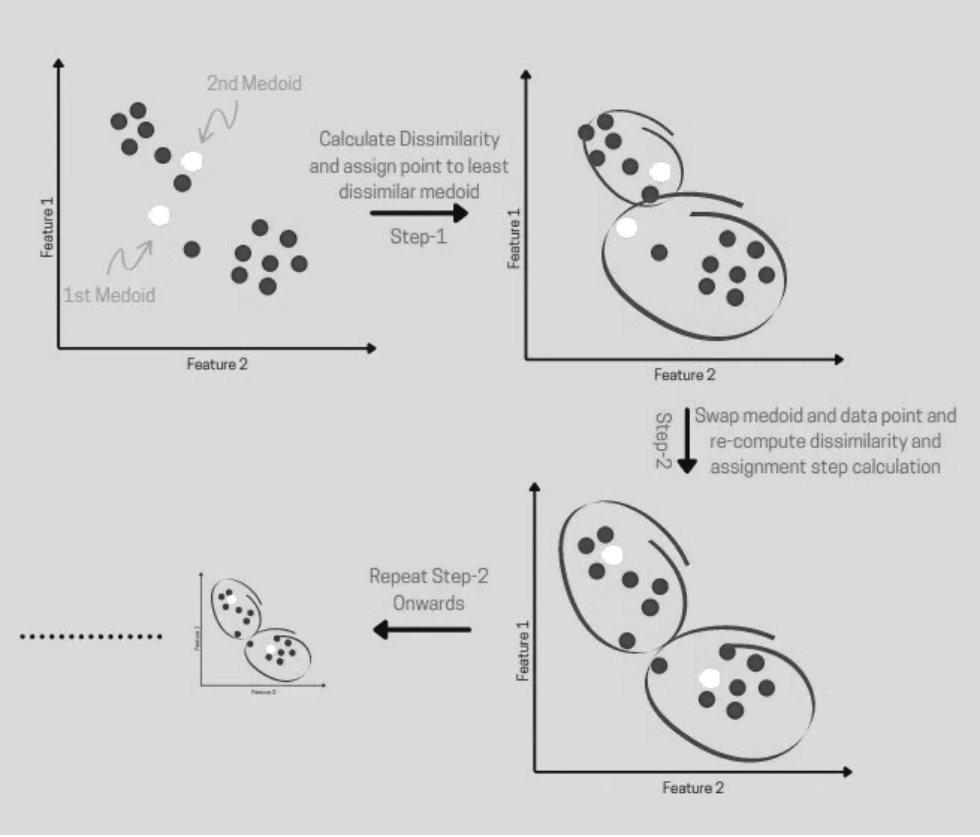
objects[N]: *array of data objects*
clusters[K]: *array of cluster centers*
membership[N]: *array of object memberships*

```
kmeans_clustering( )
1   while δ/N > threshold
2     δ ← 0
3     for i ← 0 to N-1
4       for j ← 0 to K-1
5         distance ← | objects[i] - clusters[j] |
6         if distance < dmin
7           dmin ← distance
8           n ← j
9       if membership[i] ≠ n
10        δ ← δ + 1
11        membership[i] ← n
12      new_clusters[n] ← new_clusters[n] + objects[i]
13      new_cluster_size[n] ← new_cluster_size[n] + 1
14    for j ← 0 to K-1
15      clusters[j][*] ← new_clusters[j][*] / new_cluster_size[j]
16      new_clusters[j][*] ← 0
17      new_cluster_size[j] ← 0
```

[Reference : Link]

Kmedoids_clustering():

1. Begin by randomly selecting k points from the dataset, designating them as the initial medoids.
2. For the remaining data points, determine the distance to each medoid and assign each point to the cluster associated with the closest medoid.
3. Determine the overall cost by adding up the distances between each data point and its corresponding medoids.
4. Select a random point as a new medoid, replacing the previous medoid, and repeat steps 2 and 3.
5. If the total cost with the new medoid is lower than that of the previous medoid, accept the change and proceed to step 4.
6. If the total cost with the new medoid is higher, undo the swap and repeat step 4.
7. Continue these iterations until no further changes occur with the new medoids, effectively classifying data points into distinct clusters.



[Reference: Link]

Testing:

Dataset :

The dataset contains information on various attributes for different countries. These attributes include child mortality, exports, health expenditure, imports, income, inflation, life expectancy, total fertility rate, and GDP. These features provide insights into a country's socio-economic and health indicators, such as the number of child deaths, economic activity, healthcare investment, average income, inflation rate, life expectancy, population growth, and overall economic performance. The dataset can be utilized for clustering analysis, such as K-means and K-medoids clustering, to identify similarities and patterns among countries based on these attributes.

Results :

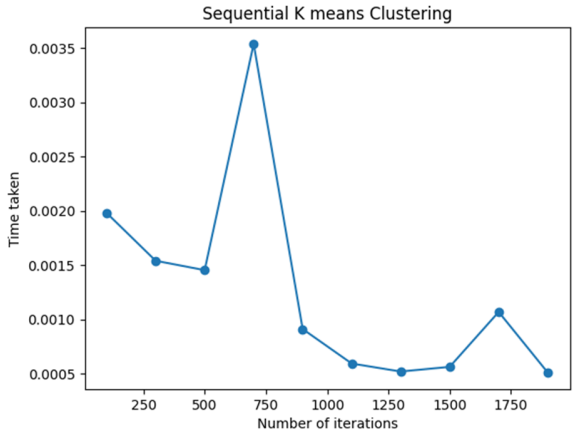
Here is the comparison of Silhouette Score between sequential and parallel K-Means Clustering algorithm:

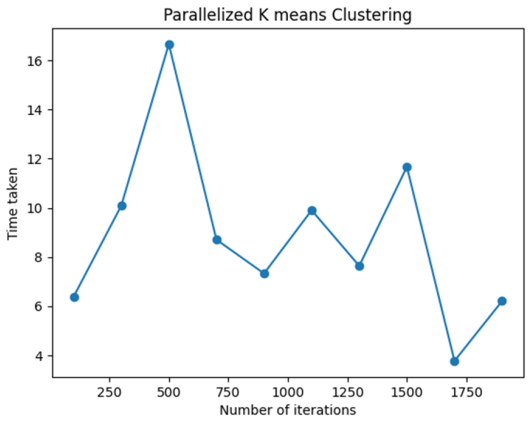
```
Performance of Sequential algorithm :  0.6003679947620904
Performance of Parallel algorithm :  0.6003689044610714
```

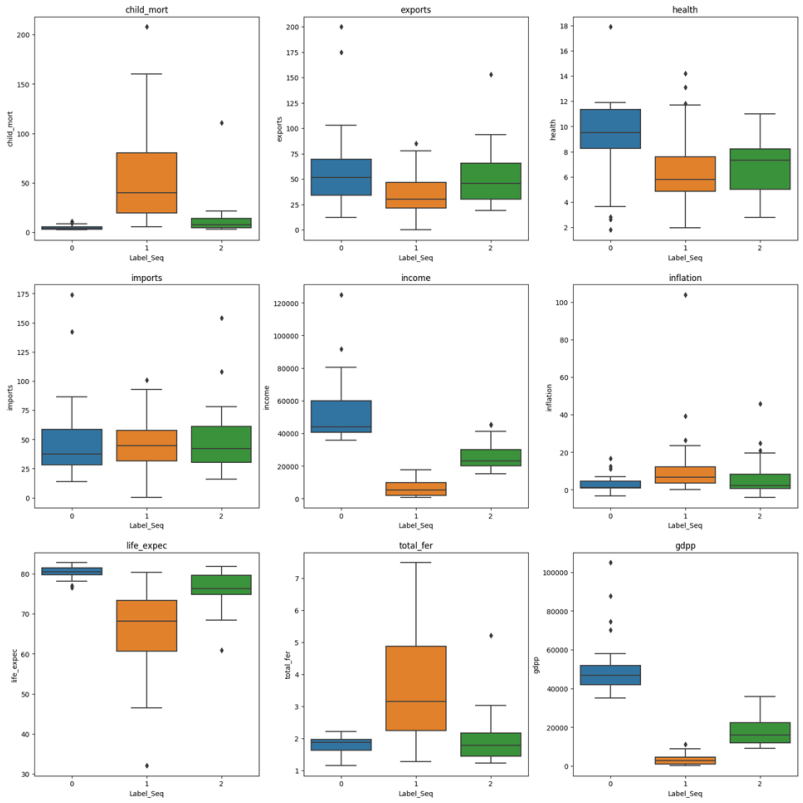
Similarly, these are the Silhouette Score of sequential and parallel K-Medoids Clustering algorithm:

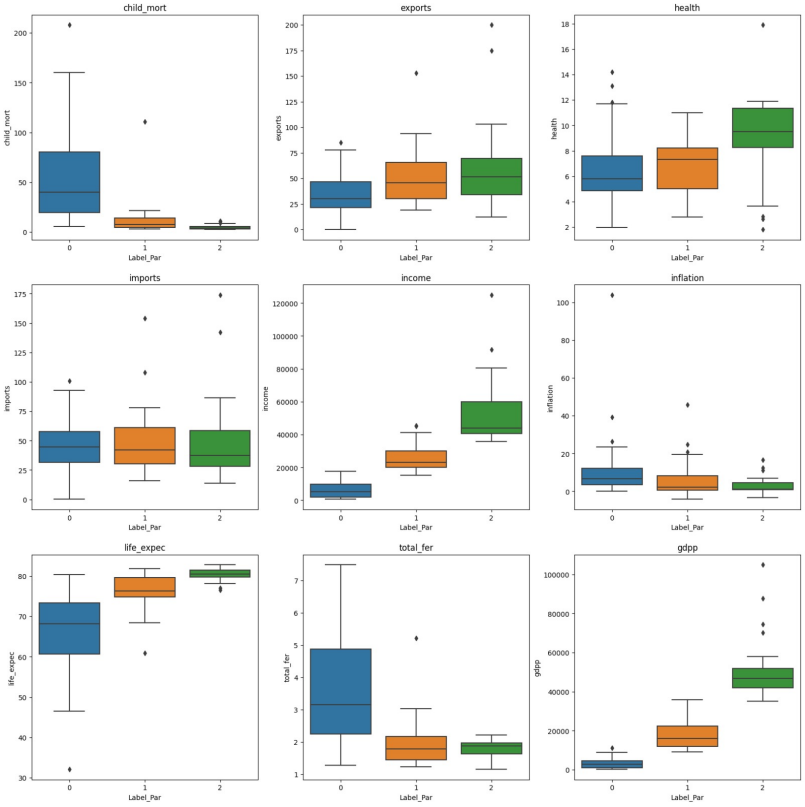
```
Performance of Sequential K-medoids algorithm:  0.6492494728121732
Performance of Parallel K-medoids algorithm:  0.6498575328920972
```

Results for clustering k-means algorithm:



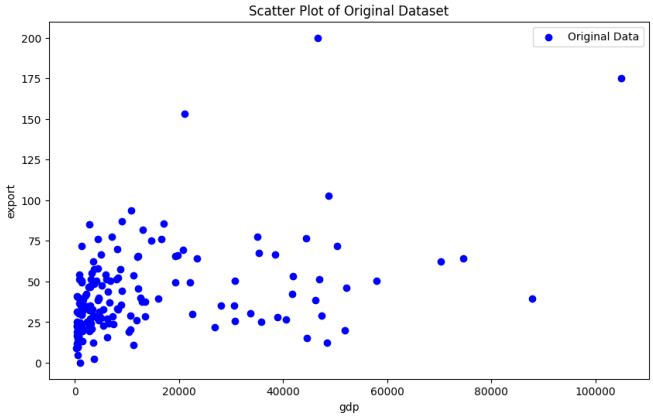


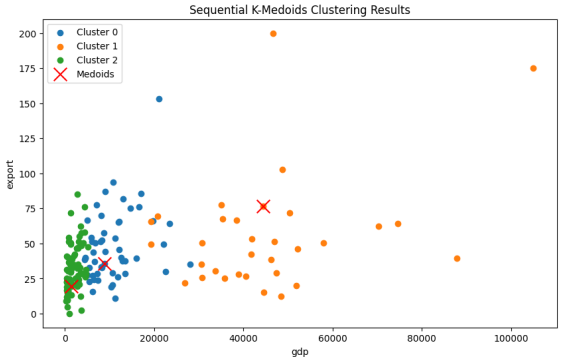

child_mort
exports
health
imports
income
inflation
life_expec
total_fer
gdpp
Label_Seq

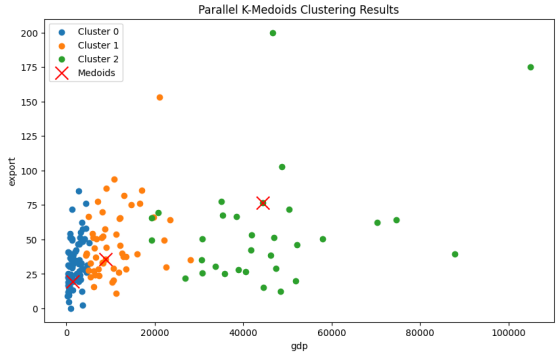

child_mort
exports
health
imports
income
inflation
life_expec
total_fer
gdpp
Label_Par

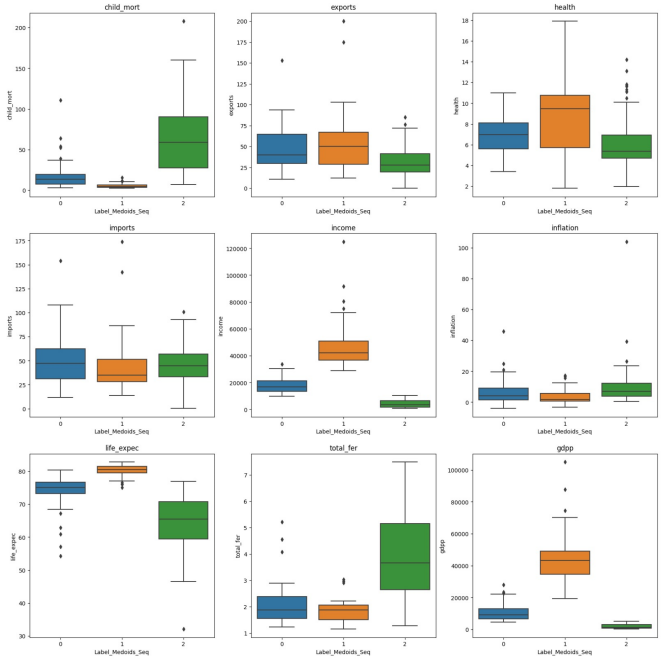
Results for clustering k-medoids algorithm:

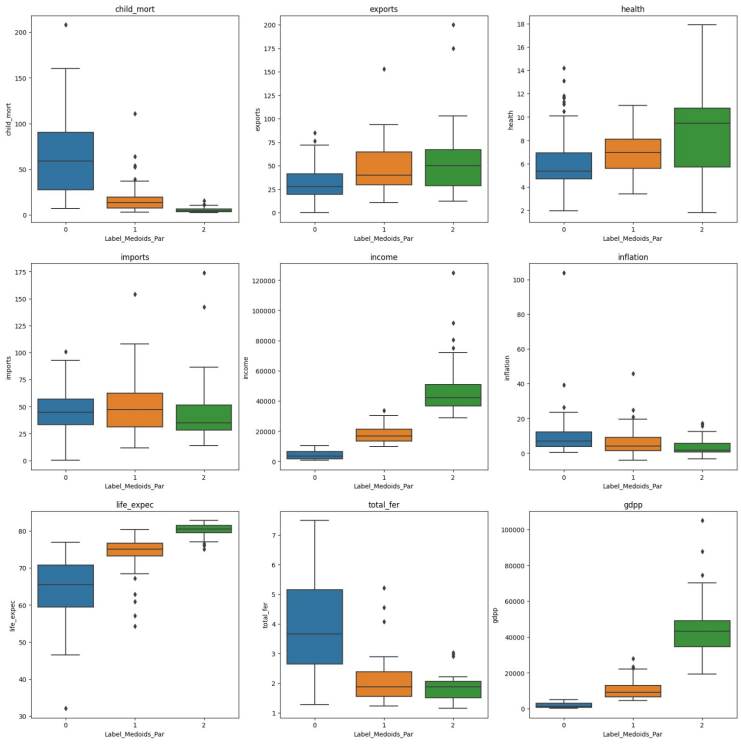
Cluster assignment in k-medoids:











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.