Unsupervised-based Distributed Machine Learning for Efficient Data Clustering and Prediction

Vishnu Baligodugula[1], Fathi Amsaad[1], Vincent Schmidt[1], and Noor Zaman Jhanjhi[1]

[1]Affiliation not available

January 26, 2024

Abstract

Unsupervised ML-based approaches have emerged for driving critical decisions about training data samples to help solve challenges in many life critical applications. This paper proposes parallel and distributed computing unsupervised ML techniques to improve the execution time of different ML algorithms. Various unsupervised ML models are developed, implemented, and tested to demonstrate the efficiency, in terms of execution time and accuracy, of the serial methods as compared to the parallelized ones. We developed sequential, parallel, and distributed cloud computing unsupervised ML models based and determined the most efficient model through comparative analysis. As a case study, sequential, parallel, and distributed approaches of Simple K-Means, Minibatch K-means, and Fuzzy C-Means are investigated to study the developed models' efficiency using country datasets for multiple organizations to train and test the developed model. Parallel and distributed computing models are developed utilizing could computing architect, i.e., cloud Amazon SageMaker, to study their efficiency in the execution time and model accuracy. The results show that the proposed parallel and distributed Fuzzy C-Means outperforms the other two clustering methods in terms of execution time with 0.932ms and 0.623ms with a minimal impact on the accuracy of the developed models.

Unsupervised-based Distributed Machine Learning for Efficient Data Clustering and Prediction

*Vishnu Baligodugula, *Student Members, IEEE;*
*Fathi Amsaad, †Vincent Schmidt, and ‡Noor Zaman Jhanjhi, *Senior Members, IEEE*

***Abstract*—Unsupervised ML-based approaches have emerged for driving critical decisions about training data samples to help solve challenges in many life critical applications. This paper proposes parallel and distributed computing unsupervised ML techniques to improve the execution time of different ML algorithms. Various unsupervised ML models are developed, implemented, and tested to demonstrate the efficiency, in terms of execution time and accuracy, of the serial methods as compared to the parallelized ones. We developed sequential, parallel, and distributed cloud computing unsupervised ML models based and determined the most efficient model through comparative analysis. As a case study, sequential, parallel, and distributed approaches of Simple K-Means, Minibatch K-means, and Fuzzy C-Means are investigated to study the developed models' efficiency using country datasets for multiple organizations to train and test the developed model. Parallel and distributed computing models are developed utilizing could computing architect, i.e., cloud Amazon SageMaker, to study their efficiency in the execution time and model accuracy. The results show that the proposed parallel and distributed Fuzzy C-Means outperforms the other two clustering methods in terms of execution time with 0.932ms and 0.623ms with a minimal impact on the accuracy of the developed models.**

***Index Terms*—Clustering, K-Means, Minibatch K-Means, Fuzzy C-Means, Parallel MPI, Parallel Cloud Computing.**

I. Introduction

In machine learning, one of the primary goals is to extract valuable insights or patterns from large datasets. One widely adopted technique for grouping similar data is the data clustering algorithms. These algorithms evaluate the similarity among distinct objects within a dataset and subsequently group them based on these similarities. With the growing volume of data being amassed using scientific data collection techniques, it has become increasingly imperative to ensure that machine learning algorithms are effective, efficient, and scalable. One way to improve machine learning algorithms' performance is to parallelize them, leveraging modern computing infrastructures. However, the efficacy of parallelization is heavily contingent upon the data organization and choice of parallelization approach.

Simple K-Means algorithm is one of the first and most commonly utilized techniques. There are a few issues related to the K-Means algorithm. They are as follows: one is the initial centers initialized randomly, which is a significant issue. The other issue is that there will need to be some idea of the clusters in the dataset. Minibatch K-Means is a clustering algorithm that splits a dataset into a fixed number of groups or clusters.

Instead of processing the entire dataset, Minibatch K-Means works on smaller, randomly chosen subsets of data known as mini-batches. This approach makes it more efficient and capable of handling larger datasets that may not fit into memory. The algorithm assigns each data point to the nearest centroid and updates centroid positions based on the mean of the assigned data points. While Minibatch K-Means is faster and more scalable, it may only sometimes produce optimal results compared to the standard K-Means algorithm, mainly when the mini-batch size is too small.

Fuzzy C-Means (FCM) is an effective clustering algorithm that groups data points into distinct clusters based on their similarities. This unique algorithm enables overlapping clusters by ascribing a fuzzy membership value to each point, signifying its membership degree to each cluster. FCM continually calculates and updates the cluster centers and membership values until a predefined stopping point is reached. Many scholars have concentrated on parallelizing these techniques based on these issues.

Nevertheless, all these algorithms have a few drawbacks, One such drawback is the utilization of parallel systems with constrained programming models that automatically parallelize data line by line. This approach assumes the need to store all data in memory line by line simultaneously, which becomes highly impractical when dealing with extensive datasets containing millions of records. Consequently, a trade-off between speed and efficiency is often observed, where prioritizing one aspect may come at the expense of the other. To achieve optimal performance, it is essential to strike a balance between efficiency and accuracy.

This paper aim to investigate multiple Parallel and distributing computing for efficient unsupervised clustering. We developed different ML models for countries with funding for necessities and assistance during disasters and environmental

The authors would like to express their sincere gratitude for the technical and financial support provided by the Air Force Research Lab (AFRL) under the Assured and Trusted Digital Microelectronics Ecosystem (ADMETE) grant, BAA-FA8650-18-S-1201. Additionally, partial support for this work was received from the National Security Agency (NSA) funding to Wright State University in Dayton, Ohio, USA. This project adhered to CAGE Number: 4B991 and DUNS Number: 047814256.

Vishnu Baligodugula and Fathi Amsaad, ***Corresponding Authors***, are affiliated with the Department of Computer Science and Engineering, Wright State University, 3640 Colonel Glenn Hwy, Dayton, OH 45435, USA. Emails: (ghimire.18, fathi.amsaad)@wright.edu

Vincent Schmidt is a Senior Research Computer Engineer at the United States Wright Patterson Air Force Research Lab (US WP-AFRL) Email: (vincent.schmidt us.af.mil

Noor Zaman Jhanjhi is associated with the School of Computer Science Faculty of Innovation and Technology, Taylor's University, 47500 Subang Jaya, Selangor, Malaysia. Email: noorzaman.jhanjhi@taylors.edu.my

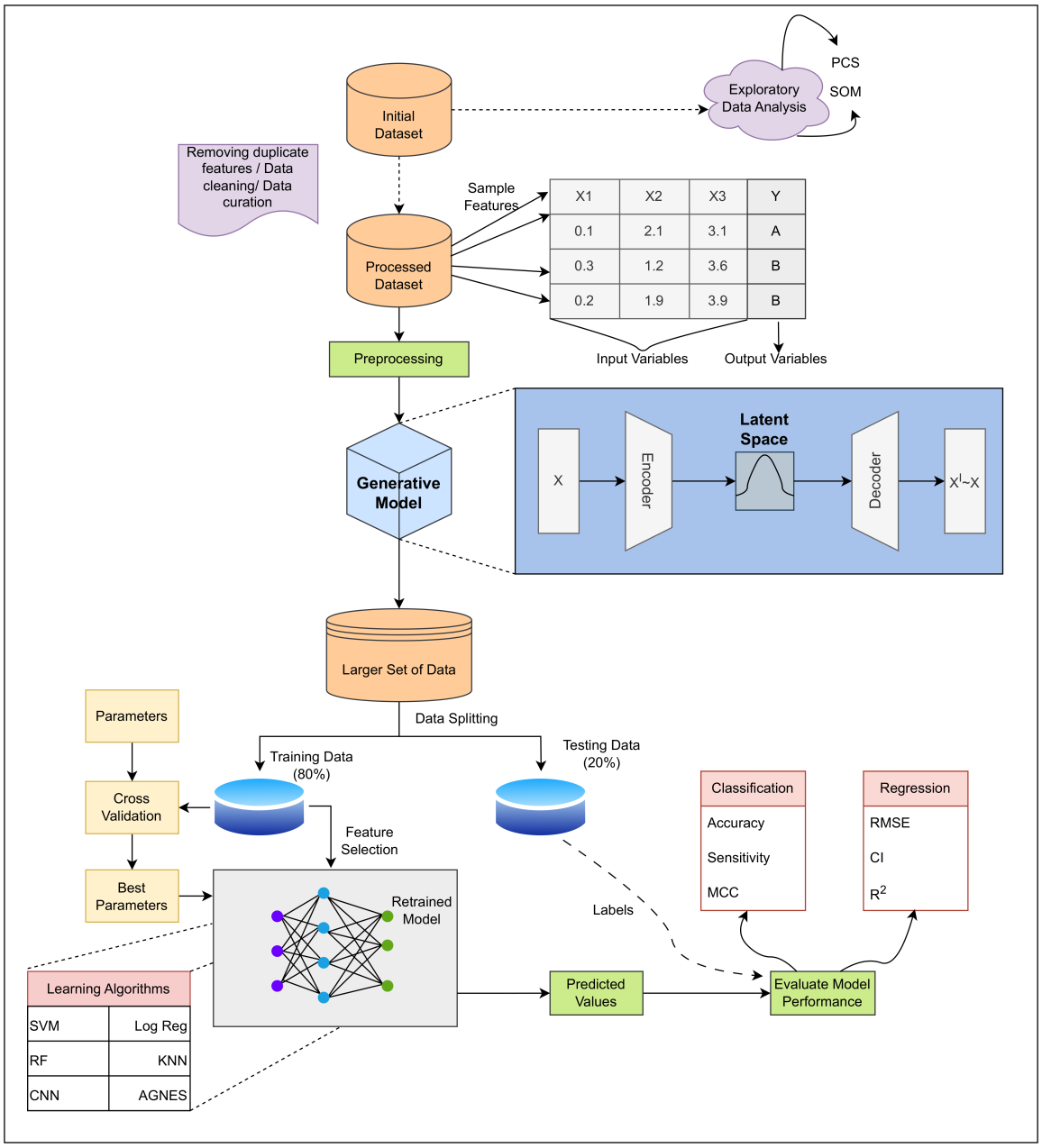

Fig. 1. Basic Model of Machine Learning

harm. For that, we divided the nations into categories based on the economic and medical factors that will contribute to their overall development. Organizations can determine which country needs support with the aid of analysis.

The K-Means, minibatch K-Means, Fuzzy C-Means serial approach, open MPI parallel technique, and AWS distribution strategy were all used in the investigation. We tested the speed and accuracy of the algorithms using the three different ways to make the comparison, and we then presented the findings. We used Python to implement parallelized MPI-based algorithm on AWS for computing distribution from scratch. Similarly, Minibatch K-Means, and Fuzzy C-Means utilizing the AWS cloud platform. The results shows that parallel and distributed computing MPI approaches can enhance the performance and quality of the solution of the developed ML-based models.

The reset of this paper is organized as follows. Section 2 provides background information on machine learning algorithms, Message Passing Interface and Amazon SageMaker. Section 3 related work. Section 4 focuses on parallelization strategies, describing the methods and requirements necessary for parallelization. Section 5 details the research contributions to K-Means, MiniBatch K-Means, and Fuzzy C-Means, as well as the understanding of parallelization and cloud algorithms, and the experimental setup.

Section 6 covers the evaluation matrix. Section 7 presents the results, discussions, and conclusions regarding the performance of cloud and parallel clustering distribution techniques

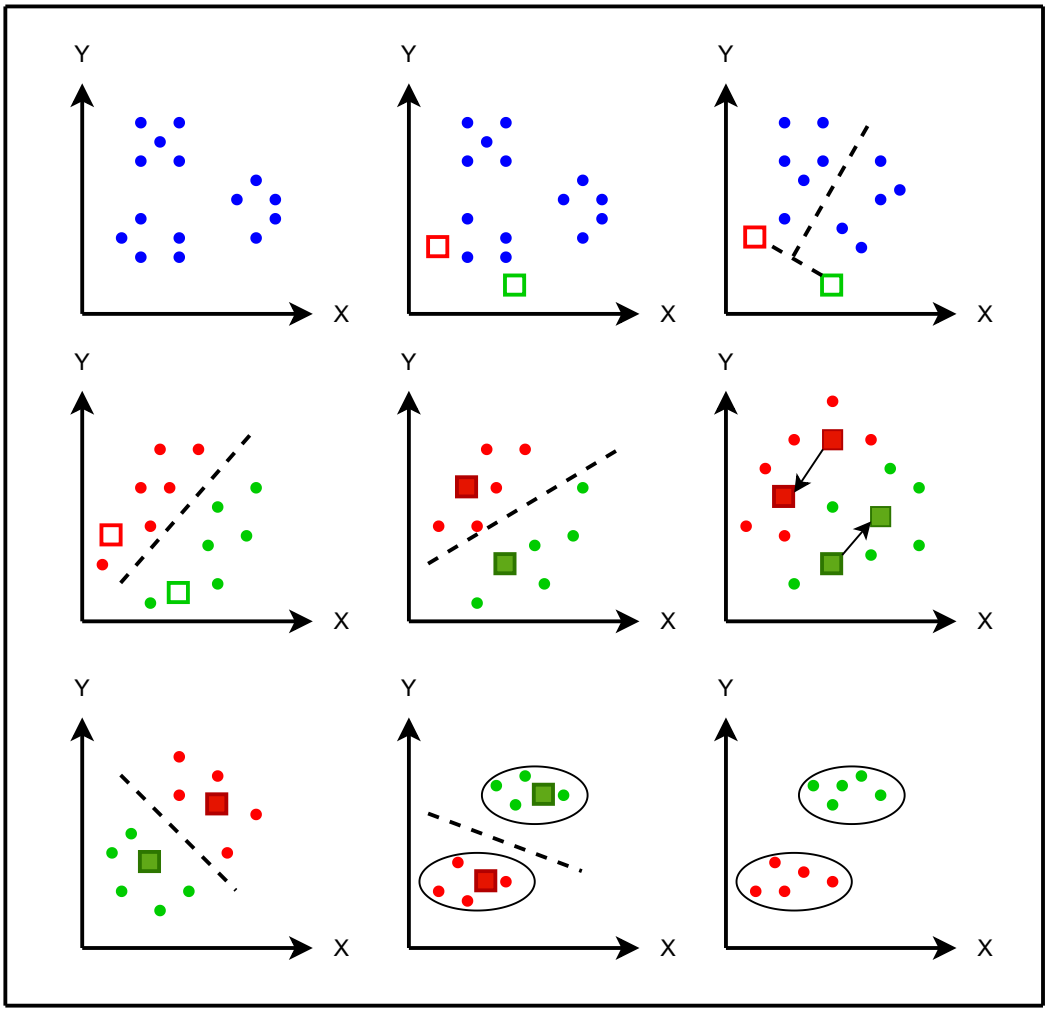

Fig. 2. Clusters formation

versus sequential clustering. Section 8 explores potential applications. Finally, the References are included.

II. Background

This section will cover six major topics: Machine Learning models, K-Means, Minibatch K-means, Fuzzy C-Means, Parallel Message passing interface and AWS Sage maker

A. Machine Learning

Machine learning is a subclass of artificial intelligence which allows computer systems to improve performance by learning from experiences without human instruction. It entails using large datasets to train algorithms capable of detecting patterns, making predictions, and enhancing decision-making abilities. Machine learning is classified into three main categories: supervised learning, unsupervised learning, and reinforcement learning, each having its unique techniques and areas of application.

Supervised learning deals with labeled data wherein the algorithm learns to recognize patterns, relationships, and correlations between inputs and outputs to produce accurate predictions or classifications. Unsupervised learning, on the other hand, works with unlabeled data, which means that the algorithm must identify and extract insights from the data by itself without any predefined labels. Reinforcement learning, often used in robotics and game development, relies on trial-and-error to learn the best approach to solve a particular task.

As machine learning continues to evolve, it has found applications across several industries, including healthcare, finance, retail, and transportation. It has enabled organizations to analyze diverse data sources, make informed decisions, and increase operational efficiencies. With the growing availability of data and processing power, the potential for machine learning is enormous, and its impact on various industries is expected to be transformative and here i am presenting basic model of ML Figure 1.

B. K-Means Algorithm

These grouping issues in machine learning or information science are resolved using unsupervised learning K-Means Clustering. In this chapter, we will study the K-means clustering method, how something operates, and how to apply it in Python.

Unsupervised training method K-Means Cluster divides a piece of unlabeled information into various clusters. Thus, K describes how several clusters that were not previously present should be generated due to the procedure; for instance, if K=2, the outcome will primarily be two types. If K=3, the outcome will be three clusters, and so on.

This iterative method separates the unlabeled information into k distinct clusters, each containing one dataset and sharing a set of features. It allows us both to categorize all data into different groups. It offers a workable technique for quickly and effectively determining the groups within the unlabeled dataset without having to do any training.

Every cluster has a corresponding centroid in this centroid-based approach. The principal objective of such an approach is to reduce the distance between the data points and the groups they belong.

This method starts with an intake of an unlabeled dataset, separates this into k clusters, and continues the procedure until no better clusters are found. In this method, the k value must be known in advance.Its two main functions of the k-means suggested method are:

- Determines the optimal quantity of K points or centroids via an iterative algorithm.
- Each piece of information is paired with the closest k-center.Those information fragments near a certain k-center combine to form a cluster.

Each cluster is unique and has samples that only share a small amount in a joint.

The following stages illustrate how the K-Means method functions :

- To figure out how many groups, choose K.
- Pick K locations or cluster centers randomly. (It could not be the supplied data.)
- Allocate every piece of data to its nearest centroid, which will create precisely the number of clusters that have been predetermined.
- Determine the variation, then relocate every cluster's centroid.
- Reassign every piece of data to an updated centroid of every cluster by repeating the third section.
- Move to Phase 4 if there is a reassign; otherwise, go over to Finished.
- A finished design.

Now let us analyze the graphical plots to comprehend the steps mentioned earlier:

Below points is the explanation of graphs each point representing each graph formation Figure 2.

- Consider that there are two independent variables, P1 and P2.
- To identify the data set and divide it into various clusters, let us use a value of k for clusters or K=2. This implies that we will divide those data sets into two separate groups.To create a cluster, k points or the centroid must be chosen randomly. These coordinates may be taken from the dataset or in different locations. Since these two do not appear in the dataset, It was choosing them as the k points in this instance.
- Currently, we would identify the nearest K-point and centroid for every data point on the scatter plot. To find the distance separating two points, we will estimate it using the math we studied. To create a midpoint between both, the two centroids shall do such.
- It is evident from the previous picture that locations along the line to the left were near a K1 center, which is red, and white spots along the lines to the right are near a red centroid. For better comprehension, let us color both red and green.We will repeat the procedure by picking a new centroid to locate the nearest cluster. Let us determine those centroids centers of gravity ordered to select their fresh centroids.
- We would then reallocate every piece of data here to a new centroid. You can conduct the same procedure for locating a mean line in this, and your average should look such as the illustration of 5 one in above figure.There is one red point on the left end of the line, plus two green points on the lines. Thus, fresh centroids will be assigned to such three locations.
- Since reassign has occurred, we once more go to step 4, locating fresh centroids called K-points.Determining the exact gravitational centers of a new centroid would allow us to continue the operation.
- We will construct the average lines once more and reassign all pieces of data because we have the updated cluster centers.
- This is established due to the need for more consistent data points along both sides of the boundary.
- Now that our model is complete, we may exclude the two remaining groups and the presumptive centroids.

1) Elbow Method: The Elbow Mothod is widely used for determining the ideal number of clusters. A WCSS-valued idea is applied in this technique. The term "total variations inside a cluster" is abbreviated as "WCSS" and means Within Cluster Sum of Squares. This formula listed below is used to determine the number for WCSS (for 2 clusters):

The WCSS equation is:

$$WCSS = \sum_{j=1}^{k} \sum_{P_i \in C_j} (distance(P_i, C_j))^2 \qquad (1)$$

From the formula,

In this equation, k represents the number of clusters, Cj represents the j-th cluster, and Pi represents the i−th point in the dataset.

The distance function in the equation calculates the distance between a data point Pi and the centroid of its assigned cluster Cj. The WCSS metric is important because it measures how well the data points are clustered together. The lower the WCSS value, the better the clustering solution.

The goal of the K-means clustering algorithm is to minimize the WCSS by iteratively adjusting the cluster centroids until convergence is achieved. The final result of the K-means algorithm is a set of k cluster centroids that best represent the data in the dataset.

Its elbows technique is so named because the diagram depicts a steep bend that resembles an elbow. A graph for

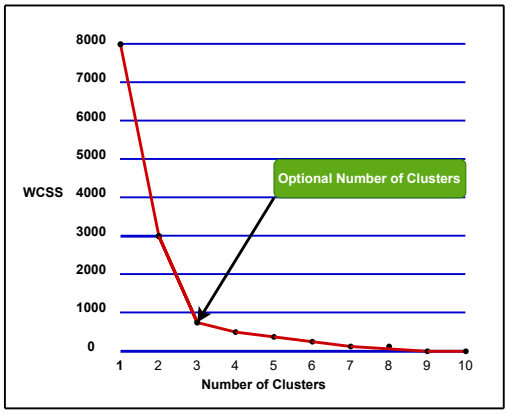

Fig. 3. Elbow Method Graph Representation

the elbows approach resembles what illustration below Figure 3

C. Minibatch K-Means

Minibatch K-means is a widely used clustering algorithm that divides a given dataset into K clusters by minimizing the distances between the data points and their associated cluster centroid. In contrast to the standard K-means algorithm, Minibatch K-means processes the data's small, randomly selected subsets, or Minibatches, to expedite convergence while maintaining comparable clustering accuracy. It is considered efficient and scalable for large datasets that exceed the available memory capacity. The algorithm assigns each data point to its closest cluster center and updates the centroids iteratively until convergence is achieved. The number of Minibatches and the batch size are hyperparameters that can be optimized to enhance the clustering performance and accuracy of the resulting partitions.

D. Fuzzy C-Means

Fuzzy C-means is a prominent unsupervised clustering approach used to group data points into distinct clusters based on their similarity. Unlike complex clustering methods such as K-means, Fuzzy C-means assigns each data point a degree of membership, represented as a fuzzy value, for every cluster. This allows a data point to be part of multiple clusters at the same time with different membership degrees. The algorithm adapts the cluster centers and their fuzzy memberships iteratively based on the data point similarity and the level of overlap among the clusters. The hyperparameter 'fuzziness' regulates the degree of blurriness of the memberships, with higher values leading to more fuzziness and an increased number of clusters. Fuzzy C-means can be utilized to address various clustering problems and provide soft classification outcomes for the data points.

E. Parallel Message Passing Interface

A standard way to communicate throughout many computers that run concurrent programs all over distributed storage is indeed the message-passing interface (MPI).

The term "node" refers to a group of processors, or even a group of core processors on a single computer, in parallel Computing. Generally, every node inside a parallel configuration focuses on a distinct aspect of an immense processing challenge. It thus becomes difficult to coordinate the activities of every simultaneous network, transfer data across nodes, or exert control over the entire parallel cluster. The message-forwarding interface defines a standardized set of commands for such activities. When a message is sent to an entity, parallel process, subroutine, function, and thread, it is usually called "passing a message." That message is again utilized to begin a different technique.

Although no formal reference implementation, such as the International Organization for Standardization or the IEEE Institute of Electrical and Electronics Engineers, had supported MPI as a benchmark, it is widely regarded as an industry norm. It serves as the base for most network systems used by parallel computing programming. Researchers also produced several MPI versions.

Fortran, C, C++, Python and Java procedures or libraries have usable syntax defined by MPI.

1) Features of the Message Passing Interface that are advantageous:

- Standardization: Previous message queue packages have been supplanted by MPI, which is now a widely recognized mainstream technology.
- A large group formulated it: Despite MPI is not a nationalized, a committee of manufacturers, integrators, and users came up with it.

2) Flexibility: Since MPI has already been developed for numerous distributed memory designs, users can move their software to other systems which the MPI standard supports without changing the source code.

- Efficiency: Usually, functionality is tailored to MPI's device, and vendor implementations could be tuned to take advantage of built-in hardware characteristics.
- Feature set: Maximum performance on parallel processing systems and groups is a crucial feature of MPI. Furthermore, over 100 specified routines make up the fundamental MPI-1 architecture.

3) Terminology used by MPI: Key terms and instructions: A few fundamental MPI ideas or instructions are listed below Figure 4:

- Comm: This MPI communication classes link several stages. An enclosed consultation processes an individual identifier from a communication instruction, which arranges it into an orderly topology. For instance, MPI COMM WORLD is a signal for global communication.
- Color: This means that a process is assigned a color, and all activities with that coloring are housed within the same communication. MPE Make color array is a color-related instruction that modifies the offering of the product.
- Key: A key determines the ranking or ordering of a procedure in communication. The order is established by the application's position inside the communication if two methods are assigned the same key.

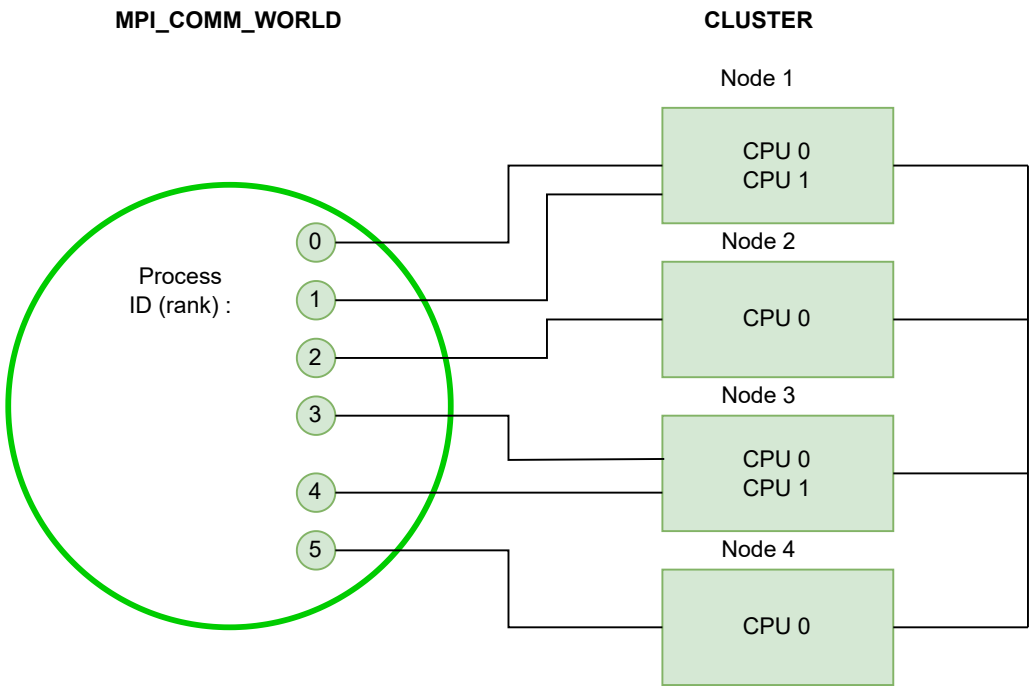

Fig. 4. The rank assigned to every CPU is demonstrated using an MPI COMM process with numerous endpoints distributed across four cluster.

- New comm: Fresh communication can be created using this instruction.
- Categories of data collected: A description of the kind of data exchanged among tasks is required for MPI functions. These variables can be predefined with the help of MPI INT, MPI CHAR, or MPI DOUBLE.
- Refer: It communicates among two particular operations. Two popular delaying techniques for juncture messaging were MPI Send and MPI Recv. Block is the method by which the transmitting and receiving systems hold off on transmitting and finishing a communication until a whole signal has been appropriately transmitted and received.
- Cooperative foundations: All activities in a particular process must communicate to perform these group tasks. However, one method is MPI Bcast, which distributes information from a single node to each process in a process.
- One-sided: Usually, when this phrase refers to external communications like MPI Put, MPI Get, or MPI Accumulate. Publishing to storage, receiving to recollection, or minimizing operations on a single recollection between activities are specially mentioned.

4) MPI's Past and Iterations: Around 1991, a tiny research group from Austria started talking about either a message-carrying interface. A Center for Studies in Parallel Computing supported a workshop about message-passing protocols inside a distributed memory system that occurred in Williamsburg, Virginia, per year afterward. Therefore To develop the standard method, a work was formed.

The prototype of MPI-1 was developed in Nov 1992, and the specification was published in 1993 just at Supercomputing '93 conference. Around 1994, MPI version 1.0 was published after receiving more comments and modifications. Since then, MPI has been open to everyone working in the powerful computational field, with more than 40 companies currently involved.

About 115 capabilities are offered under the more dated MPI 1.3 specification, sometimes known as MPI-1, and more than 500 features and a high degree of older systems using MPI-1, a subsequent MPI 2.2 specification, or MPI-2, are available. Unfortunately, a complete MPI-2 version is not offered by all MPI implementations. Along with variable system integration and remote storage functions,MPI-2 introduced novel concurrent I/O. The Nov. introduction of the MPI-3 specification results in increased speed, multicore or cluster capabilities, and improved scalability, with availability and high interoperability. MPI 4.0 was published by The MPI Forum around 2021. It featured innovative features like Permanent Communes, segmented networking, and a new product layout. Now, MPI 5.0 is being created.

F. AWS Sage Maker

1) Overview: Most data analysts use the hosted platform to develop, train, and publish machine learning algorithms, and they were regrettably still unable to change resources as required. To go live quicker and with less expense, AWS Sage Maker makes it simpler for programmers to design and retrain models. This essay will review the capabilities, use applications, and advantages of AWS Sage Maker and Computer Vision using AWS Sage Maker.

2) AWS: A cloud-based system that offers services that are ordered over the web is known as Amazon Web Services. Every public cloud type may be developed, monitored, and

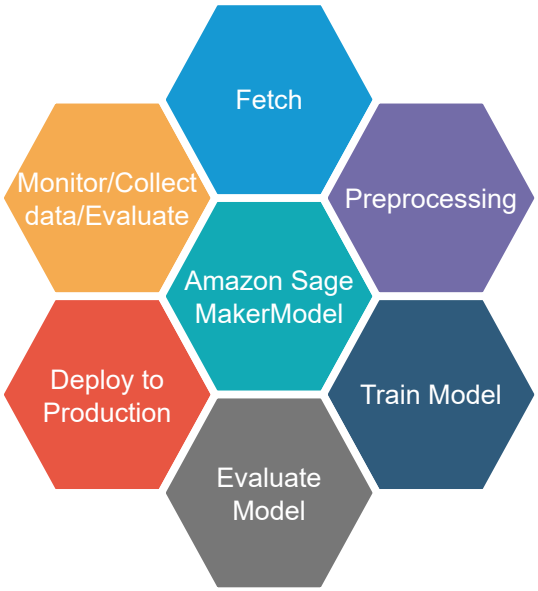

Fig. 5. Model of AWS sage maker.

deployed using AWS services, which is where AWS Sage Maker is useful.

3) About AWS Sage Maker: Amazon manages the machine learning services offered as Sage Maker. Sage Maker enables data researchers or programmers to create machine learning models quickly, train those, and afterward instantly release them into a virtual machine that is prepared for use in operation. It usually includes methods for machine learning designed for application in a distributed scenario using extensive data sets. Sage Maker provides flexible, dispersed education options that fit existing operations Figure 5.

4) Attributes of Sage Maker: Sage Maker now features new features that Amazon has developed since its debut in 2017. Accessibility to functionality is made possible via AWS Contributory factor Studios, an Integrated Development Environment that combines all abilities. One of two methods can be utilized to build a Jupiter notebook:

- Sage Maker Studio's internet version of the IDE.
- A machine learning server in Amazon Sage Maker that Amazon EC2 hosts.

5) Utilizing AWS Sage Maker and machine learning: Let us examine the development, testing, fine-tuning, then deployment of a model for machine learning utilizing AWS Sage Maker.

6) Builds:

- There are approximately 15 popular learning algorithm skills and making included.
- It enables us to select the configuration settings needed for the notebook instances.
- To start coding, a user may use a notebook example (for building model training tasks).
- Choose and enhance the necessary methods, such as: K-Means,Linear Regression and Logic-based Regression.
- AWS Sage Maker's Jupiter notebook API allows programmers to alter Supervised Learning servers.

7) Tune and Test:

- All required modules should be made or imported.
- During Amazon model development, establish and control just a few configuration settings.
- The model is taught and tuned using Sage Maker.
- Sage Maker achieves a set of parameter tuning by combining some algorithm parameters.
- Sage Maker uses Amazon S3 for keeping records as it is a secure protocol.
- Sage Maker uses extensible ECR to control Container technology.
- Docker setup, management, and storage are made more accessible with ECR.
- The developed skills source is preserved in ECR, whereas the supervised learning is separated and stored in Amazon S3.
- Next, Sage Maker generates, learns, and stores a given input cluster in Amazon S3.

8) Implement:

- Upon adjusting, objects could be released to Sage Maker destinations.
- Just in the end, a forecast is made in real-time.
- To determine whether the strategy has achieved your company goals, one must evaluate it.

9) Uses of Sage Maker: Many different businesses use AWS Sage Maker. Machine learning groups use Sage Maker for the below task:

- Accessing and exchanging codes.
- Accelerate the creation of production-ready AI components.
- Enhance data interpretations and build more exact database schemas iteratively.
- Streamline data both in and out.
- Much information is to be processed.
- Codes for modeling exchange.

10) Advantages:

- We are making deep learning easier to use.
- More people will be able to innovate using machine learning thanks to integrated tools like computer scientists and also no potential input for analysts.
- Huge data preparation.
- Collect, categorize, and analyze vast quantities of organized (tabular) and unorganized (photos, videos, and audio) information via computer vision.
- Accelerate the advancement of computer vision.
- Without improved facilities, learning can be completed in just a few moments instead of days.
- Using specially designed tools, employees to comply might rise as much as ten times.
- Enhance the machine training cycle to be more efficient.
- Streamline or unify MLOps practices throughout a company to design, train, publish, and maintain algorithms at volume.

11) Modern Developments: Since the initial release of reinvent 2021, Amazon has added a feature for its Sage Maker Python SDK that offers abstracts to speed up models' deployments, plus Model Registry, which makes it easier to connect virtualized inferences destinations and MLOps processes. One can utilize Sage Maker Cloud Hosting Deduction with high-traffic applications now that Amazon has increased the provisions requests per endpoints limitation from 50 to 200.

12) Conclusion: AWS charges every sage Maker user for Computing, memory, or information computer resources used to create, test, deploy, and log machine learning algorithms or predictions. The expenditures of S3 were related to maintaining train and prediction data sets.The sage Maker application's design is flexible and responsive, enabling the whole lifespan of machine-learning applications through modeling creation to model implementation. This implies that the sage Maker can be used independently for the proposed model, retraining, or distribution.

III. Related Work

The parallel K-Means technique, according to Mohanavalli, Jaisakthi and Aravindan, Both a distributed memory system utilizing MPI programming and a memory version utilizing OpenMP programming has a parallel K-Means model developed based. With MPI programming, a hybrid OpenMP implementation was also tested. To calculate Amdahl's effect to evaluate the speedups achieved and the efficiency of the parallel algorithms were examined. In addition to being 50 percentage faster than MPI, the hybrid technique performed better under a balanced load [1]. They have demonstrated that this approach is added to increase the effectiveness of parallel K-Means. Both the Enslaver/Slave concept and the parallel processing method are used. The tests revealed that this technique is more effective and applicable to Yufang Zhang's situation[2]. Showed this is suggested to use an initiation technique for K-Means parallel processing, establish the first cluster centers, that not only speeds up performance but also produces consistent results conducted by Swamy, Raghuwanshi and Gholghate[3].

Boukhdhir, Lachiheb and Gouider's research focuses on increasing K-Means' capacity to deal with large datasets by speeding up its execution, suggesting a map Reduce-based approach. In addition we will suggest two additional algorithms.The first eliminates unnecessary, and the other dynamically chooses the starting cluster center to stabilize the outcome.The suggested technique is demonstrated to be significantly quicker than three different known algorithms from the literature when implemented just on the Hadoop framework[4]. Applied Parallel MapReduce-based K-Means clustering approach because it is a straightforward yet effective method for parallel programming. These results of experiments indicate that the suggested algorithm can process massive datasets on affordable equipment while scaling up and down effectively research evaluated by Weizhong, Huifang and Qing [5].

Work targeted is clicked Plus, OpenMP and CUDA just on CPU and GPU, respectively, are used to improve its modified version for the K-Means cluster. Various data, including photos of varying data quantities, are shown, including the findings, with such a focus on rather big data. Covered are various combinations of characteristics and clusters[6]. So, to speed up K-Means, the study presents two novel methods for transmitting data. Expanded Vector is the initial method (KMMR-EV). K-Means over MapReduce utilizing Boundaries File is indeed the name of the second technique (KMMR-BF). Compared to the simple MapReduce implementation of K-Means, both methods achieve speed up had undergone considerable experimental investigation using actual and artificial data, coupled with an excess assessment to demonstrate the efficacy between both methods[7].

One goal of the researcher would be to create a K-Means method version that can run on a standard PC using NVIDIA graphics cards or tackle more extensive data sets using CUDA[8]. It concentrates upon cluster method performance problems, and a more sophisticated initial centroids core approach is used. The topic of a single CPU computer's operational limit while dealing with large data sets also is investigated, and a parallel K-Means technique is investigated. The analysis shows that such advancements can significantly increase performance, allowing for the group of many data sets less precisely and fast analysis shown by Tian, Zhu, Zhang, and Liu[9].

Using GPU's cluster, we demonstrate the conception and execution of an effective parallel K-Means algorithm. They dynamically use load balancing to evenly spread demand across the many GPUs inside the cluster to improve the overall cross-performance of the parallel K-Means. Additionally, they utilize software distributed shared memory to facilitate inter-node cooperation and communication. By keeping load balance on GPU clusters, the evaluation's findings demonstrate the parallelism K-Means' enhanced results.[10].

The latest update of K-Means separates overall issues into minor issues handled separately with one or more GPUs' Broadcast Multiprocessors. They had developed the Graphics card K-Means (CUDA) authors[11]. It creates a buffer among collected data that did not alter their clusters over the following group, which might dramatically lower the burden of large data

sets. Its operation time was reduced by up to 70 Percentage inside the prototype system[12].

On such a CPU-GPU heterogeneous system, they provide a quick implementation of the spectral clustering technique. My method uses the multicore CPU's processing capability and the GPU's huge multithreading and SIMD abilities. They suggest a concurrent technique to generate a sparse refers to a network expressed in a common sparse codified form; the data points are given as input in high-dimension space. Next, using the ARPACK program, the CU SPARSE library's backward transceiver k is ordinarily quite large—where the smallest k eigenvectors were computed by three of the Channel matrix. Additionally, we use GPU's to create a rapid parallel processing K-Means method, and the solution is much quicker[13]. A technique known as the enhanced K-Means method was used to expand that standard K-Means strategy. Our experimental findings demonstrate that such an expanded K-Means algorithm may effectively organize the data when the number of components for each grouping has to be constrained. This method changes the number of attributes in every category by utilizing a greedy strategy by faliu and inkyu[14].

This genetic algorithm is utilized to optimize K-Means grouping to ensure that the shortcomings of K-Means can be overcome.K-Means using the GA algorithm suggested innovative products in this field of research when the outcomes of conventional K-Means clustering versus genetics K-Means grouping were contrasted and analyzed[15].

The first is MPI, and the second is shared memory utilizing OpenMP, or heterogeneous Computing utilizing CUDA-C-programmed NVIDIA GPUs. The first investigation contrasts the various strategies, while others have worked at speeding K-Means grouping. Where effectiveness of K-Means also has highly dependent on the first means used. We compare many thermally activated in serial and select the most effective one to use throughout the method. From modest (300300 pixels) to huge (11641200 pixels), I test the results on various photos. Our findings demonstrate that almost all three distributed programming paradigms resulted in speedups, with OpenMP used for small pictures and CUDA-C for bigger pictures yielding the best results by Janki, Miriam, and ningfang[16].provided a hybrid algorithm that decides when to employ the triangle inequality in an improved way. Numerous studies show that our approach performs better than the Computer or Multicore Kmeans[17].

The examination of share price pricing data from the prior yrs., with findings interpreted following intensive training using an algorithm for machine learning on CUDA, keeping in mind the time restrictions of trading. With the aid of machine learning approaches, the program's performance has been significantly increased. In this research, a massively parallel methodology is applied to hasten the generation of findings. Compared to typical ways of employing a solitary Processor Core Unit, the execution time is significantly decreased due to the recent high achievements of CUDA parallel computing technology (CPU). By accurately anticipating stock prices in advance, it reduced computation time by a significant margin and resulted in net profits, which is the end objective of trade.Just characterized by three groups as projected utilizing algorithms, traders could choose to maintain the share, sell it, acquire new equities, or stay impartial. If a user makes a neutral choice, that indicates he must hold onto any stock he already has and refrain from purchasing any more. This suggested approach is appropriate for trading stocks based on stock price[18].

Its efficiency of a program in a concurrent environment was enhanced by adding a K-Means clustering algorithm to an MPI4py module. This report examines the effectiveness of executing the K-Means method consecutively versus using a Message Passing Interface (MPI) parallel design for grouping information in the form of operational costs and processing time[19]. This study presented a parallelization K-Means method for sparse, elevated text (PKHT). This suggested technique provides an 11x shorter runtime by utilizing both GPUs with MPI by xiaolei, yanking, and yuxin[20].

Maximum performance, flexibility, and portability are all characteristics of MPI as a message-passing type system. This led to the motivation for this research to present MKmeans, a similar K-Means clustering technique with MPI. The approach allows efficient use of the clustering algorithm inside a parallel setting. Research using experimental statistics shows that MKmeans operates well on a variety of large amounts of data with very few time resources[21]. Researchers investigate leveraging NVIDIA Graphics Processing Units (GPUs) written with CUDA C to enhance the speed of K-Means clustering. Various optimization methods are being used, including using constant memory for cluster data, shared memory, or picture data. Its outcomes were assessed on various images, ranging in size from tiny (256x256 pixels) to large (1024x1024 pixels) and with 4 to 256 clusters. For four clusters, we observe that now the serial version is typically 9x slower than the parallel version. Even as the number of clusters rises to 256, the speedup jumps to 57x[22].

We suggested a parallel implementation of the conventional K-Means algorithm to run it on the Hadoop distributed framework. The research findings show that when clustering a vast volume of data, our suggested K-Means method works better than conventional K-Means[23]. In terms of overhead expenses and execution, the overall efficiency of K-Means grouping the information is examined among parallel and sequential implementation in the Message Passing Interface Infrastructure by Ragunthar, Ashok, and Gopinath[24].

The study utilizes NVIDIA's Compute Unified Device Architecture (CUDA) to create the GPU-based Harmony K-Means Algorithm in content clustering. During this testing, compared with CPU-based software, the GPU-based application could achieve a speedup of increase than Twenty times[25].An approach suggested in this research is based upon the modified K-Means algorithm of a Hadoop platform; initially, a preliminary clustering is obtained using the canopy algorithm, after which the pinpoint of the exact or cluster numbers is also adapted new using the ISODATA algorithm. Lastly, integrate the MapReduce distributed computation architecture with the K-Means method to achieve the ideally planned CI-K-Means technique to improve the convergence point list and K calculated value by the Canopy method. The experiments

show that now the CI-K-Means technique resolves the issues associated with the Canopy K - means algorithm's inconsistent result in a more excellent and also the complexity of choosing the Centre for the K-Means approach. Its correctness, overall speedup proportion, and grouping efficiency have already been vastly enhanced compared to both approaches used before improvement[26].

They examine two examples of calculating dense similarity matrices to cluster using vast data sets. By sparsifying a vector, contrast one technique using the Nyström methodology. Next, decide to sparsify the matrices while keeping their closest neighbors, plus we look into their parallelization. On dispersed computers, we parallelize both memory usage and computations. We demonstrate the practicality of our parallel approach by conducting an empirical investigation on two colossal data sets: a large-scale photo dataset with 2,121,863 examples and a document data set with 193,844 examples[27].

The subsequent work has developed a probabilistic theory for grouping using a random point scientific process. This approach perfectly mimics Bayes decision theory regarding classification: Bayes clustering procedures available with the least predicted errors provided known underlying processes and a defined objective function. As a result, clustering becomes an ongoing method rather than a subjective one. In this study, we start to comprehend such an algorithm by presenting the circumstances where the given opportunity used in traditional K-Means clustering becomes efficient in the novel Bayes clustering concept[28].

The description of methodological improvements may allow for high computation savings in average squared data clustering. A parallel statistics program P-CLUSTER, which runs on a desktop system, now includes additional upgrades.Unsupervised categorization of photos with a familiar texture was the subject of investigations. A 96 decreasing trend in the calculation was made for some data sources[29].

The K-Means technique is suggested in this study as a single special instruction multiple data (SIMD) architecture processors (GPUs) driven algorithm. Both objects and the reconfiguration of k-centroids are assigned to the GPU concurrently inside this approach to minimize the computationally costly parts of the entire unit. With the most current development of GPUs with computational integrated system architectures, they have built such Graphics K-Means algorithm (CUDA). The numerical tests showed that the efficiency of GPU-based K-Means might be up to 40 times quicker than that of CPU-based K-Means[30]. Furthermore, for this research, we embarked on an in-depth exploration of many existing studies including [32], [33], [34], [35], [36], [37], employing a comprehensive approach that encompassed [research methodologies/tools].

IV. Parallelization Strategies

The parallel method for data mining uses a parallel architecture. Moreover, for implementation purpose parallel algorithm is used. A standard parallel clustering procedure includes at least the three phases listed. The first phase is Partition. What happens is like during this phase, we will divide the information into smaller datasets. The second phase is Computing. Here we run the clustering method on each processor's local dataset. Each processor's clustering method may be different.

The third phase is Integration. To get the overall result, we have merged all the information, which are clusters obtained from each processor. There are two data techniques which are as follows. Finding reasons for the information set's highly changeable components, or locating and explaining outliers, is the aim in several applications. Other applications aim to comprehend the variances of the overwhelming bulk of the given dataset components without any concern for the extremes. The first type of data mining is primarily used in science, while the second type is business applications.

In first-class applications, it appears that Computing is necessary. Because we are still determining how beneficial sampling from a vast dataset will be in implementations of the latter type. Thus, parallel Computing has a big future as a platform for data processing. However, it has yet to be apparent if this is the direction data mining will take. We can understand how the technique is already employed by looking at the applications in which the most extensive data centers are used for data gathering.

The parallelization of data involves a variety of techniques. The first method is data parallelism, and the aim is to distribute the data appropriately and divide it into processors, each of which computes the given data in parallel. Data parallelisms are classified into two types: record-based and attribute-based.

The second method is the parallelization of tasks. Each record's likelihood of performing a data mining strategy is the same when redistributed, making equally distributing information achievable. Data partitioning based on records is commonly used in real-world parallel data mining applications. If the characteristic Partition is incorrect, the information connection breaks, and the processing quality decreases.

The second strategy is Task Parallel. It breaks the entire solution method into many sub-procedures and sends them to separate processors. It also includes two ways for achieving task parallelism. One uses a divide-and-conquer technique, assigning the work and the subtask to a designated processor. The other is based on a scheduler, which dynamically assigns data to available processors.

The execution method involves utilizing many CPUs to perform the same job and several CPUs to complete a separate one. The one picked is determined based on the area we are implementing and the data structure. Regardless of the parallel method, the load-balancing problem must be addressed. If data is misallocated, the burden may be unequal.

The third alternative is a hybrid; they need help resolving the issue. Upon properly splitting the data and spreading it among each processing unit, jobs are assigned to a designated processor in this approach, according to the data properties assigned to each processing unit.

After properly breaking down information and transmitting data to each processing, tasks are assigned to a designated processor by the attributes assigned to each processing unit. The data generated by the chain is combined when each job executing on each processor is finished. Task and data parallel techniques are used in this situation, and data interchange occurs before and after job execution, which is advantageous.

Algorithm 1 K-Means Algorithm

K is cluster
N is data objects
clusters[K] cluster centers in array
objects[N] data objects in array
membership[N] objects memberships in array
Set threshold Θ
repeat
  $G \leftarrow 0$
  for $a = 0$ to $N - 1$ **do**
    for $b = 0$ to $K - 1$ **do**
      $distances \leftarrow objects[a] - clusters[b]$
      if $distances < d_{\min}$ **then**
        $d_{\min} \leftarrow distances$
        $n \leftarrow b$
      end if
    end for
    if $membership[a] \neq n$ **then**
      $G \leftarrow G + 1$
      $membership[a] \leftarrow n$
    end if
    $newcluster[n] \leftarrow newcluster[n] + objects[a]$
    $newclustersize[n] \leftarrow newclustersize[n] + 1$
  end for
  for $b = 0$ to $K - 1$ **do**
    $clusters[b][*] \leftarrow newcluster[b][*]/newclustersize[b]$
    $newcluster[b][*] \leftarrow 0$
    $newclustersize[b] \leftarrow 0$
  end for
until $G/N \leq \Theta$

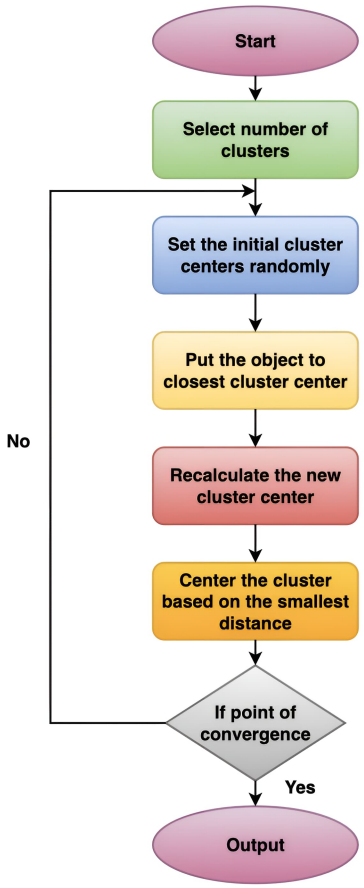

Fig. 6. Flowchart of the of K-Means Algorithm.

In addition to minimizing transmission costs, the job is not interrupted.

V. Methodology

This section discusses a study focused on improving the quality of the information in a country data set using K-Means, Minibatch K-Means, and Fuzzy C-Means methods. The study uses parallelization techniques to speed up the execution times of all methods and obtain more accurate results. With the help of the elbow method, the optimal number of clusters is perceived using this method.

The study discusses different techniques for parallelizing: sequential K-Means, Minibatch K-Means, and fuzzy C-Means, parallelization using MPI (Message Passing Interface), and a cloud-based method. These techniques are implemented from scratch in Python, a popular programming language for machine learning and data processing. The results of this study may be helpful for organizations interested in aiding impoverished countries, as it provides more accurate information about the country data set.

A. Sequential Process of K-Means

Calculating centroids and cluster formation is done progressively with sequential K-means. Here is a detailed guide and breakdown of how K-Means works.

The code is a Python implementation of the K-Means clustering technique. The objective of the method is to divide a collection of N objects into K clusters according to how identical they are, where K is the number of clusters Algorithm 1.

The K-Means algorithm uses this code to cluster data. It initializes the cluster centers and places data objects into clusters. Then, it refines the cluster centers and data object membership iteratively until reaching a threshold value that terminates the algorithm. To manage data, it uses variables to handle cluster centers, data objects, and membership information. Cluster centers are an array of size K, while data objects and membership data are arrays of size N.

The code computes the distances between objects and each cluster center during each iteration, assigning objects to the nearest cluster based on these calculations. If the object is assigned to a new cluster, a variable counter increments to reflect changes in its membership data. The new cluster center updates by summing its data object values and dividing by the cluster size.

Finally, the algorithm checks if the termination threshold is met. If not, it updates the cluster centers and repeats the process.

Algorithm 2 Minibatch K-Means

Begin
Initialize: k, mini-batch size b, iterations k, data collection X
Set: Every $s \in C$ with an x drawn at random to X
$a \leftarrow 0$
for $j \leftarrow 1$ to k do
  $M \leftarrow b$ examples picked randomly from X
  for $x \in M$ do
    $d[x] \leftarrow f(C, x)$
  end for
  for $x \in M$ do
    $s \leftarrow d[x]$
    $a[s] \leftarrow a[s] + 1$
    $n \leftarrow 1/a[s]$
    $s \leftarrow (1 - n)s + nx$
  end for
end for
Stop

Algorithm 3 Fuzzy C-Means Algorithm

Initialize $U = [u_{ij}]$ matrix, $U^{(0)}$
At k-step: calculate the center vectors $C^{(k)} = [c_j]$ with $U^{(K)}$
$C_i = \frac{\sum_{j=1}^{n} u_{ij}^m x_j}{\sum_{j=1}^{n} u_{ij}^m}$
Update $U(k)$, $U(k+1)$
$u_{ij} = \frac{1}{\sum_{k=1}^{c} (\frac{\|x_i - c_j\|}{\|x_i - c_k\|})^{2/(m-1)}}$
$\|U^{(k=1)} - U^{(k)}\| < \epsilon$ then STOP
Else
Return to Step 2;

B. Minibatch K-Means

Here is an explanation of the code, step by step Algorithm 2:

For each point x in dataset X, a centroid s is initialized randomly from the set of possible centroids C. The algorithm then runs for k iterations, each with the following steps.

- A mini-batch M of b examples is randomly sampled from the dataset X.
- For each example x in the mini-batch M, the distance between x and each centroid in C is computed using the function f.
- For each example x in the mini-batch M, the closest centroid s is identified based on the computed distances, and the count of examples assigned to that centroid a[s] is incremented.
- For each example x in the mini-batch M, the position of the assigned centroid s is updated using a weighted average of its previous position and the new example x. The weight given to the previous position is determined by the number of examples already assigned to that centroid, i.e., s ← (1-n)s+nx, where n = 1/a[s].
- At the end of k iterations, the final positions of the centroids are returned.

The main idea behind this code is to iteratively improve the positions of the centroids by adjusting their position based on the examples in each mini-batch. By randomly sampling the examples and updating the centroids incrementally, the algorithm can scale to large datasets while producing high-quality clustering results.

C. Fuzzy C-Means

Here is an explanation of the code, step by step Algorithm 3:

- To perform Fuzzy C-Means clustering, we first initialize the membership matrix, U, by assigning random values to it or by setting it to a predefined initial value.
- At each iteration, denoted by the k-step, we calculate the center vectors of the clusters, representing the centroids of the clusters. The center vectors are calculated using the membership matrix U from the previous iteration.
- The center vectors are calculated using the membership matrix U from the previous iteration. Specifically, we calculate the center vectors, Ci, by taking a weighted average of the data points, xj, according to their membership values, uij.
- A higher membership value for a data point implies a more prominent weight for that data point. Once we have calculated the center vectors, we update the membership matrix for the next iteration by recalculating the membership values, uij, for each data point, xi, based on their distance from the center vectors, cj.
- The new membership value, uij, depends on the sum of the distances from the data point xi to all the center vectors, cj, raised to the power of 2/(m-1), where m is a fuzziness parameter that controls the level of fuzziness or overlapping among clusters.
- We then check if the difference between the membership matrices at the k-step and the (k+1)-step is lower than a certain threshold, epsilon(e). If the difference is smaller than epsilon, the algorithm terminates and returns the final membership matrix and center vectors.
- Otherwise, the algorithm returns to step 2 to recalculate the center vectors and update the membership matrix for the next iteration. The algorithm repeats steps 2 to 7 until convergence, which implies that the difference between the membership matrices at two consecutive iterations is less than or equal to epsilon.

D. Parallel Techniques using Message Passing Interface

MPI is one of the best methods for implementing parallel computing. The apparent reason for that is because of the standard libraries it has. The motivation is to aim for exactness and efficacy. With the help of open MPI, we can parallelize the K-Means, Minibatch K-Means, and Fuzzy C-Means. MPI is supported in many languages. In this, the datasets were divided equally among many of the MPI processes, and Using OpenMP instructions, data labeling for each local data within the process was carried out.

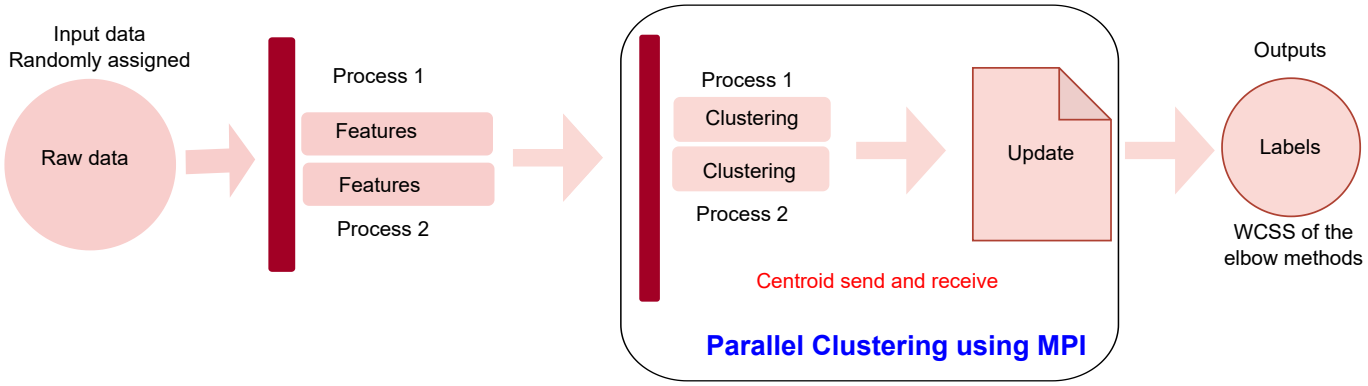

Fig. 7. A description of how to put the divide and combine plan into practice of Parallel clustering techniques using MPI.

Each attribute has its dimension in the result and is created in the space based on the number of attributes. The result generated using this methodology will be assimilated with the basic version of the technology to determine whether the approach is superior and how much performance measurements, such as time for completion, are generated. This method was evaluated on the same dataset, termed ”Country Data Set.”The information we will validate is based on the testing dataset, and we broke that complete information into train and test for training and testing the model. Here we used eighty for training and twenty for testing, i.e., 80:20.

To implement this, we use Jupiter Notebook. Where the ”divide and combine” scheme for parallel K-Means, Minibatch K-Means, and Fuzzy C-Means clustering using the Message Passing Interface (MPI) library, the process can be divided into several steps. Firstly, the data points should be divided into smaller subsets that different processes can independently process. Each process would then compute the centroid of a subset of data points using all algorithms until it converges to a stable value. After that, the partial centroids obtained from each process must be combined to obtain the final cluster of centroids. This can be achieved using the MPI reduce() function, which combines the partial centroids to ensure that all processes receive the same result Figure 7.

Finally, the results should be verified by comparing the final centroids with the expected values or by measuring the clustering quality using a performance metric such as the Sum of Squared Errors (SSE). The divide and combine scheme in parallel clustering using MPI provides an efficient way to process large datasets by dividing them into smaller subsets and processing them in parallel, resulting in faster execution times and increased scalability.

1) Steps involved in Parallel Techniques using MPI: The algorithm starts by choosing the number of clusters, k, and giving each node one of the k cluster centers. It is crucial to remember that the clusters must be distributed evenly among the nodes, i.e., k/n, where n is the total number of nodes.

After assigning the cluster centers, each node actively computes the mean centers of the k/n groups, disseminates the centers to all nodes using MPI (Message Passing Interface), receives the cores from other nodes, computes the distance between each data point and its nearest collection. When

Fig. 8. Flow of Parallel Techniques using MPI.

the cluster assignations stop changing, it actively achieves convergence. Finally, the process is actively repeated until reaching convergence.

A crucial element of the clustering algorithms is convergence. It ensures the algorithms terminate once the cluster assignments have stabilized and no more adjustments are required. This enhances the clustering process' effectiveness and precision. In conclusion, a clustering algorithm is a valuable tool for grouping data into groups, and convergence is essential to the algorithm's precision and effectiveness.

2) MPI methodology consists of several stages:

- We first determine the centroid based on the number of nodes.
- At this point, the master component sends data chunks based on centroid to various nodes. These broadcasts are

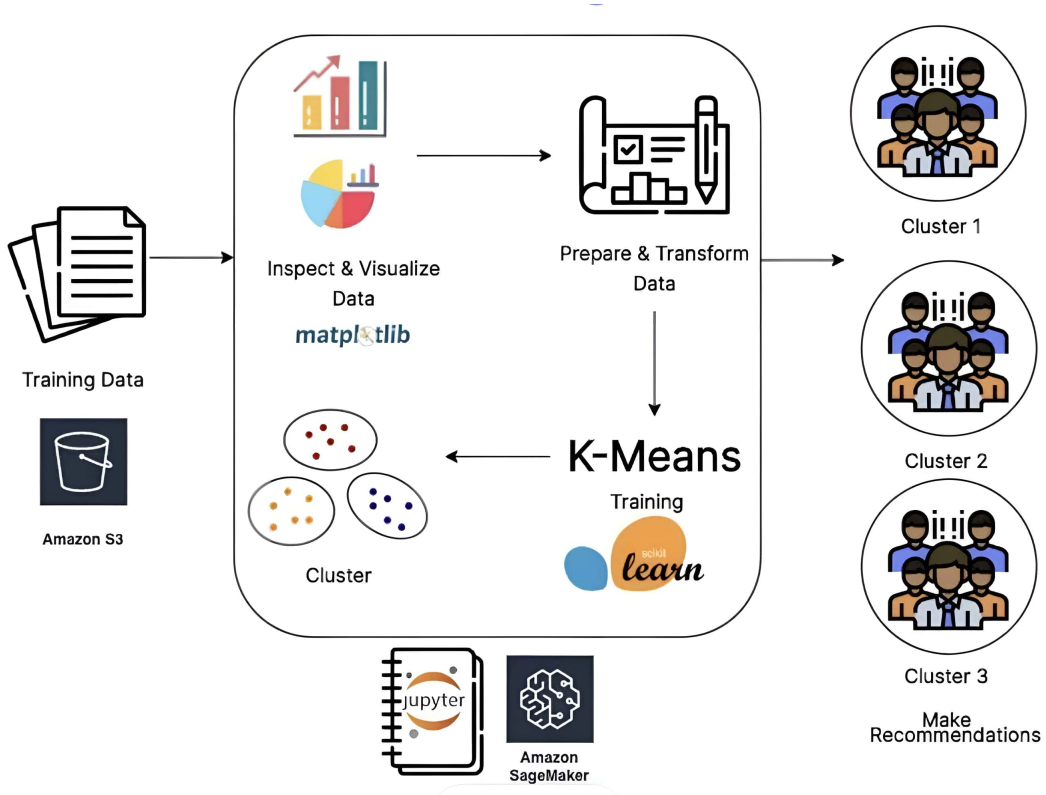

Fig. 9. How K-Means works in Amazon Sage Maker.

made possible by comm = MPI.COMM WORLD. And communication bias.

- The means of each node is used to calculate centroids, which are then broadcast to other nodes.
- Data points are dispersed using communication scatter. This process is repeated until convergence occurs.

E. Amazon Sage Maker

Amazon SageMaker is a cloud-based machine learning platform that enables developers to build, train, and deploy machine learning models. It offers a range of tools for data labeling, training, experimentation, model deployment, and monitoring. Amazon SageMaker uses a pay-per-use pricing model, which makes it more cost effective for small-scale projects and large-scale enterprises.

1) Working of parallel clustering techniques using MPI in AWS: Parallel clustering techniques using MPI in AWS SageMaker work by distributing the clustering algorithms across multiple compute nodes using the Message Passing Interface (MPI) protocol.

The input data is divided into various subsets. Each subset is given its own compute node when a training job is started in SageMaker using the clustering algorithms and MPI. Then, using MPI, the clustering algorithms are applied parallel to each subset to process the data across numerous nodes.

During the parallel processing, each computed node calculates its own local centroids and assigns data points to the nearest centroid. Then, the intermediate results are merged to obtain the final clustering solution.

The results are typically stored in an S3 bucket, which can be accessed by other AWS services or downloaded for further analysis. Using SageMaker for parallel clustering with MPI allows for the efficient processing of large datasets and significantly reduces the time required for clustering tasks.

Overall, parallel clustering using MPI in AWS SageMaker enables clustering in the context of distributed computing of big datasets and provides a scalable and efficient way to process data using multiple compute nodes.

A managed service machine learning (ML) Amazon's Elastic Compute Cloud (Amazon EC2) notebook example, which continues to run Jupiter Software, is an Amazon Sage Maker notebook example. This same notebook example is used to build and manage Jupiter notebooks for information extraction, training, and deploying machine learning models.

To begin the analysis, an Exploratory Data Analysis (EDA) of the chosen datasets will be conducted, followed by the training and creation of a model using AWS Sagemaker's built-in version of clustering. The resulting recommendation engine can then be deployed using an AWS Sagemaker inference endpoint and integrated into an application via an API. These steps were performed within an AWS Sagemaker-hosted Jupyter Notebook instance running on an ml.t2.xlarge instance.

2) Steps to generate a Sage Maker notebook example:

- Go to https://console.aws.amazon.com/sagemaker/ to access an Amazon Sage Maker console.
- Select Notebook cases, then start creating notebook examples.
- I will provide additional data just on the Generate Note-

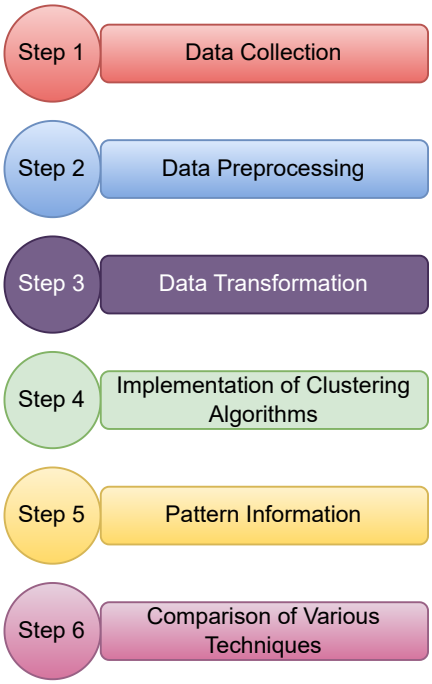

Fig. 10. Implementation Process.

book example site (when the ground is not noted, start leaving the default settings).

- Inside the Notebook example input box, enter an identity for one's notebook example. Select item ml.t2.medium as the Notebook Test case. It is the cheapest example form supported by notebook cases and is adequate for this workout. If the ml.t2.medium applies differently and is not accessible in one existing AWS Environment, select ml.t3.medium.
- Select a console category to generate the notebook example in Console Identification. This console form chooses the operating system and the Jupiter edition in which one's notebook instance is formed—View Amazon Linux 2 vs. Amazon Linux notebook cases for more data on application framework identification kinds. View Jupiter version control for more data.
- Add a new position for IAM, then start creating a role. The above IAM position is granted access to every S3 bucket with the word Sage maker. Such authorizations are obtained via the Amazon Sage Maker Full Access strategy, which Sage Maker associates with the position.
- Sage Maker starts an ML compute example in the above scenario, a notebooks example in a matter of seconds, and connects a 5 GB Amazon EBS storage capacity to it. This same netbook instance includes a Jupiter notebook domain controller that has been preselected, Sage Maker, AWS SDK library services, and a set of Anaconda library services.
- Generate a Notebook Example for further data on generating a Sage Maker netbooks example.

F. Implementation

1) Data Collection: We utilized Python programming language and machine learning techniques to implement algorithms on the KAGGLE country dataset, which consists of nine key factors. From those nine key factors, we use only four key factors to predict what organizations can use to determine financial assistance to nations. These nine attributes include Child Mortality, Exports, Health, Imports, Income, GDP, Inflation, Life Expectancy, and Total Fertility.

To perform the analysis, we used an Apple M1 processor with a memory of 16GB. Our primary objective was to use these algorithms to identify countries that may require financial aid based on their performance in these four factors.

2) Data Pre-Processing: First, the code sorts the DataFrame data by the values in the gdpp column in ascending order using the sort values function and then resets the index of the sorted DataFrame using the reset index function with the argument drop=True. This creates a new DataFrame with the same data as data but with the rows sorted by GDP per capital in ascending order and a new index.

Then, the code creates a scatter plot using the plot scatter function with the x values as the index of the sorted DataFrame and the y values as the gdpp column of the sorted DataFrame. The plot xlabel, plot ylabel, and plot title functions are used to add axis labels and a title to the plot, and plot show is used to display the plot.

Next, the code computes the quartiles of the 'gdpp' column using the quantile() function with the argument [0, 0.33, 0.67, 1] to get the lower quartile (0-33), middle quartile (33-67), and upper quartile (67-100) values. After that, the code defines the category labels as ['Low', 'Medium', 'High'].

Then, the code adds a new column to the DataFrame called GDP Category using the pd cut function with the arguments bins=quantiles and labels=categories. This function creates categories for the gdpp values based on the quantiles computed earlier and assigns the category label to each gdpp value in the new GDP Category column.

Finally, the code drops the columns health, life expec, total fer, and 'child mort from the DataFrame using the drop function with the arguments columns, inplace=True, and axis=1. This removes the specified columns from the DataFrame, without creating a new DataFrame.

In summary, this code performs data cleaning and visualization tasks on a DataFrame that contains information about various countries. It sorts the DataFrame by GDP per capital, creates a scatter plot of the GDP per capital values, categorizes the GDP per capital values into low, medium, and high categories based on quartiles, and drops some columns from the DataFrame that are not needed for further analysis.

3) Data Transformation: Since data transformation entails transforming raw data into a similar and standardized style, it is essential to clustering. Data transformation's primary goal is to reduce the influence of different scales, ranges, and skewness in the variables because these factors can significantly affect the results of clustering. Normalization, standardization, PCA transformation, and log transformation are a few examples of the data transformation methods frequently used in clustering. The best data conversion technique must be

chosen depending on the issue domain and the data's nature. If the appropriate data transformation methods are applied, clustering may yield precise and insightful results.

4) Implementation of Clustering Algorithms:

1) Sequential K-Means: The code provided reads a processed data file containing information about different countries. It initializes an empty list called sse to store the sum of squared errors (SSE) for different values of k. Then it iterates through the range of k values from 2 to 6. A K-Means object is initialized with the current value of k, and the algorithm is fitted to the dataset. The SSE for the fitted model is then appended to the sse list. The next block of code selects the input column for the K-Means technique and plots the SSE for different values of k. The plot shows that the optimal number of clusters is likely 4, as the SSE begins to level off at that point.
   The code initializes a K-Means object with k=4 and fits it to the dataset. The appropriate prediction method is then used to predict the cluster assignments for each data point. These predicted cluster assignments are then added as a new column to the DataFrame.Finally, the code prints out the centroid coordinates for each cluster. In summary, the K-Means technique clusters similar data points into k clusters. The code provided reads a file of processed data, calculates the SSE for different values of k, and selects the optimal number of clusters based on the plot. It then predicts cluster assignments for each data point and adds them to the DataFrame. Finally, it prints the centroid coordinates for each cluster.
2) Sequential Minibatch K-Means: This code performs clustering on a dataset using the Mini Batch K-Means algorithm. The dataset is loaded from a CSV file, and the columns for clustering are specified. The range of k values to try is defined, and an empty list is initialized to store the sum of squared errors (sse).
   A loop is then executed over each k value in the defined range. A Mini Batch K-Means model is initialized with the current k value and fitted to the dataset in each iteration. The inertia attribute of the model is then appended to the sse list. This indicates how well the data is being clustered, with a lower sum of squared errors indicating a better fit.
   Once the loop completes, the results are plotted as a line graph with k on the x-axis and the sum of squared errors on the y-axis. This plot can be used to identify the optimal number of clusters. This code sets the number of clusters to 4, and the model is fitted to the dataset using the prediction method.
   The predicted clusters and the original dataset are then plotted as a scatter plot, with each point colored according to its cluster assignment. The cluster centers are also plotted as red circles. The resulting plot shows how the data has been partitioned into distinct clusters based on the specified features.
3) Sequential Fuzzy C-Means: This code performs fuzzy c-means clustering on a dataset of country data. First, the data is loaded from a CSV file, and non-numeric columns are removed. The remaining numeric columns are then normalized using StandardScaler.
   The number of clusters is set to 4, and the fuzzy exponent parameter (m) and the maximum number of iterations are also defined. The fuzzy c-means clustering algorithm is then initialized with the normalized data, number of clusters, m, and maxiter values.
   The algorithm is run, and the predicted cluster membership for each data point is obtained by taking the argmax of the u matrix. This assigns each data point to a cluster based on the degree of membership in each cluster.
   The predicted clusters are added to the original dataframe, and the number of countries in each cluster is printed to the console. This indicates how well the data has been clustered and can help with further analysis or decision-making. Fuzzy c-means clustering is functional when data points can belong to multiple clusters simultaneously, allowing for more nuanced clustering than traditional k-means clustering.
4) Parallel K-Means Minibatch K-Means and Fuzzy C-Means Using MPI: This code is an implementation of the K-means clustering algorithm using MPI(Message Passing Interface) for parallelization. The code is divided into two parts: the master part (rank 0) and the worker part (other ranks).
   In the master part, the code reads a CSV file containing data and initializes the number of clusters (num clusters) and other necessary variables. The data is divided into chunks, and each chunk is sent to a worker using MPI's scatter operation. The initial centroids are also broadcasted to all workers.
   In the worker part, each worker receives its chunk of data and calculates the distance between each data point and the centroids. It assigns each data point to the cluster with the closest centroid. It then counts the number of samples in each cluster and sends the count to the master using MPI's gather operation. The worker also receives the updated centroids from the master using MPI's all reduce operation.
   This process of calculating distances, updating clusters, and centroids is repeated until the centroids no longer change. The final cluster assignments from all workers are gathered at the master, and the adjusted Rand score is calculated to compare the results with the labels obtained from running KMeans from scikit-learn (kmeans).
   Finally, the code visualizes the clusters and centroids using matplotlib. It also performs some analysis, such as printing the frequency of clusters for different GDP categories, replacing cluster labels to match the original labels, computing a confusion matrix, and printing classification metrics.
   Overall, this code demonstrates parallelizing the K-means algorithm using MPI and compares the results with scikit-learn's KMeans implementation. It also provides visualizations and analysis of the clustering results. We will replace the KMeans function with MiniBatchK-Means and FuzzyCMeans and run the same code.
5) Parallel K-Means, Minibatch K-Means, and Fuzzy C-

Means using MPI in Cloud: As discussed above, we use the same code to run in AWS SageMaker, which employs parallel processing with MPI and all algorithms to cluster data.

5) Pattern Information: In Clustering, data points are grouped according to how alike they are using pattern information. The method finds patterns in the data by repeatedly assigning each data point to the closest centroid and adjusting the centroid by averaging all the data points given to it. Up until the centroids stop moving or the maximum number of repetitions has been achieved, this procedure is continued. The ensuing clusters show collections of data points with comparable patterns or traits. By comparing new data points to existing clusters, these clusters may be utilized to predict future data points and understand the underlying patterns within the data.

6) Comparison of various Techniques: We will compare all three methods of execution times, including in terms of execution speed, accuracy, precision, recall, and f1 slope and support.

G. Setup of Exploration

We tested and implemented all algorithms using Jupyter Notebook on my computer, and we also utilized Amazon SageMaker Jupiter Notebook for cloud services. We used a computer with an Apple M1 processor and 16 GB of RAM to run our algorithms during testing. To supplement our methodology, we used a KAGGLE dataset that contained 167 rows and ten columns, totaling 13.2 kb.

We utilized machine learning techniques and various Python libraries to implement our methodology. Our approach involved thorough data analysis and preprocessing, then implementing and testing various algorithms on the cleansed data. We then analyzed the results to determine the best approach for accurately predicting financial aid requirements for different nations.

VI. Result and Conclusion

A. Elbow Method

The Elbow method is a practical approach for determining the optimal no of clusters in a model. Applying this method makes it possible to identify the most appropriate number of distinct clusters for a given dataset Figure 11.

We can create a new model with the optimal number of clusters identified, which is 4, and then examine the cluster assignments for each observation. Essentially, we would assign each data point to one of the 4 clusters established on their characteristics or similarities. This process can provide valuable insights into patterns and relationships within the data, potentially aiding in making informed decisions or predictions and the graphs below.

The structure of the clusters is displayed in the graph, and it includes the following algorithms Fig. 15 16 17 18 19 20 21 22 23

1. Sequential K-Means.
2. Parallel K-Means using MPI.
3. Parallel K-Means using cloud.

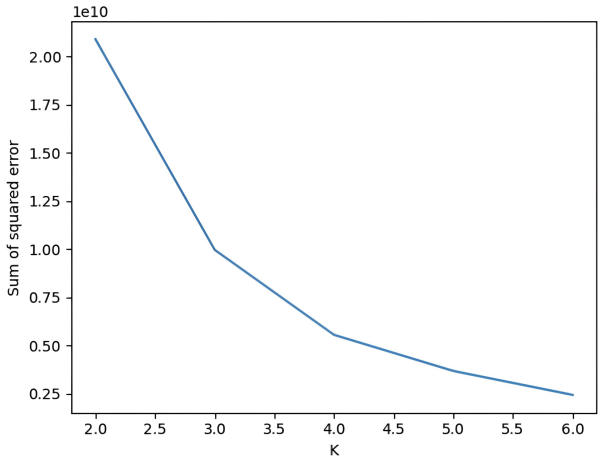

Fig. 11. Distortion Score Elbow Curve for Clustering.

4. Sequential Minibatch K-Means.
5. Parallel Minibatch K-Means using MPI.
6. Parallel Minibatch K-Means using Cloud.
7. Sequential Fuzzy C-Means.
8. Parallel Fuzzy C-Means using MPI.
9. Parallel Fuzzy C-Means using Cloud.

The confusion matrix is a tool that helps to summarize the performance of a model by displaying the number of correct and incorrect predictions made for each class in the dataset. It consists of three columns and three rows, corresponding to the three classes in our data. The true values are represented on the y-axis while the predicted values are represented on the x-axis. Using the values obtained from the confusion matrix, precision, recall and F1 scores can be calculated for the model. The confusion matrices for three different clustering algorithms; Sequential K-Means Figure 12, Sequential Minibatch K-Means Figure 13 and Sequential Fuzzy C-Means Figure 14.

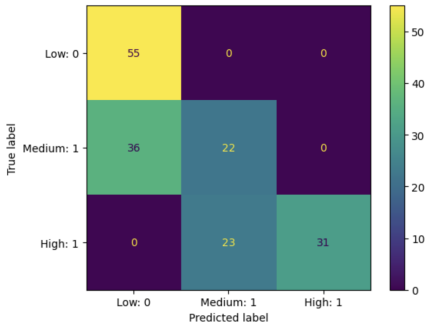

Fig. 12. Confusion matrix of Sequential K-Means.

TABLE I
Evaluation Matrix of Sequential Clustering

Sequential	Accuracy	precision	recall	F1-score
K-Means	0.45	0.68	0.66	0.67
Minibatch K-Means	0.48	0.75	0.72	0.71
Fuzzy C-Means	0.46	0.69	0.65	0.63

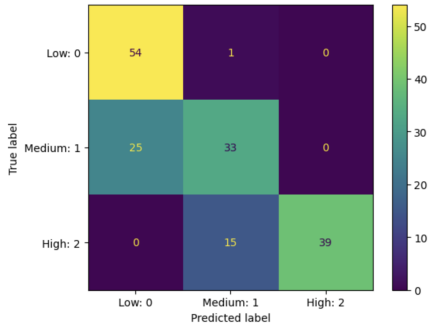

Fig. 13. Confusion matrix of Sequential Minibatch K-Means.

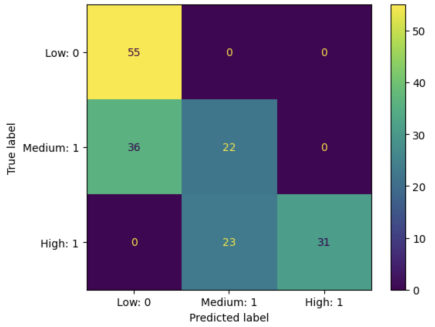

Fig. 14. Confusion matrix of Sequential Fuzzy C-Means.

The table shows that Minibatch K-Means performs the best out of the three algorithms in all four metrics. It has higher sequential accuracy, precision, recall, and F1-score than K-Means and Fuzzy C-Means. This indicates that Minibatch K-Means is the most effective algorithm in accurately grouping the data points into clusters Table I. Also, we presented a graphical comparison representation of Figure 24.

TABLE II
EVALUATION MATRIX OF PARALLEL CLUSTERING USING MPI

Parallel using MPI	Accuracy	precision	recall	F1-score
K-Means	0.47	0.75	0.72	0.71
Minibatch K-Means	0.46	0.69	0.65	0.66
Fuzzy C-Means	0.45	0.67	0.66	0.67

The table shows that when implementing these clustering algorithms using MPI parallelism, K-Means performs slightly better than Mini-Batch K-Means and Fuzzy C-Means in terms of accuracy, precision, recall, and F1-score. It achieves higher values in most of the metrics, indicating better overall performance. However, as always, it's important to consider the specific dataset and problem context when choosing the appropriate algorithm and parallelization strategy. Table II. Also, we presented a graphical comparison representation of Figure 25.

The table shows that all three algorithms when implemented on Amazon Sage Maker. K-Means has the highest accuracy, followed by Fuzzy C-Means and Minibatch K-Means, It achieves higher values in most of the metrics, indicating better overall performance Table III. Also we presented graphical comparison representation Figure 26.

TABLE III
EVALUATION MATRIX OF PARALLEL CLUSTERING USING MPI IN AMAZON SAGE MAKER

Amazon Sage Maker	Accuracy	precision	recall	F1-score
K-Means	0.48	0.74	0.71	0.70
Minibatch K-Means	0.45	0.68	0.66	0.67
Fuzzy C-Means	0.46	0.70	0.69	0.68

Finally, by comparing the performances of three clustering algorithms K-Means, Fuzzy C-Means, and Minibatch K-Means we can see that the sequential implementation using these algorithms outperforms the parallel implementation using MPI or Amazon Sage Maker. Among the three algorithms, Minibatch K-Means has shown better results compared to K-Means and Fuzzy C-Means, having the highest accuracy, recall, precision, and F1-score values in the sequential implementation.

The table displays the execution times of three clustering algorithms: K-Means, Mini-Batch K-Means, and Fuzzy C-Means, for both sequential and parallel techniques using MPI and AWS Sage Maker, measured in milliseconds. Significantly, the parallel implementations using AWS Sage Maker exhibited greater efficiency when compared to the parallel implementations using MPI and the sequential implementations for all three algorithms. Notably, Fuzzy C-Means demonstrated the shortest execution time for both parallel implementations, highlighting its potential suitability for clustered data analysis, and the implementation of Fuzzy C-Means in parallel using AWS Sage Maker may therefore provide the most effective means of optimizing algorithm performance and scalability Table IV.

TABLE IV
COMPARING EXECUTION TIME OF ALL TECHNIQUES

Techniques	Exeution Times(ms)
Sequential K-Means	2.853
Parallel K-Means using MPI	1.050
Parallel K-Means using Cloud	0.828
Sequential Minibatch K-Means	2.310
Parallel Minibatch K-Means using MPI	1.215
Parallel Minibatch K-Means using Cloud	0.921
Sequential Fuzzy C-Means	1.285
Parallel Fuzzy C-Means using MPI	0.932
Parallel Fuzzy C-Means using Cloud	0.623

Overall Considering these findings, cloud computing platforms like AWS Sage Maker have the potential to offer substantial advantages for data processing in parallelized clusters. Also, you can see the graphical representation of Figure 27

B. Discussion

The independent features are imports, exports, health, and inflation. Based on these four features, clusters are formed. We determine the model accuracy using the GDP feature from the dataset and the clusters. The GDP values are categorized as low, medium, and high. Based on the GDP category, we determine whether a country requires financial assistance or

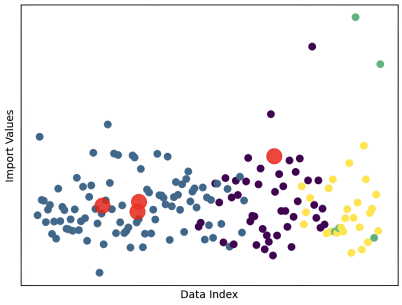

Fig. 15. Sequential K-Means

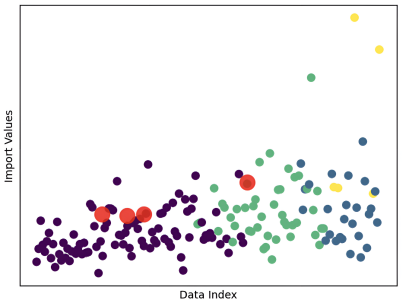

Fig. 16. Parallel K-Means using MPI

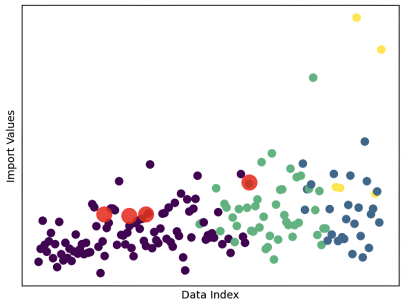

Fig. 17. Parallel K-Means using Cloud

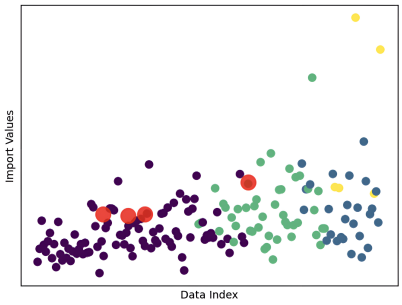

Fig. 18. Sequential Minibatch K-Means

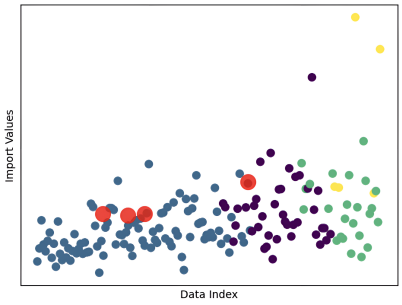

Fig. 19. Parallel Minibatch K-Means using MPI

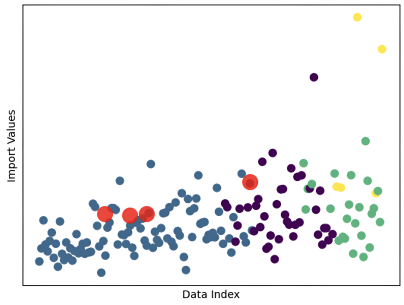

Fig. 20. Parallel Minibatch K-Means using Cloud

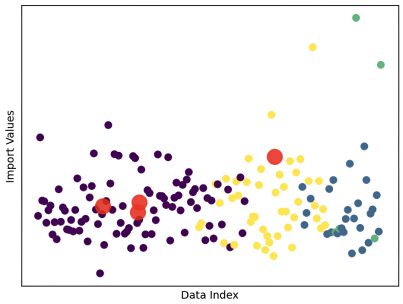

Fig. 21. Sequential Fuzzy C-Means

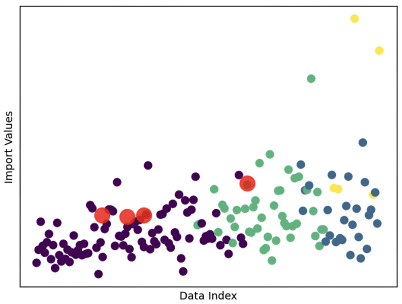

Fig. 22. Parallel Fuzzy C-Means using MPI

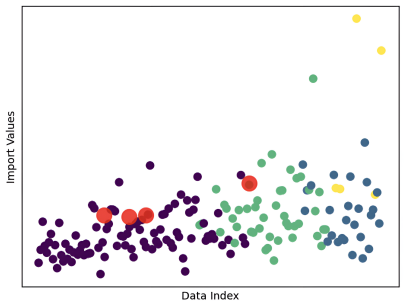

Fig. 23. Parallel Fuzzy C-Means using Cloud

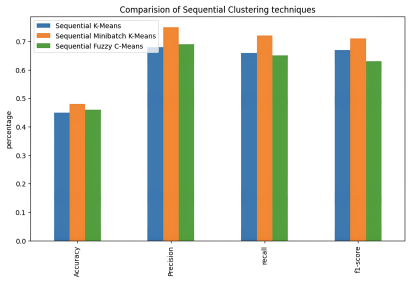

Fig. 24. Comparison of Sequential Clustering techniques

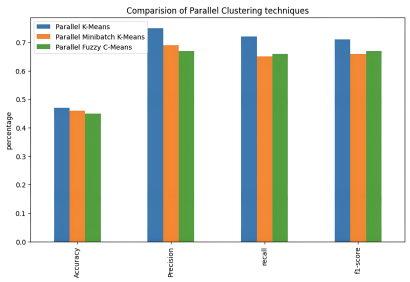

Fig. 25. Comparison of Parallel Clustering techniques

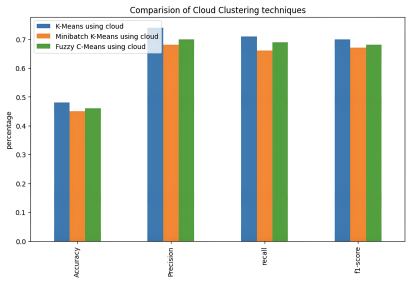

Fig. 26. Comparison of Cloud Clustering techniques

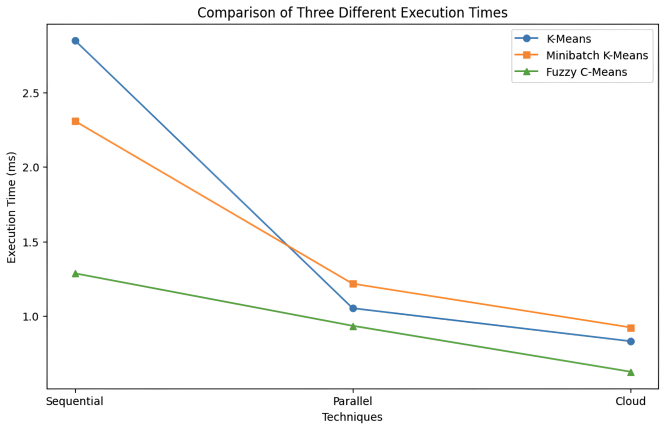

Fig. 27. Comparison of Execution times in all techniques.

not. If the GDP is categorized as low, it signifies that the country needs help. For medium GDP, the country may or may not require assistance, and for high GDP, the country does not require help.

From Figure 28, the green color indicates help is not required, the red color indicates help need, and the yellow might need help and shows the countries in need of assistance and those that may require assistance. As a result, an NGO or organizations can quickly analyze data and assist countries.

The utilization of parallel clustering techniques results in numerous advantages over sequential clustering approaches, including enhanced efficiency, flexibility, scalability, fault tolerance, resource utilization, and real-time capabilities. These benefits make parallel clustering techniques the preferred option for dealing with large-scale datasets and time-sensitive clustering tasks.

The managed and user-friendly environment provided by Amazon SageMaker is ideal for developing and deploying machine learning models that include clustering algorithms. It offers scalability, managed infrastructure, integration with AWS services, flexibility, monitoring capabilities, and cost-effectiveness advantages that make it a leading choice over manual setup and management of parallel clustering techniques that utilize MPI.

C. Conclusion

Machine learning (ML) is a field devoted to understanding and building methods that let machines leverage data to enhance the performance and accuracy of different applications. A subset of machine learning is closely related to data clustering, which focuses on grouping data samples into various groups for data prediction, detection, identification, etc.

Unsupervised ML algorithms are the most effective methods for data clustering. This work investigates various approaches and identifies factors influencing the algorithm's efficiency. Among them are the effects of the starting cluster on accuracy and the way the data is parallelized on efficiency. Finding the best approach based on the available data takes time and effort. We concentrated on country data and analyzed it using various parallel approaches and cloud computing. The analysis is based on four different country factors, and this analysis will assist various organizations that are prepared to assist the country.

The results demonstrate that implementing these models on a cloud platform significantly improves their performance in terms of execution time. By evaluating accuracy scores and confusion matrix values, we were able to compare the effectiveness of the three models. The findings indicated that when utilizing the sequential process, Minibatch K-Means achieves better performance compared to K-Means and Fuzzy C-Means. However, in cases where the parallel process was used, K-Means produced better results than the other two models.

D. Future Work

Our future work will concentrate on analyzing data using MapReduce and Multiprocessing, as well as developing an application that receives data as input and recommends the most efficient method for speedy execution. We aim to enhance the accuracy, precision, recall, and F1-score of our results. Additionally, we will experiment with different clustering algorithms for cluster initialization and construct an algorithm to assess the dataset and determine the most appropriate parallelization technique.

References

[1] S. Mohanavalli, S. Jaisakthi, and C. Aravindan, "Strategies for parallelizing k means data clustering algorithm," in Communications in Computer and Information Science, 2011, pp. 427–430.

[2] Y. Zhang, Z. Xiong, J. Mao, and L. Ou, "The study of parallel k-means algorithm," in 2006 6th World Congress on Intelligent Control and Automation, vol. 2, 2006, pp. 5868–5871.

[3] P. Swamy, M. Raghuwanshi, and A. Gholghate, "An improved approach for k-means using parallel processing," in 2015 International Conference on Computing Communication Control and Automation, 2015, pp. 358–361.

[4] A. Boukhdhir, O. Lachiheb, and M. S. Gouider, "An improved mapreduce design of kmeans for clustering very large datasets," in 2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA), 2015, pp. 1–6.

[5] M. Jaatun, G. Zhao, and C. Rong, "Parallel k-means clustering based on mapreduce," in Cloud Computing Lecture Notes in Computer Science, 2009, pp. 674–679.

[6] M. Baydoun, M. Dawi, and H. Ghaziri, "Enhanced parallel implementation of the k-means clustering algorithm," in 2016 3rd International Conference on Advances in Computational Tools for Engineering Applications (ACTEA), 2016, pp. 7–11.

[7] S. A. Ghamdi and G. D. Fatta, "Efficient parallel k-means on mapreduce using triangle inequality," in 2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech), 2017, pp. 985–992.

[8] S. Zhong, S. Lin, G. Xu, and K. Shi, "The expansibility research of k-means algorithm under the gpu," in 2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS), 2016, pp. 734–737.

[9] J. Tian, L. Zhu, S. Zhang, and L. Liu, "Improvement and parallelism of k-means clustering algorithm," Tsinghua Science and Technology, vol. 10, no. 3, pp. 277–281, 2005.

[10] E. Kijsipongse and S. U-ruekolan, "Dynamic load balancing on gpu clusters for large-scale k-means clustering," in 2012 Ninth International Conference on Computer Science and Software Engineering (JCSSE), 2012, pp. 346–350.

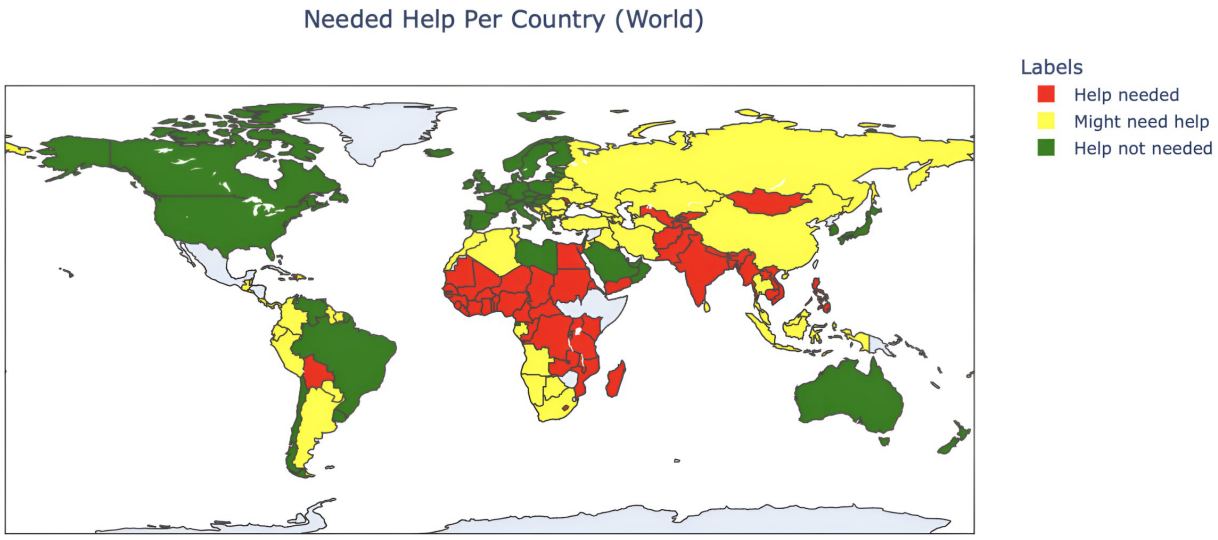

Fig. 28. Help needed per country (World).

[11] H. Fakhi, O. Bouattane, M. Youssfi, and O. Hassan, "New optimized gpu version of the k-means algorithm for large-sized image segmentation," in 2017 Intelligent Systems and Computer Vision (ISCV), 2017, pp. 1–6.

[12] C. M. Poteras, M. C. Mihaescu, and M. Mocanu, "An optimized version of the k-means clustering algorithm," in 2014 Federated Conference on Computer Science and Information Systems, 2014, pp. 695–699.

[13] Y. Jin and J. F. Jaja, "A high performance implementation of spectral clustering on cpu-gpu platforms," in 2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), 2016, pp. 825–834.

[14] F. Yi and I. Moon, "Extended k-means algorithm," in 2013 5th International Conference on Intelligent Human-Machine Systems and Cybernetics, vol. 2, 2013, pp. 263–266.

[15] S. Kapil, M. Chawla, and M. D. Ansari, "On k-means data clustering algorithm with genetic algorithm," in 2016 Fourth International Conference on Parallel, Distributed and Grid Computing (PDGC), 2016, pp. 202–206.

[16] J. Bhimani, M. Leeser, and N. Mi, "Accelerating k-means clustering with parallel implementations and gpu computing," in 2015 IEEE High Performance Extreme Computing Conference (HPEC), 2015, pp. 1–6.

[17] J. Wu and B. Hong, "An efficient k-means algorithm on cuda," in 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum, 2011, pp. 1740–1749.

[18] S. Kumari, N. Patil, P. Nankar, and M. Kulkarni, "Cuda parallel computing framework for stock market prediction using k-means clustering," in 2020 International Conference on Smart Electronics and Communication (ICOSEC), 2020, pp. 467–473.

[19] R. Taha, S. Alshakrani, and A. Alqaddoumi, "Implementing parallel computing to enhance the performance of k-mean algorithm," in 2021 International Conference on Data Analytics for Business and Industry (ICDABI), 2021, pp. 140–143.

[20] X. Shan, Y. Shen, and Y. Wang, "A parallel k-means algorithm for high dimensional text data," in 2018 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW), 2018, pp. 1–2.

[21] J. Zhang, G. Wu, X. Hu, S. Li, and S. Hao, "A parallel k-means clustering algorithm with mpi," in 2011 Fourth International Symposium on Parallel Architectures, Algorithms and Programming, 2011, pp. 60–64.

[22] S. Karbhari and S. Alawneh, "Gpu-based parallel implementation of k-means clustering algorithm for image segmentation," in 2018 IEEE International Conference on Electro/Information Technology (EIT), 2018, pp. 0052–0057.

[23] Ansari, Z. Afzal, A. Sardar, and T. H, "Data categorization using hadoop mapreduce-based parallel k-means clustering," in Journal of the Institution of Engineers, 2019, pp. 95–103.

[24] T. Ragunthar, P. A. N. Gopinath, and M. Subhashini, "A strong reinforcement parallel implementation of the k-means algorithm using message passing interface," in Materials Today: Proceedings, 2021, pp. 3799–3802.

[25] Z. Gao, E. Li, and Y. Jiang, "A gpu-based harmony k-means algorithm for document clustering," in IET International Conference on Information Science and Control Engineering 2012 (ICISCE 2012), 2012, pp. 1–4.

[26] H. Zhao, "Research on improvement and parallelization of k-means clustering algorithm," in 2021 IEEE 3rd International Conference on Frontiers Technology of Information and Computer (ICFTIC), 2021, pp. 57–61.

[27] W.-Y. Chen, Y. Song, H. Bai, C.-J. Lin, and E. Y. Chang, "Parallel spectral clustering in distributed systems," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 33, no. 3, pp. 568–586, 2011.

[28] L. A. Dalton, "On the optimality of k-means clustering," in 2013 IEEE International Workshop on Genomic Signal Processing and Statistics, 2013, pp. 70–71.

[29] D. Judd, P. McKinley, and A. Jain, "Large-scale parallel data clustering," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, no. 8, pp. 871–876, 1998.

[30] B. Hong-tao, H. Li-li, O. Dan-tong, L. Zhan-shan, and L. He, "K-means on commodity gpus with cuda," in 2009 WRI World Congress on Computer Science and Information Engineering, vol. 3, 2009, pp. 651–655.

[31] Zıa Ur Rahman, M., Vardhan, B.V., Jenith, L., Rakesh Reddy, V., Surekha, S., Srinivasareddy, P. (2022). Adaptive Exon Prediction Using Maximum Error Normalized Algorithms. In: Mathur, G., Bundele, M., Lalwani, M., Paprzycki, M. (eds) Proceedings of 2nd International Conference on Artificial Intelligence: Advances and Applications. Algorithms for Intelligent Systems. Springer, Singapore. https://doi.org/10.1007/978-981-16-6332-1 44.

[32] Adeyemo, V., Abdullah, A., JhanJhi, N., Supramaniam, M. & Balogun, A. Ensemble and deep-learning methods for two-class and multi-attack anomaly intrusion detection: an empirical study. *International Journal Of Advanced Computer Science And Applications*. **10** (2019)

[33] Ghosh, G., Verma, S., Jhanjhi, N., Talib, M. & Others Secure surveillance system using chaotic image encryption technique. *IOP Conference Series: Materials Science And Engineering*. **993**, 012062 (2020)

[34] Almusaylim, Z., Zaman, N. & Jung, L. Proposing a data privacy aware protocol for roadside accident video reporting service using 5G in Vehicular Cloud Networks Environment. *2018 4th International Conference On Computer And Information Sciences (ICCOINS)*. pp. 1-5 (2018)

[35] Shahid, H., Ashraf, H., Javed, H., Humayun, M., Jhanjhi, N. & AlZain, M. Energy optimised security against wormhole attack in iot-based wireless sensor networks. *Comput. Mater. Contin.* **68**, 1967-81 (2021)

[36] Sennan, S., Somula, R., Luhach, A., Deverajan, G., Alnumay, W., Jhanjhi, N., Ghosh, U. & Sharma, P. Energy efficient optimal parent selection based routing protocol for Internet of Things using firefly optimization algorithm. *Transactions On Emerging Telecommunications Technologies*. **32**, e4171 (2021)

[37] Hussain, S., Ahmed, U., Liaquat, H., Mir, S., Jhanjhi, N. & Humayun, M. IMIAD: intelligent malware identification for android platform. *2019 International Conference On Computer And Information Sciences (ICCIS)*. pp. 1-6 (2019)

Vishnu Vardhan Baligodugula (Student Member, IEEE) is a Graduate Research Assistant working under the supervision of Dr. Fathi Amsaad, an Assistant Professor of Computer Science and Engineering at Wright State University in Dayton, Ohio, USA. He received a Master of Science degree in Computer Science from Wright State University in Spring 2023. His research interests include Machine Learning, Parallel Programming, and Distributed Computing.

Fathi Amsaad (Senior Member, IEEE) is an Assistant Professor of Computer Science and Engineering at Wright State University, Dayton, Ohio, USA. He received the Bachelor's degree in Computer Science from the University of Benghazi, Libya, in 2002. He received a dual Master's degree in Computer Science/ Computer Engineering from the University of Bridgeport, CT, USA, in 2011/ 2012. He received a Ph.D. degree in Engineering with emphases in Computer Science and Engineering from the University of Toledo, OH, USA, in 2017. His research interest include AI Hardware Security, Machine Learning and Trusted AI. Both government and industry fund Dr. Amsaad's research including NSF, AFRL, AFOSR, Intel, NSA, and Ohio department of education. He has served as an Organizer, Program Chair, Technical Program Committee member, Gust Editor, and on the Reviewer Board for several international conferences and journals. In addition to his research activities, Dr. Amsaad has established teaching experience in hardware security, IoT and embedded systems security, distributed computing, digital systems, and network administration and security curriculum.

Vincent Schmidt is a Senior Research Computer Engineer at the United States Air Force, boasting a robust 19-year tenure in Dayton, Ohio. He holds a Doctor of Philosophy (PhD) in Computer Engineering from Wright State University (1996-2002), complemented by a Master of Science (MS) in Computer Engineering (1993-1995) and a Bachelor of Science (BS) in Computer Engineering (1986-1990) from The Ohio State University. Since joining in September 2004, he has played a pivotal role in diverse projects, demonstrating his commitment to advancing computer engineering. Dr. Schmidt's academic journey is marked by excellence. His wealth of experience and academic prowess positions Dr. Schmidt as a vital contributor to the field of Computer Engineering. His research interests encompass an array of domains, including data visualization, emergency management, pattern clustering, social networking, text analysis, content-based retrieval, data mining, geographic information systems, sensor fusion, software engineering, and more. Dr. Schmidt's unwavering dedication, coupled with his comprehensive background, underscores his significant impact on the advancement of computer engineering within the United States Air Force.

Noor Zaman Jhanjhi (N.Z Jhanjhi) is currently working as a Professor in Computer Science (Cybersecurity), Program Director for the Postgraduate Research Degree Programmes in computer science, Acting Program Director for Master of Applied Computing MAC, Director of the Center for Smart Society (CSS5) at the School of Computer Science at Taylor's University, Malaysia. He has been nominated as the world's top 2% research scientist globally, the top researcher in computer science in Malaysia by Scopus in terms of publications and nominated as an Outstanding faculty member by the MDEC Malaysia for the year 2022. He has highly indexed publications in WoS/ISI/SCI/Scopus, and his collective research Impact factor is more than 900 plus points. His Google Scholar H index is 46, and I-10 Index is close to 202, and his Scopus H index is 36, with more than 550 publications on his credit. He has several international Patents on his account, including Australian, German, and Japanese. He edited/authored over 40 research books published by world-class publishers, including Springer, Taylors and Frances, Willeys, Intech Open, IGI Global USA, etc. He has excellent experience supervising and co-supervising postgraduate students, and more than 33 Postgraduate scholars graduated under his supervision. Prof. Jhanjhi serves as Associate Editor and Editorial Assistant Board for several reputable journals, such as PeerJ Computer Science, CMC Computers, Materials & Continua, Computer Systems Science and Engineering CSSE and Frontier in Communication and Networks. He received Outstanding Associate Editor for IEEE ACCESS. Active reviewer for a series of top-tier journals has been awarded globally as a top 1% reviewer by Publons (Web of Science). He is an external Ph.D./Master thesis examiner/evaluator for several universities globally and evaluated 50 plus theses. He has completed more than 40 internationally funded research grants successfully. He has served as a Keynote/Invited speaker for more than 60 international conferences and chaired international conference sessions internationally. He has vast experience in academic qualifications, including ABET, NCAAA, and NCEAC, for 10 years. His research areas include Cybersecurity, IoT security, Wireless security, Data Science, Software Engineering, and UAVs.