IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 7 August 2024, accepted 8 September 2024, date of publication 19 September 2024, date of current version 24 October 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3463712

RESEARCH ARTICLE

The Parallel Fuzzy C-Median Clustering Algorithm Using Spark for the Big Data

MOKSUD ALAM MALLIK[1,2], NURUL FARIZA ZULKURNAIN[1], SUMRANA SIDDIQUI[3], AND RASHEL SARKAR[4]

[1]Department of Electrical and Computer Engineering, International Islamic University Malaysia, Kuala Lumpur 53100, Malaysia
[2]Department of Computer Science and Engineering, Lords Institute of Engineering and Technology, Hyderabad 500091, India
[3]Department of Computer Science and Engineering, Deccan College of Engineering and Technology, Hyderabad 500001, India
[4]Department of Computer Science and Engineering, The Assam Royal Global University, Assam 781035, India

Corresponding author: Moksud Alam Mallik (malammallik@lords.ac.in)

ABSTRACT Big data for sustainable development is a global issue due to the explosive growth of data and according to the forecasting of International Data Corporation(IDC), the amount of data in the world will double every 18 months, and the Global Data-sphere is expected to more than double in size from 2022 to 2026. The analysis, processing, and storing of big data is a challenging research concern due to data imperfection, massive data size, computational difficulty, and lengthy evaluation time. Clustering is a fundamental technique in data analysis and data mining, and it becomes particularly challenging when dealing with big data due to the sheer volume, velocity, and variety of the data. Big Data frameworks like Hadoop MapReduce and Spark are potent tools that provide an effective way to analyze huge datasets that are being processed by the Hadoop cluster. Apache Spark is one of the most widely used large-scale data processing engines due to its speed, low latency in-memory computing, and powerful analytics. Therefore, we develop a Parallel Fuzzy C-Median Clustering Algorithm Using Spark for Big Data that can handle large datasets while maintaining high accuracy and scalability. The algorithm employs a distance-based clustering approach to determine the similarity between data points and group them in combination with sampling and partitioning techniques. In the sampling phase, a representative subset of the dataset is selected. In the partitioning phase, the data is partitioned into smaller subsets that can be clustered in parallel across multiple nodes. The suggested method, implemented in the Databricks cloud platform provides high clustering accuracy, as measured by clustering evaluation metrics such as the silhouette coefficient, cost function, partition index, clustering entropy. The experimental results show that c=5, which is consistent for cost function with the ideal silhouette coefficient of 1, is the optimal number of clusters for this dataset. A comparative study is done to validate the proposed algorithm by implementing the other contemporary algorithms for the same dataset. The comparison analysis exhibits that our suggested approach outperforms the others, especially for computational time. The developed approach is benchmarked with the existing methods such as MiniBatchKmeans, AffinityPropagation, SpectralClustering, Ward, OPTICS, and BRICH in terms of silhouette index and cost function.

INDEX TERMS Data clustering, big data framework, fuzzy C means, fuzzy C median, spark.

I. INTRODUCTION

In today's world, the World Wide Web (WWW) is considered the archive of an enormous amount of data. By 2025, the worldwide datasphere is expected to grow to 175 zettabytes, according to IDC [1]. Social media platforms like Facebook and Twitter have grown in importance as channels for communication and idea sharing. The quick information exchange on social networks is their most significant aspect [106]. The substance and complexities of WWW are increasing day by day. Presently the circumstances are such that we are suffocating in data yet starving for knowledge.

The associate editor coordinating the review of this manuscript and approving it for publication was Yiming Tang.

Because of these circumstances, data clustering is extremely important to get valuable data from WWW.With the vast amount of data generated on the internet, businesses [2] must understand user preferences and provide tailored experiences to enhance engagement and satisfaction. Target marketing, performance prediction, fraud detection, manufacturing, and medical diagnosis are highly demanding. Data clustering [3], [4], [5] is the task of collecting information objects into batches using an unsupervised learning framework, which is utilized to pit the fundamental likeness of information and split the data set in the direction of many subgroups. Each information subset is a cluster; the chosen trials inside the collection are similar to one another, but the trials bridging different clusters are not similar in any way. Generally, the similarity of trials is determined by Euclidean metric, Manhattan metric, Markov metric, Chebyshev metric, cosine affinity, Pearson metric, Jaccard affinity, probability solidity, etc [6], [7]. Clustering approaches have been extensively utilized in real-world scenarios, such as client assembling in business schemes, identifying spam over the Internet, gene sequence classification in bioinformatics, surveying industrial electricity utilization behavior in the electricity market, etc. [8], [9].

In healthcare and other centralized systems and applications, the processing of data in real-time across multiple information systems is a bottleneck [107]. The advancement of imaging technologies has made it possible to obtain more comprehensive medical data. Nevertheless, this advancement has come at the cost of an increase in data volume, making it more challenging to analyze medical picture data by human vision [108]. In today's market, big data is in very high demand due to its extremely large datasets that have been painstakingly combined from several fields and are meant to expand quickly [10], [11], [12]. The size and complexity of this data set are too huge and complicated for typical data management techniques to store or analyze it effectively. Big data is a compilation of information from several sources that is generally described using five criteria: volume, value, variety, velocity, and veracity [13], [14]. A big data framework is a set of tools, technologies, and practices designed to efficiently process, store, and analyze massive volumes of data that exceed the capabilities of traditional data processing systems [15], [16]. These frameworks enable organizations to extract valuable insights from large and complex datasets. Several prominent big data frameworks are Hadoop, Storm, Kafka, Apache Spark, Flink, and Samza. Figure 1 shows the big data framework.

Many researchers have made significant contributions to the field of big data through techniques, specialized tools, technologies, and frameworks such as the Hadoop ecosystem, Apache Spark, NoSQL databases, stream processing platforms, and machine learning algorithms [17]. More advanced and sophisticated techniques need to be developed with a focus on minimal computation time and efficient processing utilizing less storage [18], [19].

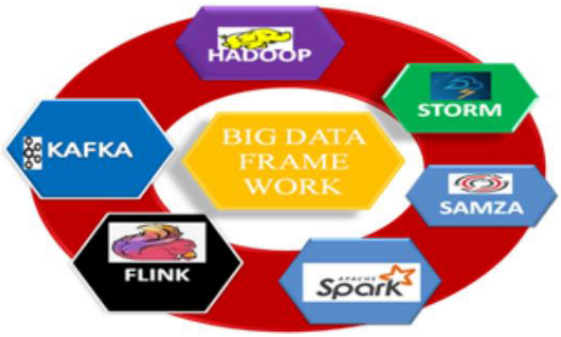

FIGURE 1. Big data framework.

When processing big data, especially with a focus on speed and efficiency, researchers are required to develop clustering algorithms that can handle large datasets while minimizing computation time [20], [21]. There are a few popular clustering algorithms: K-Means Clustering, Fuzzy C-Means, Hierarchical ClusteringDBSCAN (Density-Based Spatial Clustering of Applications with Noise), Mean Shift Clustering, Gaussian Mixture Models (GMM), Agglomerative Clustering, Spectral Clustering, and Affinity Propagation [9], [22]. The fuzzy C-Means (FCM) is one of the most popular clustering techniques [19], [23]. It is a variation of the classic K-Means algorithm that allows data points to belong to multiple clusters to varying degrees of membership [24], [25]. Due to its straightforward computation and high-quality partition, the FCM approach has been extensively employed in the field of log data. However, because of how much processing it requires, the sequential FCM algorithm is too sluggish to complete the partition operation in a reasonable length of time [26]. FCM's computational complexity can make it less suitable for large datasets with many features. In this paper a new algorithm has been developed, the Fuzzy C-least median (FCLM) algorithm which is an improvement to Fuzzy C-means (FCM) algorithm. As it is concerned with the least value among medians, it wipes out means squared error and eliminates the effect of outliers of FCM. A new parallel clustering dubbed parallel FCLM technique has been further developed for weblog big data on the distributed memory computing platform using Apache Spark. The originality of this study is in the creation of a brand-new fuzzy C median algorithm based on Spark that offers a demanding Partition Coefficient and Silhouette Score for improved clustering with recent research work. The proposed research mainly concerns parallel clustering algorithms under multi-machine clustering techniques for big data clustering techniques (See Figure 2).

This paper is organized as: Section I provides a brief introduction about this paper, Section II gives the literature survey of big data clustering, Section III is brief about the methodology, Section IV elaborates on the experimental results and the conclusion and future work of this paper are present in Section V.

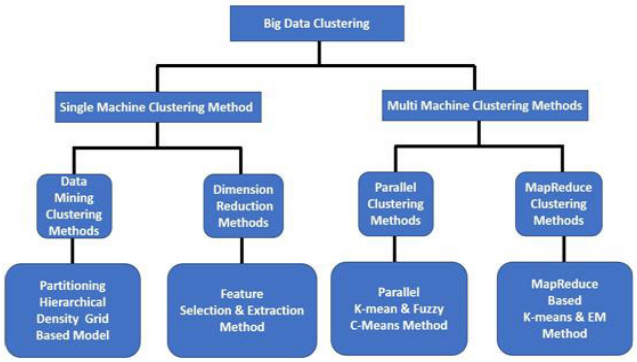

FIGURE 2. Structure of big data clustering techniques.

II. LITERATURE SURVEY

A. BACKGROUND STUDY

In the context of Big Data, it's important to choose clustering algorithms and techniques that are scalable, efficient, and capable of handling high-dimensional data [12]. Distributed computing frameworks like Apache Hadoop and Apache Spark are often used to parallelize and distribute clustering tasks across large clusters of machines, making it feasible to analyze and process massive datasets efficiently [27], [28], [29]. In the recent few years, the research study for parallel clustering algorithms in big data frameworks has been a continuous phenomenon [30], [31]. The clustering algorithms are widely categorized as follows: K-means clustering algorithm, K-means variant, Fuzzy C-means clustering algorithm, Collaborative filtering technique, Possibility C-means clustering algorithm, and Optimization based clustering algorithm (see Figure 3).

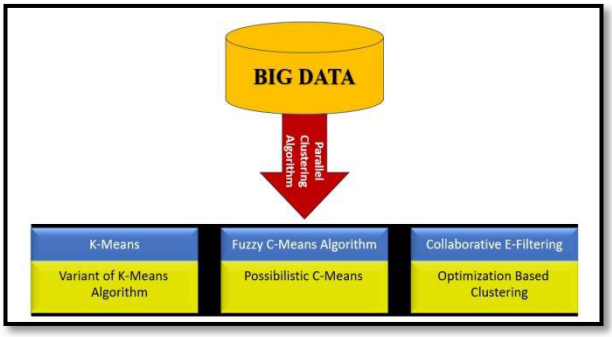

FIGURE 3. Classification of distinct parallel clustering techniques for big data frame.

K-Means: The K-means clustering technique, among many others, is popular due to its straightforward algorithm and quick convergence [32]. The strengths of K-means are simple, easy to understand, computationally efficient, works well for spherical clusters, equally sized, and scalable to large datasets with optimizations (e.g., Mini-Batch K-Means) [33]. The shortcomings are sensitive to the choice of initial centroids which can lead to suboptimal results. It is incapable of handling outliers and non-spherical or irregularly shaped clusters [34], [35].

Fuzzy C-Means (FCM): Another popular clustering method is fuzzy C-means clustering (FCM). An effective approach for mining data and deriving rules from a dataset in which fuzzy features are prevalent is the fuzzy C-means (FCM) algorithm [36]. The FCM allows data points to belong partially to multiple clusters, providing a softer assignment. The FCM is slightly more computationally intensive than K-Means [37].

Possibility C-Means (PCM): In 1996, Krishnapuram and Keller presented the Possibilistic C-Means Clustering (PCM). PCM eliminates the requirement that the sum of memberships equal one and is more resistant to noise [38] is similar to FCM but provides more flexibility in modeling uncertainty and can model conflicting information better than FCM.

Optimization-Based Clustering Algorithms: Optimization-based clustering algorithms are a class of unsupervised machine learning techniques that aim to partition data points into clusters in such a way that an objective function is optimized [39], [67].

DBSCAN (Density-Based Spatial Clustering of Applications with Noise): DBSCAN identifies clusters based on the density of data points [40]. It doesn't require specifying the number of clusters in advance [41].

Hierarchical Clustering: Hierarchical clustering builds a tree-like hierarchy of clusters by iteratively merging or splitting clusters based on certain linkage criteria (e.g., single-linkage, complete-linkage, or average-linkage). The optimal number of clusters can be determined by analyzing the dendrogram [42].

Spectral Clustering: It involves transforming the data into a lower-dimensional space using the eigenvectors of this matrix and then applying K-Means or another clustering algorithm in the reduced space [43].

Collaborative filtering (CF) is the process of selecting or assessing information based on the views of other individuals. With the use of CF technology, significant amounts of data can be filtered by bringing the opinions of numerous, online communities together [44].

The increasing number and variety of data on the internet inspires to study of parallel clustering techniques for big data [45]. Clustering log data in the context of big data is driven by the need to extract meaningful information, detect anomalies and security threats, optimize resource usage, and improve the overall management and analysis of large and complex log datasets [46], [47]. To handle the uncertainty and complexity of user behavior, provide personalized insights, adapt to changing patterns, and support various applications in web analytics, content recommendation, and online marketing optimization, it's essential to preprocess and prepare the web log data properly and consider the scalability of FCM for large-scale weblogs by potentially exploring distributed or parallelized implementations [5].

TABLE 1. Comparative study of parallel clustering algorithm.

Research	Findings	Limitations
A New Criterion for Improving Convergence of Fuzzy C-Means Clustering by Joaquín Pérez-Ortega et al.(2024)[118]	There was a noticeable decrease in the number of iterations without a discernible impact on the quality of the answer. It's important to note that the suggested method typically produces better results as dataset sizes rise.	The proposed algorithm should focus on using variations of the weighted Euclidean distance for various parameter weight values.
Big data fuzzy C means algorithm based on bee colony optimization using an Apache Hbase by Seyyed Mohammad Razavi et al (2021)[119]	The suggested algorithm performs well and is highly precise. This algorithm performs significantly better than previous big data clustering techniques in terms of execution time on large datasets.	The algorithm's I/O time is extremely low since the interim results from each algorithm iteration are saved in the Hbase table.
A Bi-directional Fuzzy C-Means Clustering Ensemble Algorithm Considering Local Information by Chunhua Ren et al (2021)[120]	Local Information Bi-directional FCM (LI_BIFCM)clustering ensemble strategy that improves clustering quality by overcoming border point misclassification. The LI_BIFCM clustering method exhibited superior performance compared to three clustering ensemble strategies and four conventional clustering algorithms.	LI_BIFCM not provide base clustering optimization. Its application to practical situations like consumer analytics can be given more consideration.
FCM-Enhanced Logarithmical particle Swarm Optimization (ELPSO) by Jian Zhang et al. (2020)[121].	FCM-ELPSO can better balance exploration and exploitation because of its remarkable convergence ability and prevention of local optimum. When compared to other clustering techniques, FCM-ELPSO can generate higher quality clusters on the chosen datasets with a lower standard deviation, particularly in the high dimension and huge data instances. The UCI datasets show good performance for FCM-ELPSO.	The suggested techniques has no practical application in a variety of domains, including text mining, picture segmentation, and medical issues. FCM-ELPSO perform not well for complicated datasets.
A fuzzy c-means algorithm based on the relationship among attributes of data and its application in tunnel boring machine(SVR-FCM) by Maolin Shi et al. (2019)[122].	(SVR–FCM) is proposed for the design and analysis of engineering systems. SVR–FCM algorithm shows higher effectiveness and advances the SVR–FCM algorithm compared with other popular clustering algorithms	SVR–FCM) is not applicable for complex engineering systems.
K-Means Hadoop Map Reduce (KM HMR) by Sreedhar C . et al.(2017)[14].	The suggested KM-HMR techniques outcomes have performed better as compared to other clustering techniques concerning accomplishment time.	The difficulties of implementing map and reducing for large datasets, the inability to provide effective job scheduling for vast datasets.
A Spark-Based Parallel Fuzzy c -Means Segmentation Algorithm for Agricultural Image Big Data by L Binet al.(2019)[15].	The agricultural image testing set sees a 128 % performance increase over the Hadoop-based method using the Spark-based parallel FCMs technology.	The results are satisfactory only for 10 computing nodes. This algorithm is suitable only for agricultural image big data.
Arpan Man Sainju, Danial Aghajarian, Zhe Jiang, and Sushil Prasad, "Parallel Grid-based Colocation Mining Algorithms on GPUs for Big Spatial Event Data by Arpan M Sainju et al. (2017)[16].	GPU-grid-join+ outperforms GPU-grid-join on the Nvidia P100 GPU by 4 to 12 times, and OpenMP implementation on an Intel(R) Xeon(R) CPU with 12 cores by 56 to 126 times. The speedup ranges from 3 to 7-fold for synthetic data and 9 to 42-fold for real data.	The proposed approach (integrating GPU implementation with MapReduce frameworks) is not suited for a cluster of nodes.For the scenario of streaming input data, it does not perform well.

TABLE 1. *(Continued.)* **Comparative study of parallel clustering algorithm.**

Clustering on Big Data Using Hadoop Map Reduce by Nadeem Akthar et al. (2015)[17].	The main benefit of selecting the data points from the large-scale datasets is to prevent outlier points from being involved in the final development of the cluster.	The gap calculation function is a very complex and time-consuming process.
A good K-means construct clustering technique for huge scope information by Ankita Sinha, and Prasanta K. Jana (2016)[18].	By automating the input clusters beforehand, the recommended K-Means methodology overcomes the resolution concerns that plague the K-Means clustering method. Even as the amount of data and the number of systems increased, the performance of the Spark framework-based K-means clustering method improved.	It was executed in Apache Spark frame and did not include a few significant factors of big data such as veracity and velocity. The huge streaming real-time data could not be clustered successfully.
AcceleratedMapReduce-found K-Prototypes (AMRKP) clustering technique for handling big data framework by Mohamed Aymen Ben H. K.(2017)[19].	As a result of the drastically reduced number of I/O operations, the reading and writing of the data in this case were done only once. This plan uses a cut-off approach to reduce the unneeded distance between the cluster's center and its data points to speed up the clustering process. The new AMRKP exceeds traditional clustering approaches in terms of effectiveness and scalability.	Iteration count reduction and scalability improvement were not possible with AMRKP.
A Concurrent K-Medoids technique for Clustering built on MapReduce by M. OmairShafiq and Eric Torunski(2016)[20].	This clustering technique proved effective, straightforward, and capable of handling datasets with varied properties, such as volume, velocity, and diversity.	Did not apply to clusters of bigdata-intensive applications.
MapReduce-based K-frameworks (MR-KP) by Mohamed Aymen Ben H.K.(2015)[21].	It has evolved into a popular and effective clustering approach for large, mixed datasets.	This approach was ineffective for real-time applications like fraud detection.
MapReduce fuzzy c-mean (MR-FCM) clustering Algorithm by **Simone** A Ludwig(2015)[22].	The MR framework was used to parallelize this clustering technique by showing the map and reducing the function procedure.	This technique is not capable of handling huge data, which consists of gigabytes of data.
An enhanced FCM clustering algorithm by Minyar Sassi Hidriet al.(2017)[23].	An upgraded FCM clustering technique utilizes sampling varied with a divide and combine scheme for clustering large information. The first step is to separate the data into distinct subsets and use each node individually. Subsets of the subsets were then sampled and then divided at random into subsamples. With the resources available, this algorithm ran efficiently with reduced time and space complexities.	Only works well with the supplied resources and has increased time and space complexity. Performs tedious data division into distinct subsets.
A weighted kernel possibilisticc means(wkPCM) technique built on cloud-based computing for clustering large-scale information by Qingchen Zhang and ZC.(2014)[24].	To reduce the distortion caused by the noisy data, the kernel weights were combined to define the object's relevance in the kernel clustering. The distributed wkPCM clustering method that was suggested was based on the MR frame, which offers real-time data sets impressive computing performance.	In the suggested wkPCM technique for clustering large-scale information, the survey of complex space and great features is lost.
Privacy-Preserving Big-OrderPCM (PPHOPCM) by Qingchen Zhang et al.(2017)[25].	The distributed HOPCM technique is based on the MR framework to deal with big data. The PPHOPCM was used to keep the information on the cloud server by implementing the Brakerski-Gentry Vaikuntanathan (BGV) hide strategy on HOPCM. Polynomial functions were used to modify the cluster centers and membership matrix in the PPHOPCM.The developed PPHOPCM technology successfully clustered the enormous dataset and safeguarded cloud data.	HOPCM strategies' order can be enhanced with the aid of cloud computing servers for peak extensible large-scale information clustering.

TABLE 1. *(Continued.)* **Comparative study of parallel clustering algorithm.**

Clustering-based Collaborative Filtering (ClubCF) by Rong Hu et al.(2014)[26].	The aim is to provide the same facilities for recruitment in the same clusters for the recommendation of services. It consists of two parts. To make the data sets ready for the following processing, they are divided into small clusters in the first section. The determined clusters are subjected to CF in the section that follows. The cluster's administrator count was lower than the number of offered online services. The temporal complexity of CF was less complex in comparison.	This technique could not solve the shortage matter which can be increased by the semantic survey for better inclusion.
Unorganized data survey on big data using MapReduce by Subramani V. et al.(2015)[27].	Suggested the predictive mechanism-based CF approach for the effective processing of large-scale data simultaneously. Effective storage aggregation, filtering, and maintenance are performed using the MR framework. The CF was used to clean the data. The resulting clustering strategy was improved by using sentiment analysis to convert the input data into emoticons and tokens. The complexity analysis performance improved significantly as a result of the simulation.	The old Apache Mahout Cooperative filtering approach for the proposal generation procedure was not systematic enough and can be still improved further for large-scale data clusters.
Social media created large-scale data clustering using genetic algorithm by P. Sachar and V. Khullar(2017)[28].	GA with the help of Hadoop developed finer performance time and space complexities that in turn reduced the user cost for clustering. The effectiveness and repetitious nature of the GA were its key assets, which is why the Java-based GA mentioned above was chosen.	As a result of its flawless nature, the algorithm that utilized a hereditary algorithm for huge-scope information clustering can in any case upgrade the profitability of Big Data Analytics.
Top-quality clustering of big data and settling void grouping issue with a dynamic half-and-half algorithm by J. Karimov and M. Ozbay(2015)[29].	Proposed a cross-browsing Clustering with Empty Clustering Solution by integrating the Fireworks and Cuckoo Search (CS) approach with some interests related to centroid computation. Initially, to eliminate empty clustering issues some indicative points were opted. The hybrid method then used these sites to determine the centroid. The suggested plan's primary benefit is especially true in scenarios where the number, volume, and size of cluster specifications are predicted to increase.	The suggested cross-generative clustering framework has a runtime drawback when contrasted with other algorithms.
Parallel Semi-supervised Multi-Ant Colonies Clustering group Based on MapReduce procedure by Yan Yang et al.(2015)[30].	It was developed to deal with enormous data sets. This method integrates the pair-wise bars in every ant colony clustering task and estimates the new similarity matrix. It enhanced the computational competence of the huge information grouping through the assistance of the MR framework.	The costs of the iteration are very large which is affecting the execution of the algorithm.

B. COMPARATIVE STUDY OF PARALLEL CLUSTERING ALGORITHM

The numerous unanswered queries and issues surrounding various huge data clustering approaches are covered in this section.

Emphasize the shortcomings of the previous research that we have addressed in our proposed work, based on various research findings:

The FCM algorithm [22] is the most used clustering method; it groups comparable aspects of data without

knowing its contents, although it has problems with means squared error and the impact of outliers. The work on K-Means Hadoop Map Reduce (KM HMR) by Kumar and Mohbey faced the difficulties of implementing map and reduce for large datasets [14] The Pietrzykowski implemented, [22] MapReduce fuzzyc-mean (MR-FCM) clustering algorithm is not able to handle massive data. The FCM clustering technique was enhanced by MalikaBendechachea et al. [9], however, it increases the time and spatial complexity. A parallel FCM technique was developed on the Spark platform by Neumann and Kunkel with a performance boost of 128% over the Hadoop-based solution, although it is limited to 10 computing nodes and only agricultural image big data [15]. A new parallel clustering dubbed parallel FCLM technique has been further developed for weblog big data on the distributed memory computing platform Apache Spark. This study is unique in that it develops a novel fuzzy C median method based on Spark, which provides an exacting Partition Coefficient and Silhouette Score for enhanced clustering using the latest research findings.

III. METHODOLOGY

Unsupervised fuzzy clustering techniques with the highest popularity is the fuzzy C Means method [71]. Constrained soft clustering is one of the most used unsupervised fuzzy clustering methods [72], [73], [74]. The fundamental goal of the FCM method is to locate distinct clusters based on the samples' proximity to cluster centers [75]. The FCM approach finds out the ideal cluster centers that yield good partitioning outcomes by minimizing the objective function [76]. The sum of the distances between points in various clusters and their centers is used as a criterion for determining the cluster's centre.

The Parallel Fuzzy C-Median Clustering Algorithm is an efficient method for partitioning a dataset into clusters where each data point belongs to clusters with varying degrees of membership, and the clustering is done in parallel to improve computational efficiency. The steps involved can be categorized into three main phases: sampling, partitioning, and the distance-based clustering approach.

- **Random Sampling:** Randomly select a subset of data points from the weblog dataset. This subset should be large enough to represent the overall distribution and structure of the data.
- **Distributed Data Storage:** Divide the data into smaller, manageable segments to facilitate parallel processing. Store each partition on a different processing unit or node in a distributed computing environment.
- **Distance-Based Clustering Approach**:Perform clustering using a distance metric and the Fuzzy C-Median criterion.
    - **Membership Initialization:** Initialize the membership matrix where each element $\boldsymbol{u_{ij}}$ represents the degree of membership of data point x_i to cluster j. Ensure that the sum of memberships for each data point across all clusters equals 1.
    - **Compute Distance:** Calculate the distance between each data point and each cluster centroid using an appropriate distance metric (e.g., Euclidean distance).
    - **Update Membership Values:** Update the membership matrix based on the computed distances. Use the formula:

$$U_{ij} = \frac{1}{\sum_{K=1}^{C} \{\frac{d_{ij}}{d_{ik}}\}^{\frac{2}{m-1}}} \tag{1}$$

where $\boldsymbol{d_{ij}}$ is the distance between the data point $\boldsymbol{x_i}$ and centroid $\boldsymbol{c_j}$, and $\mathbf{m}$ is the fuzziness parameter.

    - **Compute New Centroids:** Update the centroids using the fuzzy membership values:

$$C_j = \frac{\sum_{i=1}^{N} U_{ijX_i}^{m}}{\sum_{i=1}^{N} U_{ij}^{m}} \tag{2}$$

    - **Cluster Assignment:** Assign data points to clusters based on the highest membership value.

The Parallel Fuzzy C-Median Clustering Algorithm leverages parallel processing to improve efficiency and scalability while effectively partitioning and clustering big datasets through the use of structured phases.

Using Apache Spark to implement the Fuzzy C-Median clustering algorithm requires several crucial technical stages, such as parallelization, partitioning, and sampling techniques. The specific methods and techniques to accomplish this are listed below.

1. Setup and Initialization

Before implementation, ensure that you have a Spark cluster set up. The implementation will use PySpark for simplicity.

a. Import Libraries

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col
import numpy as np
import pandas as pd
```

b. **Initialize Spark Session**

spark = SparkSession.builder.appName(''FuzzyCMedian Clustering'').getOrCreate()

2. Random Sampling

The Random Sampling approach is used to minimize the data size for faster initial grouping.

3. Partitioning

Partitioning helps distribute the workload across the cluster. Spark's partitioning strategy has been applied.

4. Parallelization Strategies

Spark inherently parallelizes operations using its RDDs (Resilient Distributed Datasets) and DataFrame APIs. Below is a step-by-step implementation of the Fuzzy C-Median algorithm, incorporating parallelization:

a. Distance Function

Define the distance function for the median-based clustering.

defmanhattan_distance(x, y):

returnnp.sum(np.abs(x - y))

b. Update Membership Function

Update the membership values for each data point to all cluster centers.

c. Update Cluster Centers

Update the cluster centers based on the current membership values.

d. Fuzzy C-Median Implementation

Implement the main loop of the Fuzzy C-Median algorithm.

5. Parallelize with Spark

Distribute the computation across Spark workers.

a. Convert Data to RDD

b. Distribute Centers and Membership Update

Distribute the membership update and center update steps.

c. Main Fuzzy C-Median with Spark

Integrate the distributed functions into the main algorithm.

6. Execution

Execute the algorithm on weblog data.

A. APACHE SPARK'S RESILIENT DISTRIBUTED DATASET(RDD) WITH DATABRICKS

MapReduce was inefficient (or intractable) for interactive or iterative computing jobs and a complex framework to learn, so from the onset, the idea of Spark evolved for a simpler, faster, and easier big data computing framework. To address the difficulty of large data processing and analysis for web log data, we provide a parallel Fuzzy C Median technique built on the Spark platform for distributed computing [77], [78]. Utilizing cloud data stored in a distributed computing platform (Databricks), iterative computing is performed by simultaneously calculating and updating the membership degrees of data points to various cluster centers. The information from the clustered data point cloud is then used to recreate the segmented data [79].

Spark is significantly faster than Hadoop MapReduce because it offers in-memory storage for intermediate calculations [80], [81]. It includes machine learning libraries (MLlib), SQL for interactive queries (Spark SQL), stream processing libraries (Structured Streaming) for dealing with real-time data, and graph processing libraries (GraphX). Apache Spark is a multi-language engine for doing big data processing, data engineering, data science, and machine learning on single-node workstations to make interactive big data applications fast and easy [12], [82]. The most basic abstraction in Spark is RDD which is a distributed query processing engine, designed for in-memory processing and a pointer to a distributed dataset for storing and computing information irrespective of data memory [83], [84]. Clustering using Apache Spark is straightforward with Databricks which is a single analytics platform for parallel processing and cluster computing [85].

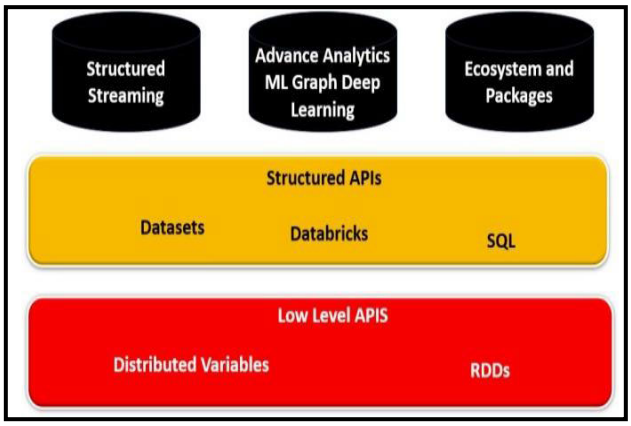

FIGURE 4. Databricks architecture.

B. DATABRICKS

Databricks is a big data engineering tool hosted in the cloud. The people behind Apache Spark developed this unified analytics platform [86], [87]. Additionally, it offers links to a range of frequently used tools such as shared Jupyter notebooks, GitHub integration, automation monitoring, scheduling, and debugging, as well as other features. Databricks provides quick installation of CPU and GPU clusters on instances on Amazon Web Services (AWS) and Azure for the greatest degree of flexibility. The community edition of the Databricks cloud environment (see Figure 5) is used for my work.

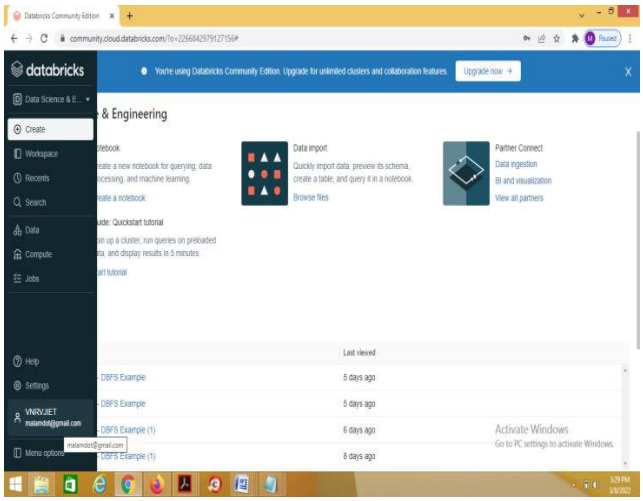

FIGURE 5. Databricks community cloud screen.

C. DATABRICKS FILE SYSTEM(DBFS)

DBFS is a Databricks File System that allows storing data for querying inside of Databricks. DBFS is an interface on top of scalable object storage that transforms native cloud storage API calls into Unix-like filesystem calls. After a cluster is shut down, files are saved to object storage, ensuring that no data is lost.

Data Flow:

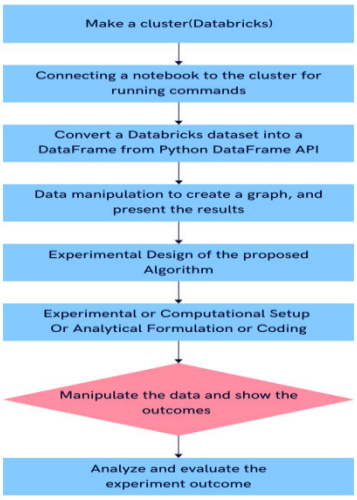

FIGURE 6. The flow of the parallel fuzzy C median algorithm.

1) CREATE CLUSTER

The simplest method to begin a new cluster is to click the New button: Click the Create Icon Create button in the sidebar and select Cluster from the options. The New Cluster page appears. The cluster needs to be named and set up.

a: CONNECT A NOTEBOOK TO THE CLUSTER AND USE IT TO RUN COMMANDS

A notebook is an online web-based user interface for a document containing narrative text, pictures, and executable code. Additionally, it discusses data visualizations, sharing visuals as dashboards, parameterizing notebooks and dashboards with widgets, utilizing notebook processes to construct complex pipelines.

b: USING THE PYTHON DATAFRAME API, CREATE A DataFrame FROM A DATABRICKS DATASET

I'm using PySpark to construct a DataFrame from Databricks.

PySpark

PySpark is Apache Spark's Python Application Programming Interface(API) [88], [89]. It is compatible with the vast majority of Spark technologies, including Spark Core, DataFrame, MLlib (Machine Learning), and Spark SQL. The Spark context is created by the PySpark Shell, which also connects the Python API to the Spark core.

PySpark's ENVIRONMENT SETUP

Download and install PySpark using the procedures below.

Step 1: Get the most recent version of Apache Spark from the official Apache Spark download page.

Step 2:Spark tar file needs to be extracted from the downloaded PySpark – SparkContext.

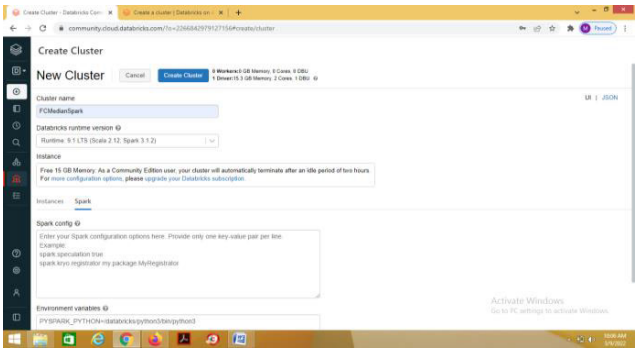

FIGURE 7. Databricks create cluster screen.

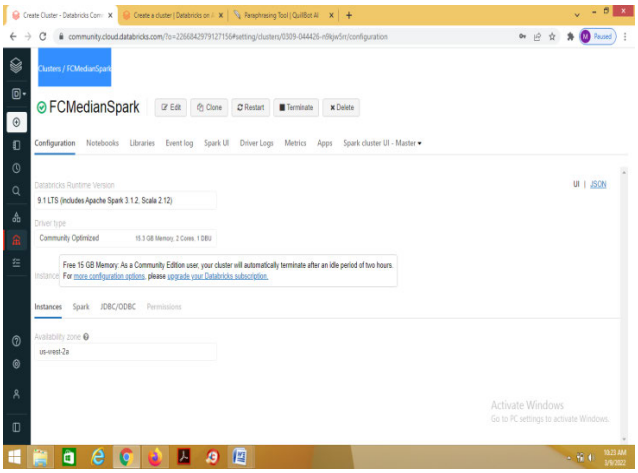

FIGURE 8. Databricks FCMedianSpark cluster screen.

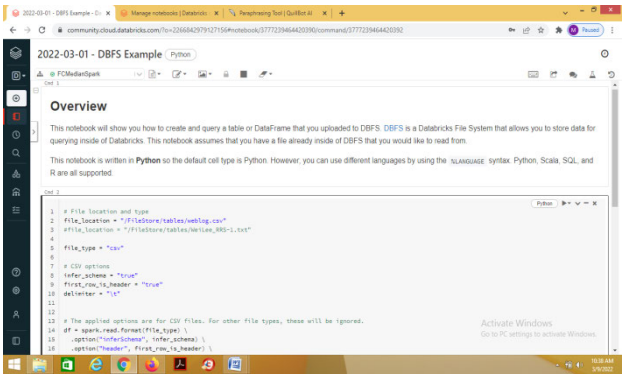

FIGURE 9. Databricks notebook screen.

The pictorial representation of data flow in PySpark is shown in Figure 11.

c: MANIPULATE THE DATA AND SHOW THE OUTCOMES

We use the weblog dataset to perform further Data Analysis.

Its analysis and utilization are essential for organizations looking to make data-informed decisions and stay competitive in the digital landscape [86]. Additionally use Databricks Notebooks to visualize data in a variety of charts, including pie charts, bar charts, scatter plots, and more.

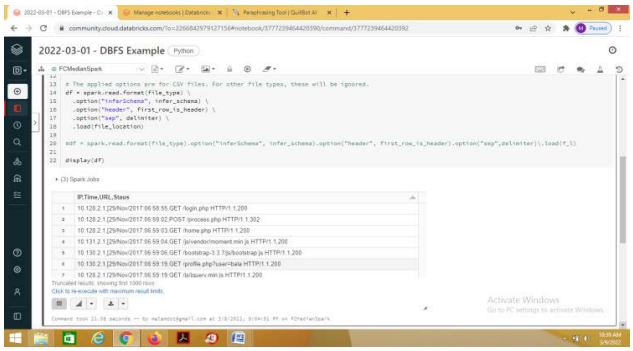

FIGURE 10. Databricks DBFS screen.

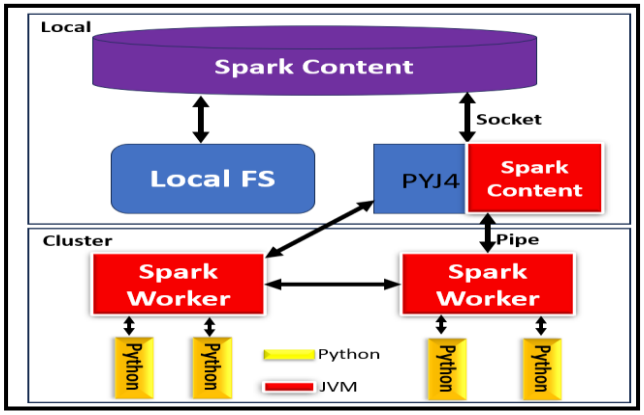

FIGURE 11. Data flow in PySpark.

Parallel Fuzzy Clustering algorithm using Spark:

We proposed an efficient parallel clustering algorithm using spark on big data. The suggested method eliminates batch effects while offering speedy and iterative data processing.

Algorithm: Fuzzy C Median (FCMedian) using Spark on big data:

Input: Data from the cloud (Databricks)

Output: Cluster membership

Step 1: Build a cloud cluster

Step 2: A notebook was connected to the cluster;

Step 3: RDDs were created using data read from the cloud.

Step 4: Distribute "V" to different computer nodes after randomly initialising or updating the cluster's "V" centre.

Step 5: Calculate the fuzzy membership 'U_{ij}' using:

$$\boldsymbol{U_{ij}} = 1 \Big/ \sum_{\boldsymbol{k=1}}^{\boldsymbol{c}} (\boldsymbol{d_{ij}}/\boldsymbol{d_{ik}})^{(\boldsymbol{2/m-1})} \tag{3}$$

Dij Value for each new cluster centers

$$\boldsymbol{D_i} = \text{Median}\{(\boldsymbol{D_{ij}}(\boldsymbol{S_k} - \boldsymbol{S_i}) * \boldsymbol{U_{ij}})\} \forall \text{i} \neq \text{k; k} = 1 \ldots \text{n} \tag{4}$$

Step 6: Compute the fuzzy centers 'v_j' using:

$$V_j = \left(\sum_{i=1}^{n} (U_{ij})^m x_i\right) / \left(\sum_{i=1}^{n} (U_{ij})^m\right), \forall_j = 1, 2, \ldots c \tag{5}$$

Step 7: New cluster centres are determined in each iteration phase using the formula below:

$$\text{p} = \text{Argmin}\{(\text{Di} : \text{n}); \forall \text{i} = 1 \ldots \text{n}\} \tag{6}$$

Step 8: Steps 5 and 6 should be repeated until the minimal 'J' value is reached or

$$||\text{U(k} + 1) - \text{U(k)}|| < \beta. \tag{7}$$

where 'k' is the iteration step. 'β' is the termination criterion between [0, 1].

$$\text{'U} = (U_{ij})\text{n} * \text{V}' \tag{8}$$

is the fuzzy membership matrix. 'J' is the objective function.

Step 9: If limits are defined, then build column charts for cluster centres with n = 2 to limits, analyze the validity indices, and pick the cluster centres with the lowest limits.

Step 10: Apply the Map operation to the distributed cluster to compute and update the membership degree U simultaneously;

Step 11: Use the mapPartitions method to compute and update the cluster centres Vi that is a part of V from the ith computing node.

Step 12: Gather and combine the cluster centres Vi from various nodes to form a new cluster centre V using the reducing procedure;

Step 13: Save the cluster data and export the data from the clustered points.

IV. EXPERIMENTAL RESULTS

A. DATASET

The input server log data is downloaded from the site https://filewatch.net. Filewatcher is a File Transfer Protocol (FTP) search engine that monitors more than two billion files over 5,000 FTP servers. The downloaded file name is "pa.sanitized-access.20070109.gz." A sample server log file entry is given below in Table 2

TABLE 2. An example of a weblog entry.

Data Attribute	Description
1168300919.015	The time of the request
1781	The elapsed time for an HTTP request
17.219.121.198	The client's IP address
TCP_MISS/200	Status code for HTTP responses
1333	In response to the request, bytes sent to the server
GET	the action that was requested
http://www.quiethits.com/hitsurfer.php-DIRECT/204.92.87.134	The URI of the item being discussed the name of the customer's client, and the hostname of the machine from which we received the solicitation.
text/HTML	The object's content type.

TABLE 3. Algorithm for data cleaning.

No. of Steps	Description of the Operation
Step 1	Scan each line of the input file one by one.
Step 2	Remove all URLs with suffixes recorded in the suffix list.
Step 3	Remove all URLs produced by web robots.
Step 4	Remove URLs with query strings.
Step 5	Take out the IP address and store it on a map.
Step 6	Code URL with URL number and store it on a map.
Step 7	Sort each line based on the IP Address encryption code.
Step 8	Print in the required fields to a yield file.

B. CLEANING OF DATA

When a person requests a web page and inputs or clicks on a URL, many URLs are typically created, such as figures, scripts, and so on. As a result, any URLs that end with a graphic should be erased. The robot's request is unwanted because it is not the generated by user, it's self-generated. Hence, we should remove them to increase the accuracy. We employed two methods for extracting robot requests. The first one is checking for an entry in "robots.txt" in web log data and the second one is removing HEAD requests [30], [36], [38]. Next is the removal of URLs with query strings. Normally URL with query strings is used for requesting extra details from the web browser. As it is unnecessary, we remove them as well [30], [36], [38]. IP Address is encrypted to hide the user's identity and to ease future processing. Furthermore, each URL will be appointed a unique number and it will be put away in a URL map along with its number [30], [36], [38]. It's critical to any organization as it enables them to make more informed judgments and comprehend the preferences of their clientele [109]. The data-cleaning algorithm is demonstrated in Table 3.

The output file after applying the data-cleaning algorithm is shown in Table 4. The output file is sorted in ascending order based on the encoded value of the IP Address.

C. USER IDENTIFICATION

After cleaning input web log data, users can be distinguished through its IP since the log file doesn't contain user login information. Next, all requests relating to the individual user are separated accordingly. The algorithm for user identification is shown in Table 5 [30], [36], [38].

TABLE 4. Output file after data cleaning.

IP	Time	Elapsed Time	Bytes	URL
IP1	1168300931.828	142	1599	1
IP1	1168300935.244	501	1617	2
IP1	1168300936.604	1	1617	3
IP1	1168300941.345	2	1593	4
IP1	1168300957.585	186	1585	6
IP1	1168300985.665	145	1563	10

TABLE 5. Algorithm for user identification.

No. of Steps	Description of the Operation
Step 1	Split every line in the input file into obliged fields.
Step 2	Store it in a Map M1 with IP Address as the key and another Map M2 as the value (i.e. required fields). The time is the key to Map M2, and the value is whatever is left of the fields.
Step 3	Sort the internal map m2 taking into account the time key.
Step 4	Print the content of the map M1 to the yield record.

TABLE 6. Output file format after user identification.

User	Time	Elapsed Time	Bytes	URL
IP1	1168300931.828	142	1599	1
	1168300935.244	501	1617	2
	1168300936.604	1	1617	3
	...	..	..	..
IP2	1168300953.645	648	260	5
	1168300990.665	143	260	14

The organization of the yield document produced after user identification is shown in Table 6.

D. SESSION IDENTIFICATION

Keeping in mind the end goal for recognizing client sessions we can approach Time Oriented Heuristics (TOH) as portrayed below [30], [36], [38]:TOH1: The time term of a session should not surpass a time limit α. Let the timestamp of the first URL request in a session is, T_1. If another URL asks for a session with timestamp T_i it is allotted to the same session if and only if T_i-$T_{1\leq}\alpha$. The principal URL asking for timestamp bigger than $T_1 + \alpha$ is taken as the first request of the following session [30], [36], [38].

TOH2: The time spent on a page visit should not surpass a time limit α. Let a URL, the most recently given to a session have a timestamp T_i. The next URL's request fitswith the same session if and only if T_{i+1}-$T_{i\leq}$ α where T_{i+1} is the timestamp of the new URL's request. This URL is now the first of the following session [30], [36], [38].

In our implementation for the interim, we are utilizing TOH1. We have chosen 30 minutes as the estimation of the limit time. The algorithm for user session identification is shown in Table 7 and the output file of session identification is shown in Table 8.

TABLE 7. Algorithm to create user sessions taking into account TOH1.

No. of Steps	Description of the process
Step 1	This step ought to be finished for every line in the information input file.
Step 2	If the Line contains a User ID, then UserId =User ID of the line.
Step 3	Print Line to output file under this User Id and the first session of same User ID.
Step 4	In case that L is the first accessed log of the user then T1 = Line.time else T2 = Line.time.
Step 5	If T2-T1$\leq$ α at that point print Line under the same session to the file.
Step 6	If it is not as in the previous step i.e. Step 5 then output User ID and corresponding line under a new session, T_1 = Line.time.

E. REDUCTION OF DIMENSIONALITY

A robust dimensionality reduction technique can be created while also enhancing clustering by removing log references to low bolster URLs (those are not boosted by a preset number of user sessions). To implement this, we are removing URLs that occur only once [30], [36], [38].

F. ASSIGNING SESSION WEIGHT

For the clustering task, the session files can be separated to remove little sessions keeping in mind the end goal to take out the variation from the data. In any case, directly removing these little measured sessions may bring about loss of a critical measure of information particularly when the quantity of these little sessions is large. Here, we assign weights to each of these sessions based on how many URLs they visited throughout their time. Session weight assignment is done based on the following equation [30], [36], [38].

$$W_{s_i} = 0, \text{if } |s_i| \leq 1$$
$$W_{s_i} = 1, \text{if } |s_i| \geq 1$$

where W is the weight and $|s_i|$ is the number of URLs accessed in a particular session.

TABLE 8. Output file of session identification.

User Session	Time	Elapsed Time	Bytes	URL
IP1S1	1168300931.828	142	1599	1
	1168300935.244	501	1617	2
	1168300936.604	1	1617	3
IP1S2	1168302738.407	81	1623	482
	1168302745.477	138	1559	483
	...	..	..	..
IP2S49	1168300953.645	648	260	5
	...	..	..	..

G. DEVELOPMENT OF USER SESSION MATRIX

We represent sessions using a matrix. Every row denotes a session and the column denotes a URL. If a URL arrives in a session, then the entry for that URL in the specific session will be more prominent than zero. There will be several events of that URL in that session. If the URL is not present then that entry will be zero. Sessions are referred to by utilizing a sparse matrix in row-major form. It reduces processing time to a great extent. After all, we are dividing to standardize the session matrix for every column by its greatest value [30], [36], [38].

H. FUZZY C MEANS USING SPARK & FUZZY C MEDIAN USING SPARK

1) BASE RAW DATA

The raw data are shown in Figure 12.

To analyze the real-time streaming of big data, Apache Spark is ideal [90]. Diverse technologies and methodologies, including distributed processing frameworks like Apache

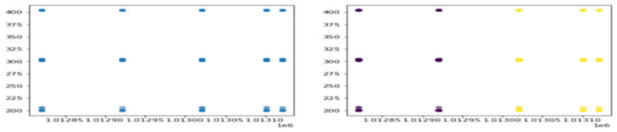

FIGURE 12. Scatter plots of raw data.

Hadoop or Apache Spark, data ingestion and storage systems like Apache Kafka or Apache HBase, and data querying languages like Apache Hive or Apache Pig, can be used to process and analyze base raw weblog big data [96]. The data processing and analysis in this case is done by Apache Spark technology [97]. By building and running both algorithms using Spark's MLlib(pyspark), experiments were conducted on a weblog dataset to compare the performance of Fuzzy C Means and Fuzzy C Median [98].

I. PARALLEL FUZZY C MEANS CLUSTERING USING SPARK

Partition Coefficient and Partition Entropy Coefficient for Fuzzy C Means Clustering using Spark shown in Figure 13 to 15.

The implemented FCM with Spark (as shown in figures 13 to 15) performed significantly better than without Spark, demonstrating the most accurate clustering in terms of computing complexity. The outcome of the FCLM algorithm running using spark is remarkably improved cluster quality.

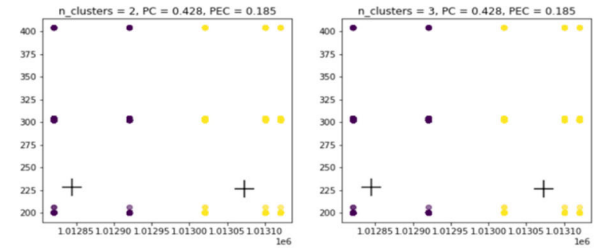

FIGURE 13. Scatter plots of PE and PEC for 2 & 3 clusters in PFCM.

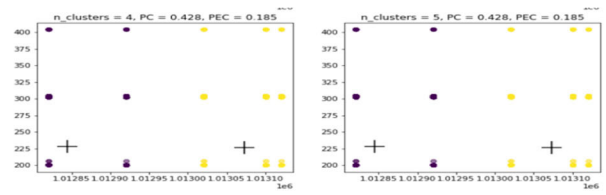

FIGURE 14. Scatter plots of PE and PEC for 4 & 5 clusters in PFCM.

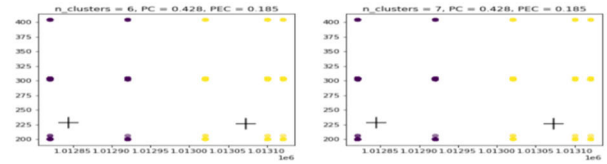

FIGURE 15. Scatter plots of PE and PEC for 6 & 7 clusters in PFCM.

J. PARALLEL FUZZY C MEDIAN CLUSTERING USING SPARK

Partition Coefficient and Partition Entropy Coefficient for Fuzzy C Median Clustering using Spark are shown in Figure 16 to 18.

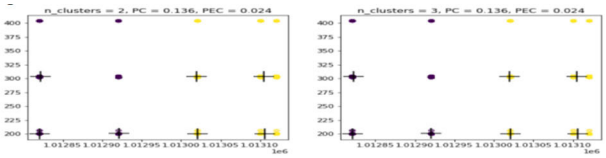

FIGURE 16. Scatter plots of PE and PEC for 2 & 3 clusters inPFCMedian.

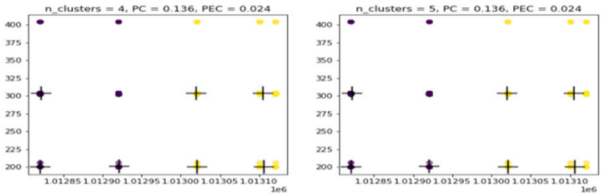

FIGURE 17. Scatter plots of PE and PEC for 4 & 5 clusters inPFCMedian.

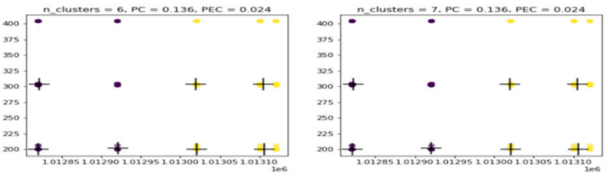

FIGURE 18. Scatter plots of PE and PEC for 6 & 7 clusters inPFCMedian.

K. PARTITION COEFFICIENT

The partition coefficient (PC), which measures the fuzzy degree of the final separated clusters, is calculated using the fuzzy partition matrix; the higher the value, the better the partition result. Fuzzy C means it gives a better partition coefficient.

$$\mathrm{PC} = \frac{1}{n}\sum_{j=1}^{k}\sum_{i=1}^{n} U_{j,i}^{m}$$

L. PARTITION ENTROPY COEFFICIENT

The fuzzy degree of the final partitioned clusters is measured using the fuzzy partition matrix by the partition entropy (PE), and the lower the number, the better the partition outcome. A very high degree of final partition is indicated by the significantly lower partition entropy coefficient produced by the implemented Fuzzy C median utilizing the Spark method.

$$\mathrm{PEC} = -\frac{1}{n}\sum_{j-1}^{k}\sum_{i-1}^{n} U_{j,i}\log_a\left(U_{j,i}\right)$$

A higher PC value suggests that there is less degree of overlap between the clusters and that they are more effectively isolated. Conversely, a higher PE value denotes a higher degree of fuzziness and greater cluster overlapping.

M. SILHOUETTE COEFFICIENT

For a clustering technique, the Silhouette Coefficient is a statistic that is used to assess the quality of the clusters. It gauges an object's cohesiveness, or similarity to its cluster, in contrast to separation, or distance from other clusters. In other words, it measures how distinct and well-defined the clusters are [99].

To determine a final score for the clustering, the Silhouette Coefficient is computed for each data point in the dataset and then averaged. A single data point's "i" silhouette coefficient has the following formula:

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

where:

- S(i) is the Silhouette Coefficient for data point 'i.'
- a(i) is the average distance from data point 'i' to other data points within the same cluster (intra-cluster distance).
- b(i) is the smallest average distance from data point 'i' to data points in a different cluster, minimized over clusters (inter-cluster distance).

The Silhouette Coefficient ranges from -1 to 1:

- A high value (close to 1) indicates that the data point is well-matched to its cluster and poorly matched to neighboring clusters. This suggests a good clustering solution.
- A value near 0 indicates that the data point is on or very close to the decision boundary between two neighboring clusters.
- A low value (close to -1) indicates that the data point is closer to a neighboring cluster than to its cluster, indicating that it may be in the wrong cluster.

We calculate the Silhouette Coefficient for each data point in a clustering solution and then average the results to get the overall Silhouette Score. The grouping is better the higher the average Silhouette Score.

1: Indicates that clusters are well separated and distinct from one another [99].

Fuzzy C Median using Spark:
C=2, Silhouette Score: 0.9052975163457041
C=3, Silhouette Score: 0.8547429823506406
C=4, Silhouette Score: 0.9923897493936394
C=5, Silhouette Score: 1.0000008513967786
C=6, Silhouette Score: 1.0000008513967786
C=7, Silhouette Score: 1.0000008513967786
C=8, Silhouette Score: 1.0000008513967786
C=9, Silhouette Score: 1.0000008513967786
Fuzzy C Means using Spark
C=2, Silhouette Score: 0.9053125353757929
C=3, Silhouette Score: 0.8547676552736689
C=4, Silhouette Score: 0.9923907732363044
C=5, Silhouette Score: 0.9998910822740511
C=6, Silhouette Score: 0.9998910822740511
C=7, Silhouette Score: 0.9998910822740511
C=8, Silhouette Score: 0.9998910822740511
C=9, Silhouette Score: 0.9998910822740511

By receiving both the fuzzy C median using spark and fuzzy c means using spark silhouette scores, it is possible to conclude that the fuzzy c median algorithm provides effective clustering as the vast majority of its values are 1.

1) COST FUNCTION

The cost function in the context of the Fuzzy C-Median Clustering Algorithm is typically the sum of squared errors (SSE) or the sum of distances between each data point and its assigned centroid. For fuzzy clustering, the cost function is modified to account for the membership values:

$$\mathbf{J} = \sum_{i=1}^{N} \sum_{j=1}^{C} U_{ij}^{m} \mathbf{d}(x_i, c_j)$$

where U_{ij} is the membership value of the data point x_i in cluster j, m is the fuzziness parameter, and d(x_i, c_j) is the distance between xixi and centroid cj.

To determine the best cluster centers and fuzzy membership degrees, the cost function is utilized to assess the quality of the clustering solution. As it indicates the degree of compactness and separation of the clusters, the cost function is a helpful statistic for assessing the quality of the clustering solution [103], [100]. This cost has exponential time complexity and is NP-hard. Cost Function is shown in Figure 19.

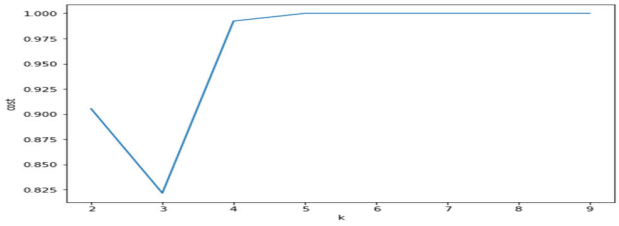

FIGURE 19. Cost function.

The results show that c=5 is the ideal number of clusters for this dataset, which is consistent with our analysis's findings.

The fuzzy c-median using the spark clustering algorithm outperforms the fuzzy c-Means using the spark technique. In this work, the very recent clustering algorithms including MiniBatchKmeans, AffinityPropagation, MeanShift, SpectralClustering, Ward, FCMD Clustering, DBSCAN, OPTICS, BRICH, and Gaussian Mixture are implemented using the PySparkdatabrick cloud environment and the obtained results are shown in Figure 20.

2) COMPUTATIONAL TIME

Shorter computational time indicates a more efficient algorithm, especially important for large datasets and real-time applications. Parallel algorithms are expected to show significant improvements in computational time over their sequential counterparts. The comparative analysis manifests that the FCMD clustering executes in less time than

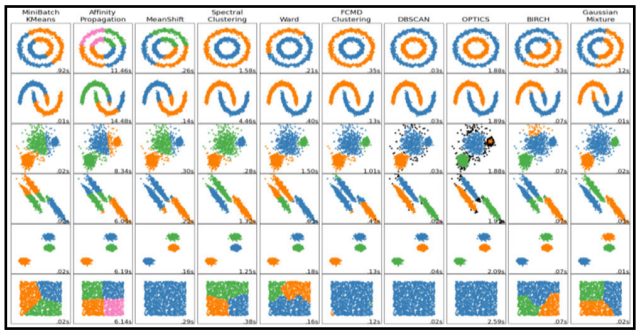

FIGURE 20. Comparison result of web log data for different clustering algorithms.

MiniBatchKmeans,AffinityPropagation,SpectralClustering, Ward, OPTICS, and BRICH.

The speedup is used to compare the parallel FCMedian method implemented in Spark to the parallel FCM algorithm's performance. A larger speedup indicates that the parallel algorithm takes up less time. The trial results is shown in Table 9

TABLE 9. Comparision result of web log data for different clustering algorithms based on SS,DBI,TT.

ALGORITHM	SILHOUTTE SCORE	DAVIES BOULDIN INDEX	TIME TAKEN
FCM	0.6109562467	0.5394433881	0.0041172504
PCM	0.6109562467	0.5394433881	0.0076501369
FPCM	0.4380097141	0.9217368601	0.0036008358
PFCM	0.4380097141	0.9217368601	0.0042347908
T2FCM	0.0526788140	0.8844041451	0.0013318062
IFCM	0.4376675407	0.9224469201	0.0012590885
NCFCM	0.0526788140	0.8844041451	0.0018930435
CFCM	0.4380097141	0.9217368601	0.0029630661
DOFCM	0.4380097141	0.9217368601	0.0065279007
KMeans	0.5089769460	0.8164371385	0.1466021538
Agglomerative	0.5124184506	0.7834062337	0.0276429653
DBSCAN	0.4864975965	0.7851603342	0.0065457821
GMM	0.5084057167	0.8165915999	0.0117471218

GMM-Gaussian Mixture Model

N. FCLMEDIAN USING SPARK PERFORMANCE

The memory consumption and processing speed concerns that come with massive data are specifically addressed by the Parallel Fuzzy C-Median Clustering Algorithm. Here's how the algorithm addresses these challenges:

1) DISTRIBUTED COMPUTING FRAMEWORKS

By leveraging frameworks like Apache Spark, the algorithm can scale horizontally. Each node handles a subset of the data, allowing the system to process large datasets efficiently.

2) MEMORY UTILIZATION

Optimize memory usage to handle large datasets without exhausting system resources.

- **Data Partitioning:** By dividing the dataset into smaller chunks, the algorithm ensures that each partition can fit into the memory of individual nodes. This approach prevents memory overflow issues that are common with large datasets.
- **In-Memory Processing:** For frameworks like Apache Spark, the algorithm takes advantage of in-memory processing capabilities, which reduce the need for repeated disk I/O operations and speed up computations.

By implementing the algorithm on a high-performance computing cluster at Databricks environment, the processing time for clustering was reduced from several hours to minutes.

3) RAND INDEX

The Rand Index assesses how comparable two clusters are to one another by comparing all sample pairs and tallying pairs assigned to the same or different groups in the actual and anticipated grouping [35]. The Rand index has a scale of 0 to 1. The Rand index is one when the two divisions line up exactly. Results from clustering are more reliable when the rand index value is close to 1 than when it is not1. The results of trials on the accuracy of various clustering approaches are shown in Figure 21. The proposed PFCMS method outperforms the FCM and FCLM approaches in terms of rand index rate, it has been shown in Table 10.

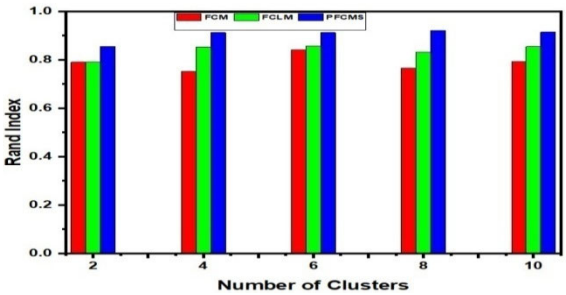

FIGURE 21. Rand index of clustering algorithms (FCM, FCLM, PFCMS).

4) F-MEASURE

A performance indicator called F-Measure is utilized to evaluate the accuracy of the clustering technique used in this case as well as the quality of the clusters. When the results for cluster formation are accurate, the F-Measure values are high.

TABLE 10. Rand index of clustering algorithms(FCM,FCLM,PFCMS).

Number of Clusters	Rand index (RI) (%)		
	FCM	FCLM	P FCMS
2	0.79	0.79	0.8554
4	0.7512	0.852	0.9123
6	0.8412	0.8562	0.9125
8	0.7652	0.8315	0.921
10	0.7923	0.8545	0.915

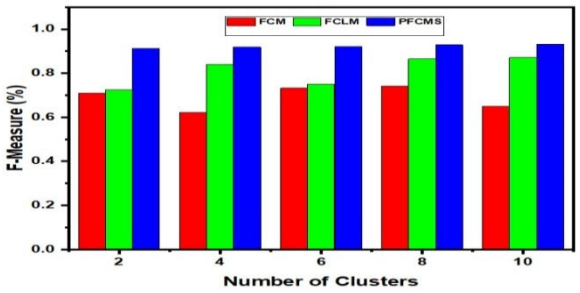

FIGURE 22. F-measure of clustering algorithms(FCM,FCLM,PFCMS).

When it comes to the accuracy of clustering, the suggested PFCMS clustering algorithm performs better than already in-use methods like FCM and FCLM. Table 11 contains the results, and Figure 22 depicts the performance comparison.

5) SUM OF SQUARED ERROR (SSE)

The most crucial and well-liked clustering criterion is SSE. The density of the clusters is evaluated. Figure 23 displays the SSE outcomes for a user session matrix's clustering technique. The suggested technique compared to the FCM and FCLM techniques, has a lower error value, as displayed in Table 12. Finding the respondents or clients who "fit best" for a specific market group (cluster) can be done statistically using cluster analysis. The more similar the consumers in a market sector are, the lower the SSE.

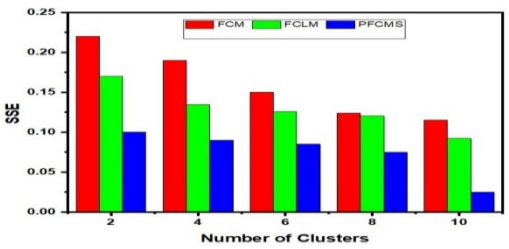

FIGURE 23. SSE of clustering algorithms (FCM, FCLM, PFCMS).

TABLE 11. F-measure of clustering algorithms(FCM,FCLM,PFCMS).

Number of Clusters	F-measure(%)		
	FCM	FCLM	P FCMS
2	0.710	0.725	0.912
4	0.623	0.841	0.918
6	0.732	0.751	0.921
8	0.741	0.864	0.928
10	0.649	0.871	0.931

TABLE 12. SSE of clustering algorithms (FCM,FCLM,PFCMS).

Number of Clusters	SSE (%)		
	FCM	FCLM	P FCMS
2	0.22	0.17	0.1
4	0.19	0.1348	0.09
6	0.15	0.1259	0.085
8	0.1238	0.1205	0.075
10	0.1151	0.0922	0.024 58

6) ACCURACY

The clustering accuracy of the proposed system Parallel FCMedian employing Spark is compared to existing methods FCM (Fuzzy C means) clustering and Fuzzy C Least Median clustering using the parameters Rand Index, F-Measure, and Sum of Squared Error (SSE). For both FCM and the proposed algorithm, give similar data information and ascertain the Rand Index, Sum of Squared error, and F-measure. Then, examine the file evaluations produced by the suggested method PFCMS (Parallel Fuzzy C Median using Spark), as well as FCM (Fuzzy C Means), FCLM (Fuzzy C Least Median), and FCM (Fuzzy C Means).

V. CONCLUSION

The most important issues with big data clustering are examined, and the Parallel Fuzzy C-Median Clustering Algorithm Using the Spark for the Big Datais proposed, developed, and tested to improve clustering quality by eradicating mean squared error and the impact of outliers, as well as to reduce lengthy evaluation times and excessive computational complexity.

For the sake of better clustering to reduce computational complexity and evaluation time the fuzzy C median technique has been implemented using Spark and analyzed its performance using a variety of performance parameters, including silhouette score, execution time, rand index, partition coefficient, partition entropy coefficient, and cost function. The fuzzy c median algorithm provides efficient grouping as the vast majority of its silhouette score values, obtained are 1, which can be inferred from the result of both the fuzzy C median using spark and fuzzy c means using spark. The majority of the rand index values obtained are closer to 1, which is clear evidence of successful data grouping for fuzzy c median using spark. Results for the cost function indicate that c=5 is the optimal number of clusters for this dataset, which is in line with the conclusions of our investigation. The proposed approach can be utilized to significantly reduce the time and memory consumption, while still maintaining clustering accuracy. The target research work endorses the significant potential performance enhancement to handle massive datasets with good clustering accuracy.

A. LIMITATIONS

Implementing Fuzzy C-Median clustering in Spark requires a good understanding of both the algorithm and Spark's programming model. The algorithm involves multiple iterations and updates to centroids and membership values, which must be efficiently distributed across the Spark cluster. Ensuring correctness and optimizing performance can be non-trivial. Spark's memory management is crucial here; improper configuration or insufficient memory allocation can lead to out-of-memory errors.

B. FUTURE ENHANCEMENT

Due to the rapid data generation that would result from the adoption of 5G technologies, a lot of research is needed to quickly handle big data analytics. Although many research works have proposed novel designs for clustering methods that take advantage of Big Data platforms, such as Apache Spark, which is designed for fast and distributed massive data processing, and despite the fact that clustering methods are significantly challenged by the recent massive growth of data, Spark-based clustering research is still in its infancy. Fuzzy C-Median using Spark is a powerful tool for processing large datasets and performing efficient clustering. In some situations, the Parallel Fuzzy C-Median Clustering Algorithm's efficacy could be increased by combining it with other machine learning or data mining methods. Subsequent research endeavours may investigate hybrid methodologies that optimise the advantages of distinct methods to enhance clustering efficacy. Future enhancements could focus on handling streaming data, improving the robustness to noisy data, incorporating domain knowledge, exploring different distance measures, and optimizing hyper parameters. These enhancements would make the algorithm even more effective and valuable for many real-world applications.

REFERENCES

[1] D. Reinsel, J. Gantz, and J. Rydning. *Data Age 2025: The Evolution of Data to Life-Critical*. Accessed: Oct. 3, 2010. [Online]. Available: https://assets.ey.com/content/dam/ey-sites/ey-com/en_gl/topics/workforce/Seagate-WP-DataAge2025-March-2017.pdf

[2] A. C. Ikegwu, H. F. Nweke, C. V. Anikwe, U. R. Alo, and O. R. Okonkwo, "Big data analytics for data-driven industry: A review of data sources, tools, challenges, solutions, and research directions," *Cluster Comput.*, vol. 25, no. 5, pp. 3343–3387, Oct. 2022, doi: 10.1007/s10586-022-03568-5.

[3] M. M. Saeed, Z. Al Aghbari, and M. Alsharidah, "Big data clustering techniques based on spark: A literature review," *PeerJ Comput. Sci.*, vol. 6, p. e321, Nov. 2020, doi: 10.7287/peerj-cs.321v0.1/reviews/3.

[4] R. Cooley, B. Mobasher, and J. Srivastava, "Web mining: Information and pattern discovery on the world wide web," in *Proc. 9th IEEE Int. Conf. Tools Artif. Intell.*, Nov. 1997, pp. 558–567.

[5] Z. Ansari, M. F. Azeem, A. V. Babu, and W. Ahmed, "A fuzzy approach for feature evaluation and dimensionality reduction to improve the quality of web usage mining results," *Int. J. Adv. Sci., Eng. Inf. Technol.*, vol. 2, no. 6, pp. 67–73, 2012, doi: 10.18517/ijaseit.2.6.248.

[6] G. Castellano, F. Mesto, M. Minunno, and M. A. Torsello, "Web user profiling using fuzzy clustering," in *Applications of Fuzzy Sets Theory* (Lecture Notes in Computer Science), vol. 4578, F. Masulli, S. Mitra, and G. Pasi, Eds., Berlin, Germany: Springer, 2007, pp. 94–101.

[7] O. Nasraoui, H. Frigui, R. Krishnapuram, and A. Joshi, "Extracting web user profiles using relational competitive fuzzy clustering," *Int. J. Artif. Intell. Tools*, vol. 9, no. 4, pp. 509–526, Dec. 2000.

[8] I. H. Sarker, "Machine learning: Algorithms, real-world applications and research directions," *Social Netw. Comput. Sci.*, vol. 2, no. 3, p. 160, May 2021, doi: 10.1007/s42979-021-00592-x.

[9] M. Bendechache, A.-K. Tari, and M.-T. Kechadi, "Parallel and distributed clustering framework for big spatial data mining," *Int. J. Parallel, Emergent Distrib. Syst.*, vol. 34, no. 6, pp. 671–689, Mar. 2018.

[10] Z. Ansari, A. V. Babuy, W. Ahmed, and M. F. Azeemz, "A fuzzy set theoretic approach to discover user sessions from web navigational data," in *Proc. IEEE Recent Adv. Intell. Comput. Syst.*, Sep. 2011, pp. 879–884.

[11] M. Khalid and M. M. Yousaf, "A comparative analysis of big data frameworks: An adoption perspective," *Appl. Sci.*, vol. 11, no. 22, p. 11033, Nov. 2021, doi: 10.3390/app112211033.

[12] Z. Milosevic, W. Chen, A. Berry, F. A. Rabhi, R. Buyya, R. N. Calheiros, and A. V. Dastjerdi, "Real-time analytics," *Big Data, Princ. Paradigms*, vol. 2016, pp. 39–61, Jan. 2016, doi: 10.1016/b978-0-12-805394-2.00002-7.

[13] N. Saeed and L. Husamaldin, "Big data characteristics (V's) in industry," *Iraqi J. Ind. Res.*, vol. 8, no. 1, pp. 1–9, Jun. 2021, doi: 10.53523/ijoir-vol8i1id52.

[14] S. Kumar and K. K. Mohbey, "A review on big data based parallel and distributed approaches of pattern mining," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 34, no. 5, pp. 1639–1662, May 2022, doi: 10.1016/j.jksuci.2019.09.006.

[15] P. Neumann and J. Kunkel, "High-performance techniques for big data processing," in *Knowledge Discovery in Big Data From Astronomy and Earth Observation*. Finland: Springer, 2020, pp. 137–158, doi: 10.1016/b978-0-12-819154-5.00017-5.

[16] J. Han, M. Kamber, and J. Pei, *Data Mining Concepts and Techniques*, 3rd ed. Waltham, MA, USA: Morgan Kaufmann Publishers, 2012.

[17] W. U. Chunqiong, "Research on clustering algorithm based on big data," *IOP Conf. Ser., J. Phys., Conf. Ser.*, vol. 1237, Jan. 2019, Art. no. 022131.

[18] G. Castellano, A. M. Fanelli, and M. A. Torsello, "Mining usage profiles from access data using fuzzy clustering," in *Proc. 6th WSEAS Int. Conf. Simul., Model. Optim.*, 2006, pp. 157–160.

[19] P. S. M. Mujeeb, R. PraveenSam, and K. Madhavi, "An inception towards better big data clustering technique," *Int. J. Eng. Adv. Technol. (IJEAT)*, vol. 8958, pp. 2249–8958, Jan. 2019.

[20] K.-H. Chen, H.-Y. Chen, and C.-M. Wang, "Bucket MapReduce: Relieving the disk I/O intensity of data-intensive applications in MapReduce frameworks," in *Proc. 29th Euromicro Int. Conf. Parallel, Distrib. Network-Based Process. (PDP)*, Mar. 2021, pp. 18–25, doi: 10.1109/PDP52278.2021.00013.

[21] N. Zahid, M. Limouri, and A. Essaid, "A new cluster-validity for fuzzy clustering," *Pattern Recognit.*, vol. 32, no. 7, pp. 1089–1097, Jul. 1999, doi: 10.1016/s0031-3203(98)00157-5.

[22] M. Pietrzykowski, "Comparison of mini-models based on various clustering algorithms," *Proc. Comput. Sci.*, vol. 176, pp. 3563–3570, Jan. 2020, doi: 10.1016/j.procs.2020.09.030.

[23] M. S. Y. Fukuyama, "A new method of choosing the number of clustersfor fuzzy c-means method," in *Proc. 5th Fuzzy Syst. Symp.*, 1989, pp. 247–250.

[24] J. C. Bezdek, R. Ehrlich, and W. Full, "FCM: The fuzzy c-means clustering algorithm," *Comput. Geosci.*, vol. 10, nos. 2–3, pp. 191–203, Jan. 1984.

[25] S. Fong, S. Deb, X.-S. Yang, and Y. Zhuang, "Towards enhancement of performance of K-means clustering using nature-inspired optimization algorithms," *Sci. World J.*, vol. 2014, pp. 1–16, Jan. 2014, doi: 10.1155/2014/564829.

[26] Z. Ansari, A. R. Faizabadi, and A. Afzal, "Fuzzy c-least medians clustering for discovery of web access patterns from web user sessions data," *Intell. Data Anal.*, vol. 21, no. 3, pp. 553–575, Jun. 2017, doi: 10.3233/ida-150489.

[27] N. Ahmed, A. L. C. Barczak, T. Susnjak, and M. A. Rashid, "A comprehensive performance analysis of apache Hadoop and apache spark for large scale data sets using HiBench," *J. Big Data*, vol. 7, no. 1, p. 110, Dec. 2020, doi: 10.1186/s40537-020-00388-5.

[28] Y. Liu, X. Zhang, J. Chen, and H. Chao, "A validity index for fuzzy clustering based on bipartite modularity," *J. Electr. Comput. Eng.*, vol. 2019, pp. 1–9, Aug. 2019, doi: 10.1155/2019/2719617.

[29] S. Theodoridis and K. Koutroumbas, "Cluster validity," *Pattern Recognit.*, vol. 2, pp. 863–913, Dec. 2016, doi: 10.1016/b978-1-59749-272-0.50018-9.

[30] K. R. G. Hewawasam, K. Premaratne, and M. L. Shyu, "Rule mining and classification in a situation assessment application: A belief-theoretic approach for handling data imperfections," *IEEE Trans. Syst., Man, Cybern., B*, vol. 37, no. 6, pp. 1446–1459, Dec. 2007.

[31] S. Liang, D. Han, and Y. Yang, "Cluster validity index for irregular clustering results," *Appl. Soft Comput.*, vol. 95, Oct. 2020, Art. no. 106583, doi: 10.1016/j.asoc.2020.106583.

[32] C. Yuan and H. Yang, "Research on K-value selection method of K-means clustering algorithm," *J*, vol. 2, no. 2, pp. 226–235, Jun. 2019, doi: 10.3390/j2020016.

[33] P. Fränti and S. Sieranoja, "How much can k-means be improved by using better initialization and repeats?" *Pattern Recognit.*, vol. 93, pp. 95–112, Sep. 2019, doi: 10.1016/j.patcog.2019.04.014.

[34] J. Zhang, G. Wu, X. Hu, S. Li, and S. Hao, "A parallel clustering algorithm with MPI—MKmeans," *J. Comput.*, vol. 8, no. 1, pp. 10–17, Jan. 2013.

[35] C. Sreedhar, N. Kasiviswanath, and P. C. Reddy, "Clustering large datasets using K-means modified inter and intra clustering (KM-I2C) in Hadoop," *J. Big Data*, vol. 4, no. 1, p. 27, Dec. 2017.

[36] T. D. Khang, N. D. Vuong, M.-K. Tran, and M. Fowler, "Fuzzy c-means clustering algorithm with multiple fuzzification coefficients," *Algorithms*, vol. 13, no. 7, p. 158, Jun. 2020, doi: 10.3390/a13070158.

[37] D. Littau and D. Boley, "Clustering very large data sets using a low memory matrix factored representation," *Comput. Intell.*, vol. 25, no. 2, pp. 114–135, May 2009.

[38] H. Xing, H. Chen, H. Lin, and X. Wu, "An interval type-2 possibilistic c-means clustering algorithm and its application," *J. Phys., Conf.*, vol. 2132, no. 1, Dec. 2021, Art. no. 012016, doi: 10.1088/1742-6596/2132/1/012016.

[39] S. Zhang and C. Duan, "Clustering optimization algorithm for data mining based on artificial intelligence neural network," *Wireless Commun. Mobile Comput.*, vol. 2022, pp. 1–16, Feb. 2022, doi: 10.1155/2022/1304951.

[40] Y. He, H. Tan, W. Luo, H. Mao, D. Ma, S. Feng, and J. Fan, "MR-DBSCAN: An efficient parallel density-based clustering algorithm using MapReduce," in *Proc. IEEE 17th Int. Conf. Parallel Distrib. Syst.*, Dec. 2011, pp. 473–480.

[41] A. Fahim, "A clustering algorithm based on local density of points," *Int. J. Modern Educ. Comput. Sci.*, vol. 9, no. 12, pp. 9–16, Dec. 2017, doi: 10.5815/ijmecs.2017.12.02.

[42] O. Akman, T. Comar, D. Hrozencik, and J. Gonzales, "Data clustering and self-organizing maps in biology," in *Algebraic and Combinatorial Computational Biology*. Cambridge, MA, USA: Academic, 2019, pp. 351–374, doi: 10.1016/b978-0-12-814066-6.00011-8.

[43] X.-Y. Li and L.-J. Guo, "Constructing affinity matrix in spectral clustering based on neighbor propagation," *Neurocomputing*, vol. 97, pp. 125–130, Nov. 2012, doi: 10.1016/j.neucom.2012.06.023.

[44] J. B. Schafer, D. Frankowski, J. Herlocke, and S. Sen, "Collaborative filtering recommender systems," in *The Adaptive Web*. Berlin, Germany: Springer-Verlag, 2007, pp. 291–324, doi: 10.1007/978-3-540-72079-9_9.

[45] M. A. Mallik, N. F. Zulkurnain, S. K. J. Ahmed, and M. K. Nizamuddin, "Identification of interested web users using decision tree classifier," in *Advances in Electrical and Computer Technologies* (Lecture Notes in Electrical Engineering). Singapore: Springer, 2021, pp. 143–156, doi: 10.1007/978-981-15-9019-1_13.

[46] S. Dash, S. K. Shakyawar, M. Sharma, and S. Kaushik, "Big data in healthcare: Management, analysis and future prospects," *J. Big Data*, vol. 6, no. 1, pp. 1–25, Dec. 2019, doi: 10.1186/s40537-019-0217-0.

[47] A. Oussous, F.-Z. Benjelloun, A. Ait Lahcen, and S. Belfkih, "Big data technologies: A survey," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 30, no. 4, pp. 431–448, Oct. 2018, doi: 10.1016/j.jksuci.2017.06.001.

[48] B. Liu, S. He, D. He, Y. Zhang, and M. Guizani, "A spark-based parallel fuzzy c-means segmentation algorithm for agricultural image big data," *IEEE Access*, vol. 7, pp. 42169–42180, 2019.

[49] A. M. Sainju, D. Aghajarian, Z. Jiang, and S. Prasad, "Parallel grid-based colocation mining algorithms on GPUs for big spatial event data," *IEEE Trans. Big Data*, vol. 6, no. 1, pp. 107–118, Mar. 2020.

[50] N. Akthar, M. V. Ahamad, and S. Khan, "Clustering on big data using Hadoop MapReduce," in *Proc. IEEE Int. Conf. Comput. Intell. Commun. Netw. (CICN)*, 2015, pp. 789–795.

[51] A. Sinha and P. K. Jana, "A novel K-means based clustering algorithm for big data," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Sep. 2016, pp. 1875–1879.

[52] M. A. B. HajKacem, C.-E.-B. N'cir, and N. Essoussi, "One-pass MapReduce-based clustering method for mixed large scale data," *J. Intell. Inf. Syst.*, vol. 52, no. 3, pp. 619–636, Jun. 2019.

[53] M. O. Shafiq and E. Torunski, "A parallel K-medoids algorithm for clustering based on MapReduce," in *Proc. 15th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2016, pp. 502–507.

[54] M. A. Ben Haj Kacem, C.-E. Ben N'cir, and N. Essoussi, "MapReduce-based k-prototypes clustering method for big data," in *Proc. IEEE Int. Conf. Data Sci. Adv. Analytics (DSAA)*, Oct. 2015, pp. 1–7.

[55] S. A. Ludwig, "MapReduce-based fuzzy c-means clustering algorithm: Implementation and scalability," *Int. J. Mach. Learn. Cybern.*, vol. 6, no. 6, pp. 923–934, Dec. 2015.

[56] M. S. Hidri, M. A. Zoghlami, and R. Ben Ayed, "Speeding up the large-scale consensus fuzzy clustering for handling big data," *Fuzzy Sets Syst.*, vol. 348, pp. 50–74, Oct. 2018.

[57] Q. Zhang and Z. Chen, "A weighted kernel possibilistic c-means algorithm based on cloud computing for clustering big data," *Int. J. Commun. Syst.*, vol. 27, no. 9, pp. 1378–1391, Sep. 2014.

[58] Q. Zhang, L. T. Yang, Z. Chen, and P. Li, "PPHOPCM: Privacy-preserving high-order possibilistic c-means algorithm for big data clustering with cloud computing," *IEEE Trans. Big Data*, vol. 8, no. 1, pp. 25–34, Feb. 2022.

[59] R. Hu, W. Dou, and J. Liu, "ClubCF: A clustering-based collaborative filtering approach for big data application," *IEEE Trans. Emerg. Topics Comput.*, vol. 2, no. 3, pp. 302–313, Sep. 2014.

[60] V. Subramaniyaswamy, V. Vijayakumar, R. Logesh, and V. Indragandhi, "Unstructured data analysis on big data using MapReduce," *Proc. Comput. Sci.*, vol. 50, pp. 456–465, Jan. 2015.

[61] G. Castellano, A. M. Fanelli, C. Mencar, and M. A. Torsello, "Similarity-based fuzzy clustering for user profiling," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell. Intell. Agent Technol.-Workshops*, Nov. 2007, pp. 75–78.

[62] P. Sachar and V. Khullar, "Social media generated big data clustering using genetic algorithm," in *Proc. Int. Conf. Comput. Commun. Informat. (ICCCI)*, Jan. 2017, pp. 1–6.

[63] J. Karimov and M. Ozbayoglu, "High quality clustering of big data and solving empty-clustering problem with an evolutionary hybrid algorithm," in *Proc. IEEE Int. Conf. Big Data*, Oct. 2015, pp. 1473–1478.

[64] Y. Yang, F. Teng, T. Li, H. Wang, H. Wang, and Q. Zhang, "Parallel semi-supervised multi-ant colonies clustering ensemble based on MapReduce methodology," *IEEE Trans. Cloud Comput.*, vol. 6, no. 3, pp. 857–867, Jul. 2018.

[65] M. Chen, S. A. Ludwig, and K. Li, "Clustering in big data," in *Big Data: Management, Architecture, and Processing*. Boca Raton, FL, USA: CRC Press, 2017, ch. 16, pp. 246–331.

[66] A. E. Ezugwu, A. M. Ikotun, O. O. Oyelade, L. Abualigah, J. O. Agushaka, C. I. Eke, and A. A. Akinyelu, "A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects," *Eng. Appl. Artif. Intell.*, vol. 110, Apr. 2022, Art. no. 104743, doi: 10.1016/j.engappai.2022.104743.

[67] Y. K. Dwivedi, E. Ismagilova, D. L. Hughes, J. Carlson, R. Filieri, J. Jacobson, V. Jain, H. Karjaluoto, H. Kefi, A. S. Krishen, V. Kumar, M. M. Rahman, R. Raman, P. A. Rauschnabel, J. Rowley, J. Salo, G. A. Tran, and Y. Wang, "Setting the future of digital and social media marketing research: Perspectives and research propositions," *Int. J. Inf. Manage.*, vol. 59, Aug. 2021, Art. no. 102168, doi: 10.1016/j.ijinfomgt.2020.102168.

[68] O. Nasraoui and R. Krisnapuram, "Clustering using a genetic fuzzy least median of squares algorithm," in *Proc. Annu. Meeting North Amer. Fuzzy Inf. Process. Soc.*, vol. 3, 1997, pp. 217–221, doi: 10.1109/nafips.1997.624040.

[69] E. Shaikh, I. Mohiuddin, Y. Alufaisan, and I. Nahvi, "Apache spark: A big data processing engine," in *Proc. 2nd IEEE Middle East North Afr. Commun. Conf. (MENACOMM)*, Nov. 2019, pp. 1–6, doi: 10.1109/MENACOMM46666.2019.8988541.

[70] A. S. Mavai and S. K. Mishra, "A survey and comparative study of different data mining techniques to implement a missing value estimator system," *Int. J. Current Eng. Technol.*, vol. 4, no. 4, Aug. 2014.

[71] Z. Ansari, M. F. Azeem, A. V. Babu, and W. Ahmed, "A fuzzy clustering based approach for mining usage profiles from web log data," *Int. J. Comput. Sci. Inf. Secur.*, vol. 9, pp. 70–79, Jun. 2011.

[72] G. Peters, F. Crespo, P. Lingras, and R. Weber, "Soft clustering—Fuzzy and rough approaches and their extensions and derivatives," *Int. J. Approx. Reasoning*, vol. 54, no. 2, pp. 307–322, Feb. 2013.

[73] L. A. Zadeh, "Recent developments and new directions in soft computing," in *Studies in Fuzziness and Soft Computing*, vol. 317. Berlin, Germany: Springer, 2014.

[74] V. Davara and A. B. Upadhyay, "Comparison of soft computing techniques for the design of microstrip patch antenna: A review paper," *Int. J. Adv. Res. Electr., Electron. Instrum. Eng.*, vol. 3, Mar. 2014.

[75] X. L. Xie and G. Beni, "A validity measure for fuzzy clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, no. 8, pp. 841–847, Aug. 1991.

[76] A. M. Ikotun and A. E. Ezugwu, "Boosting k-means clustering with symbiotic organisms search for automatic clustering problems," *PLoS ONE*, vol. 17, no. 8, Aug. 2022, Art. no. e0272861, doi: 10.1371/journal.pone.0272861.

[77] E. A. Zanaty, "Determining the number of clusters for kernelized fuzzy C-means algorithms for automatic medical image segmentation," *Egyptian Informat. J.*, vol. 13, no. 1, pp. 39–58, Mar. 2012, doi: 10.1016/j.eij.2012.01.004.

[78] K. V. Rajkumar, A. Yesubabu, and K. Subrahmanyam, "Fuzzy clustering and fuzzy c-means partition cluster analysis and validation studies on a subset of citescore dataset," *Int. J. Electr. Comput. Eng. (IJECE)*, vol. 9, no. 4, p. 2760, Aug. 2019, doi: 10.11591/ijece.v9i4.pp2760-2770.

[79] A. Haleem, M. Javaid, M. A. Qadri, R. P. Singh, and R. Suman, "Artificial intelligence (AI) applications for marketing: A literature-based study," *Int. J. Intell. Netw.*, vol. 3, pp. 119–132, Jan. 2022, doi: 10.1016/j.ijin.2022.08.005.

[80] W. Khalil, H. Torkey, and G. Attiya, "Survey of apache spark optimized job scheduling in big data," *Int. J. Ind. Sustain. Develop.*, vol. 1, no. 1, pp. 39–48, Jan. 2020, doi: 10.21608/ijisd.2020.73486.

[81] S. Armoogum and X. Li, "Big data analytics and deep learning in bioinformatics with Hadoop," in *Deep Learning and Parallel Computing Environment for Bioengineering Systems*. The Netherlands: Elsevier, 2019, pp. 17–36, doi: 10.1016/b978-0-12-816718-2.00009-9.

[82] M. T. Islam, S. N. Srirama, S. Karunasekera, and R. Buyya, "Cost-efficient dynamic scheduling of big data applications in apache spark on cloud," *J. Syst. Softw.*, vol. 162, Apr. 2020, Art. no. 110515, doi: 10.1016/j.jss.2019.110515.

[83] M. Petrov, N. Butakov, D. Nasonov, and M. Melnik, "Adaptive performance model for dynamic scaling apache spark streaming," *Proc. Comput. Sci.*, vol. 136, pp. 109–117, Jan. 2018, doi: 10.1016/j.procs.2018.08.243.

[84] S. A. K. Basha, S. M. Basha, D. R. Vincent, and D. S. Rajput, "Challenges in storing and processing big data using Hadoop and spark," in *Deep Learning and Parallel Computing Environment for Bioengineering Systems*. The Netherlands: Elsevier, 2019, pp. 179–187, doi: 10.1016/b978-0-12-816718-2.00018-x.

[85] *How to Use Spark Clusters for Parallel Processing Big Data*. Accessed: Dec. 3, 2021. [Online]. Available: https://www.freecodecamp.org/news/how-to-use-spark-clusters-for-parallel-processing-big-data-86a22e7f8b50/

[86] S. Ramírez-Gallego, S. García, J. M. Benítez, and F. Herrera, "A distributed evolutionary multivariate discretizer for big data processing on apache spark," *Swarm Evol. Comput.*, vol. 38, pp. 240–250, Feb. 2018.

[87] *Introduction to Apache Spark With Examples and Use Cases*. Accessed: Aug. 2022. [Online]. Available: https://www.toptal.com/spark/introduction-to-apache-spark

[88] *PySpark—SparkContext*. Accessed: Jul. 2022. [Online]. Available: https://www.tutorialspoint.com/pyspark/pyspark_sparkcontext.htm

[89] Accessed: Jul. 2022. [Online]. Available: https://www.datacamp.com/tutorial/pyspark-tutorial-getting-started-with-pyspark

[90] Accessed: Jul. 2022. [Online]. Available: https://www.edureka.co/blog/spark-architecture/

[91] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, and A. Ghodsi, "Apache spark: A unified engine for big data processing," *Commun. ACM*, vol. 59, pp. 56–65, Oct. 2016.

[92] *Ambari*. Accessed: Jul. 15, 2020. [Online]. Available: https://ambari.apache.org/

[93] A. Verma, A. H. Mansuri, and N. Jain, "Big data management processing with Hadoop MapReduce and spark technology: A comparison," in *Proc. Symp. Colossal Data Anal. Netw. (CDAN)*, vol. 4, Mar. 2016, pp. 1–4.

[94] G. K. Thiruvathukal, C. Christensen, X. Jin, F. Tessier, and V. Vishwanath, "A benchmarking study to evaluate apache spark on large-scale supercomputers," 2019, *arXiv:1904.11812*.

[95] S. Gopalani and R. Arora, "Comparing apache spark and map reduce with performance analysis using K-means," *Int. J. Comput. Appl.*, vol. 113, no. 1, pp. 8–11, Mar. 2015.

[96] S. Usman, R. Mehmood, I. Katib, and A. Albeshri, "Data locality in high performance computing, big data, and converged systems: An analysis of the cutting edge and a future system architecture," *Electronics*, vol. 12, no. 1, p. 53, Dec. 2022, doi: 10.3390/electronics12010053.

[97] Y. Zhao, J. Dong, H. Liu, J. Wu, and Y. Liu, "Performance improvement of DAG-aware task scheduling algorithms with efficient cache management in spark," *Electronics*, vol. 10, no. 16, p. 1874, Aug. 2021, doi: 10.3390/electronics10161874.

[98] T. Sterling, M. Anderson, and M. Brodowicz, "MapReduce," in *High Performance Computing*. The Netherlands: Elsevier, 2018, pp. 579–589, doi: 10.1016/b978-0-12-420158-3.00019-8.

[99] S. Aranganayagi and K. Thangavel, "Clustering categorical data using silhouette coefficient as a relocating measure," in *Proc. Int. Conf. Comput. Intell. Multimedia Appl. (ICCIMA)*, Dec. 2007, pp. 13–17, doi: 10.1109/iccima.2007.328.

[100] A. K. Paul and P. C. Shill, "New automatic fuzzy relational clustering algorithms using multi-objective NSGA-II," *Inf. Sci.*, vols. 448–449, pp. 112–133, Jun. 2018, doi: 10.1016/j.ins.2018.03.025.

[101] J. Pérez-Ortega, C. F. Moreno-Calderón, S. S. Roblero-Aguilar, N. N. Almanza-Ortega, J. Frausto-Solís, R. Pazos-Rangel, and J. M. Rodríguez-Lelis, "A new criterion for improving convergence of fuzzy C-means clustering," *Axioms*, vol. 13, no. 1, p. 35, Jan. 2024, doi: 10.3390/axioms13010035.

[102] J. Zhang and Z. Ma, "Hybrid fuzzy clustering method based on FCM and enhanced logarithmical PSO (ELPSO)," *Comput. Intell. Neurosci.*, vol. 2020, pp. 1–12, Mar. 2020, doi: 10.1155/2020/1386839.

[103] M. Shi, T. Zhang, L. Zhang, W. Sun, and X. Song, "A fuzzy c-means algorithm based on the relationship among attributes of data and its application in tunnel boring machine," *Knowl.-Based Syst.*, vol. 191, Mar. 2020, Art. no. 105229.

[104] C. Ren and L. Sun, "A bi-directional fuzzy c-means clustering ensemble algorithm considering local information," *Int. J. Comput. Intell. Syst.*, vol. 14, no. 1, pp. 1–14, Dec. 2021.

[105] S. M. Razavi, M. Kahani, and S. Paydar, "Big data fuzzy c-means algorithm based on bee colony optimization using an apache hbase," *J. Big Data*, vol. 8, no. 1, pp. 1–22, Dec. 2021, doi: 10.1186/s40537-021-00450-w.

[106] M. A. Al-Asadi and S. Tasdemir, "Using artificial intelligence against the phenomenon of fake news: A systematic literature review," in *Combating Fake News With Computational Intelligence Techniques* (Studies in Computational Intelligence), vol. 1001, M. Lahby, A. S. K. Pathan, Y. Maleh, W. M. S. Yafooz, Eds., Cham, Switzerland: Springer, 2022, doi: 10.1007/978-3-030-90087-8_2.

[107] M. K. Aiden, S. M. Sabharwal, S. Chhabra, and M. Al-Asadi, "AI and blockchain for cyber security in cyber-physical system," in *AI Models for Blockchain-Based Intelligent Networks in IoT Systems* (Engineering Cyber-Physical Systems and Critical Infrastructures), vol. 6, B. Bhushan, A. K. Sangaiah, T. N. Nguyen, Eds., Cham, Switzerland: Springer, 2023, doi: 10.1007/978-3-031-31952-5_10.

[108] M. A. Al-Asadi and S. Tasdemir, "Medical image analysis using machine learning techniques," in *Machine Learning and Deep Learning in Efficacy Improvement of Healthcare Systems*. U.K.: Taylor & Francis, 2022, pp. 137–153, doi: 10.1201/9781003189053-7.

[109] A. Kumari, A. Golyan, R. Shah, and N. Raval, "Introduction to data analytics," in *Recent Trends and Future Direction for Data Analytics*, A. Kumari, Ed., Hershey, PA, USA: IGI Global, 2024, pp. 1–14, doi: 10.4018/979-8-3693-3609-0.ch001.

MOKSUD ALAM MALLIK was born in Howrah, Kolkata, India, in June 1985. He received the B.Tech. (CSE) degree (Hons.) from the Maulana Abul Kalam Azad University of Technology, West Bengal, India, in 2005, the M.Tech. (CSE) degree (Hons.) from Visvesvaraya Technological University (VTU), Karnataka, India, in 2014, and the Ph.D. (CSE) degree from International Islamic University Malaysia (IIUM), Kuala Lumpur, Malaysia, in 2023. He joined as an Associate Professor and the HOD with the Department of Computer Science and Engineering-DS, LIET, Hyderabad, India, in August 2022, and also take over as the Dean Research and Development at LIET, from 2024. He has more than 17 years of teaching and industrial experience in reputed engineering colleges and renowned MNCs. He has five book chapters and 13 research publications in reputed journals (Scopus and WoS), conferences, proceedings of which are published by Springer and IEEE. His research interests include parallel clustering algorithm, big data analytics, datamining, machine learning, and data science.

NURUL FARIZA ZULKURNAIN was born in Malaysia. She received the Ph.D. degree in computer science from The University of Manchester, U.K., in 2012. She is currently an Associate Professor with the Electrical and Computer Engineering Department, Kulliyyah of Engineering, International Islamic University Malaysia. She has 19 years of teaching and research experience. As a part of the Software Engineering Research Group, her research interests are in the areas of big data, data mining, machine learning, and artificial intelligence.

SUMRANA SIDDIQUI was born in Tabuk, Saudi Arabia, in December 1985. She received the B.E. degree in computer science from Osmania University, Hyderabad, India, in 2007, and the M.Tech. degree in computer science from JNTU, Hyderabad, in 2014. She is currently pursuing the Ph.D. degree with the GITAM School of Technology, GITAM (Deemed to be University), Hyderabad. She is working with the Computer Science and Engineering Department, Deccan College of Engineering and Technology, as an Assistant Professor. Her research interests include machine learning, image segmentation, information security, and block chain technology.

RASHEL SARKAR was born in Kakripara, Assam, India, in June 1983. He received the Engineering degree from the Biju Patnaik University of Technology, Odisha, India, the M.Tech. (CSE) degree from the R.V. College of Engineering, Bengaluru, India, and the Ph.D. degree from Himalayan Garhwal University, Uttarakhand, India. He has teaching and administrative experience of more than 19 years at college and university level. Currently, he is an Associate Professor with the Department of Computer Science and Engineering, The Assam Royal Global University, Guwahati, India. He has more than 20 publications in reputed international journals and conferences. His expertise is in data mining, data science, machine learning, and artificial intelligent.