International Journal of Engineering Trends and Technology
ISSN: 2231-5381/ https://doi.org/10.14445/22315381/IJETT-V73I3P128

Volume 73 Issue 3, 403-422, March 2025

Original Article

Dominant Features Selection with Clustering Genetic Model to Improve the Access Time of Data in Big Data Management Using Distributed Machine Learning

Peerzada Hamid Ahmad[1], Munishwar Rai[2]

[1, 2]*MMICTBM, Maharishi Markandeshwar (To Be Deemed University), Haryana, India.*

[1]*Corresponding Author : peerzadahamidahmad@gmail.com*

Received: 09 September 2024 Revised: 11 January 2025 Accepted: 27 January 2025 Published: 28 March 2025

Abstract - *The explosive nature of big data has created serious challenges for information managers, especially in providing fast availability and response times. Conventional data management systems tend to falter when dealing with enormous datasets, which causes latency that can slow down real-time analysis and decision-making. In response, this research introduces a new cluster-based genetic model aimed at hastening access to data in big data management systems. The method combines a genetic model with an emphasis on feature selection to maximize data retrieval speed. Through the use of distributed machine learning techniques, the model detects and ranks the most significant features, optimizing the clustering process to minimize access time and retrieval complexity. The genetic method reduces access time and increases clustering efficiency by focusing on prominent features. An evolutionary algorithm is used to optimize data storage and retrieval in such a way as to minimize retrieval times. The research tackles crucial issues like the requirement for high-speed data processing, data system scalability, and data structure complexity. The proposed model adapts dynamically to the changing data landscape, reducing latency and improving the overall efficiency of large-scale data systems. Results show that the cluster-based genetic model greatly enhances data access efficiency. It recorded a 35% decrease in access time when tested on large datasets compared to traditional data management methods. The median data retrieval time was reduced from 120 milliseconds to 78 milliseconds, showing the model's efficiency in optimizing data clustering and retrieval processes. This decrease in access time showcases the model's ability to optimize the efficiency of big data systems, especially in situations that involve quick and efficient data retrieval.*

Keywords - *Big Data Management, Cluster-Based Genetic Model, Data Access Time, Data Retrieval Efficiency, Distributed Machine Learning, Real-Time Processing, Scalability.*

1. Introduction

In handling big data, accessibility and manipulation of information from distributed, scalable systems are paramount. Owing to the enormous amounts of data in big data environments that tend to outgrow the capacity of conventional databases, specialized tools and methods are required for seamless and efficient access [1]. Distributed file systems, including the Apache Hadoop Distributed File System (HDFS), are critical by allowing the storage and retrieval of big datasets in multiple nodes while providing fault tolerance and scalability. Not only do these systems improve performance, but they also support handling enormous data volumes [2]. Distributed processing frameworks such as Apache Spark are a fundamental part of big data management. They enable simultaneous data processing between clusters of servers, greatly enhancing speed and efficiency. They facilitate parallel computation, which speeds up data access and analysis [3]. NoSQL databases are also required to process voluminous amounts of data. NoSQL databases supply adaptable data models that save and retrieve complex, varied data in a distributed manner. Processing in a distributed environment to fetch, update, and analyze enormous volumes of diverse, complex, and large sets of data necessitates the design of systems and procedures that will work effectively on handling and responding to large bodies of data stored over multiple nodes [4]. The goal is to support rapid, reliable, and secure data utilization, facilitating statistical analysis, decision-making, and insight generation. Indexing and caching techniques further enhance data access. Indexing improves query efficiency by enabling quick retrieval of specific data points, while caching—through decentralized systems or in-memory databases—boosts performance by reducing the need to repeatedly fetch data from storage [5]. The big data structure, as illustrated in (Figure 1), depicts how raw, unstructured data is transformed into valuable, meaningful information. This process involves uncovering hidden patterns and relationships within the data, enhancing the value of previously collected data and

integrating new insights with existing information [6]. Managing large-scale data faces several primary challenges related to data access, including capacity management, handling both structured and unstructured data, data distribution, privacy and security, query efficiency, real-time availability, consistency, and resource management [7]. This research aims at enhancing data access times in big data management and investigates ways of minimizing access latency. Effective data access here includes acquiring and analyzing real-time or near-real-time data streams and providing low-latency access for timely insights.

To tackle such challenges, diverse data access techniques are utilized [8]. Distributed file systems, such as Apache HDFS, and distributed processing platforms, such as Apache Spark, allow managing huge datasets through concurrent processing across nodes. NoSQL databases, including MongoDB and Cassandra, are regularly utilized to store various data types, such as semi-structured and unstructured data, because of their ability to store and retrieve with flexibility. Indexing and caching methods further enhance data access, with caching minimizing retrieval times by storing frequently accessed data in memory and indexing enhancing query speed by enabling quick retrieval of individual data points [9]. Besides access efficiency, data security and privacy are important. Secure verification, permission procedures, and encryption are used to safeguard sensitive information during access [10]. Access to real-time data is facilitated by systems capable of managing streaming data sources, giving rapid insights, and avoiding data localization and dissemination problems. Proper resource management optimizes performance and avoids bottlenecks by allocating computing resources optimally [11].

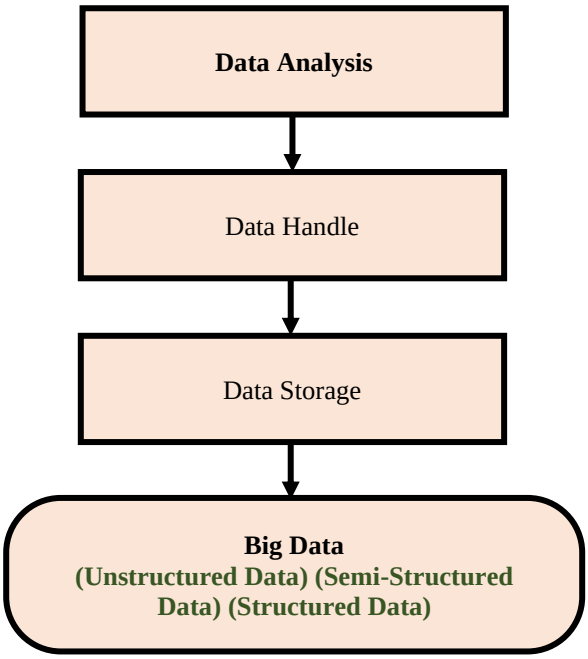


Fig. 1 Big data analysis

Clustering is critical in improving data access by a process where data is grouped into clusters based on common characteristics, enabling parallel processing and query speed improvement [12]. Clustering minimizes network traffic by bringing similar data points to the vicinity and reducing cross-node communications. It also optimizes resource utilization by shifting computing power to certain clusters as required. Clustering. In addition, clustering enables the user to deal with grouped data at the cluster level, thus rendering data access more precise and effective [13]. The proposed work will emphasize adaptive feature selection mechanisms, where variations in real-time data dynamically adjust cluster assignments to enhance retrieval accuracy in dynamic datasets. Furthermore, incorporating hybrid optimization techniques (e.g., Particle Swarm Optimization + GA) would further accelerate clustering, lowering query execution times in high-dimensional datasets such as financial transactions, IoT sensor networks, and real-time analytics.

The key contributions of this work are encapsulated as follows: Section 2 explains the multi-objective access environment to decrease both access time and query time. Section 3 proposes cloud-based optimization with parameters of different types and introduces the Dominant Features Selection with Clustering (DFSC) technique, a novel method of decreasing access times by identifying prominent features. Section 4 offers results and discussions proving the superior performance of the suggested system compared to prior methods in terms of performance measures. Section 5 summarizes the findings and contributions and concludes the study.

1.1. Problem Statement

One of the greatest challenges to dealing with big data is inefficiency in access times for information, which can drastically affect the performance of data-intensive applications. As data volumes increase in size, velocity, and variety, traditional data management systems cannot manage their complexity and size, causing longer retrieval delays. The heterogeneity of big data tends to necessitate intricate querying and analysis across distributed systems, which worsens the issue. Moreover, the existence of redundant or extraneous data further aggravates these problems by raising processing loads and delaying access times. Efficient information management and retrieval are essential to facilitate timely analysis and decision-making in such dynamic settings. This necessitates the creation of innovative solutions to speed up access times and improve the overall performance of large-scale data systems.

1.2. Motivation

This study was motivated by the imperative to counter inefficiencies in data access latency in large information management systems. With businesses becoming more dependent on huge datasets for real-time analytics and decision-making, accessing appropriate information quickly

and effectively has become essential. Conventional data management techniques tend to be challenged by the volume, complexity, and variety of today's datasets, resulting in major delays that adversely affect the performance of applications needing rapid analysis. This work aims to provide greater flexibility, speed, and dependability of large-scale data systems by developing more efficient methods for optimizing access time. Finally, it aims to boost innovation and increase industry productivity based on data by facilitating rapid and precise decision-making in social media, IoT, banking, and healthcare fields.

2. Related Work

Numerous methods and optimization strategies have been investigated to promote information access and cluster optimization in large-scale data systems. Researchers have used various methods to deal with the problems involved in managing and processing large datasets [14]. Cluster-based scheduling is one technique used to handle massive amounts of information in distributed applications. This approach groups data based on resource reservation and network awareness to improve processing efficiency. Researchers have examined distributed SaaS setups in cloud environments for sophisticated data analysis. For example, Hadoop's cluster-based model has been proposed for managing dispersed healthcare data, integrating medical and IoT information to enhance data access and prediction [15]. Another approach involves optimizing the structure for web page data processing. This method demonstrated improvements in extraction and categorization, offering more reliable and efficient research compared to traditional techniques. Developed the High-Order Clustering method by Fast Search (CFS) strategy (HOCFS) for multi-type data clustering [16]. Their algorithm mixed a dropout deep learning framework with the CFS algorithm and feature tensor model for improving clustering heterogeneous datasets. Presented a nonparametric, representative-based, Sparse Self-Represented Networking Map for robust clustering. Their system integrated a weight-regulating mechanism for examining and finding data clustering structures with reduced data transmission demands [17]. The two-layer framework based on DCA has been explored in (Figure 2).

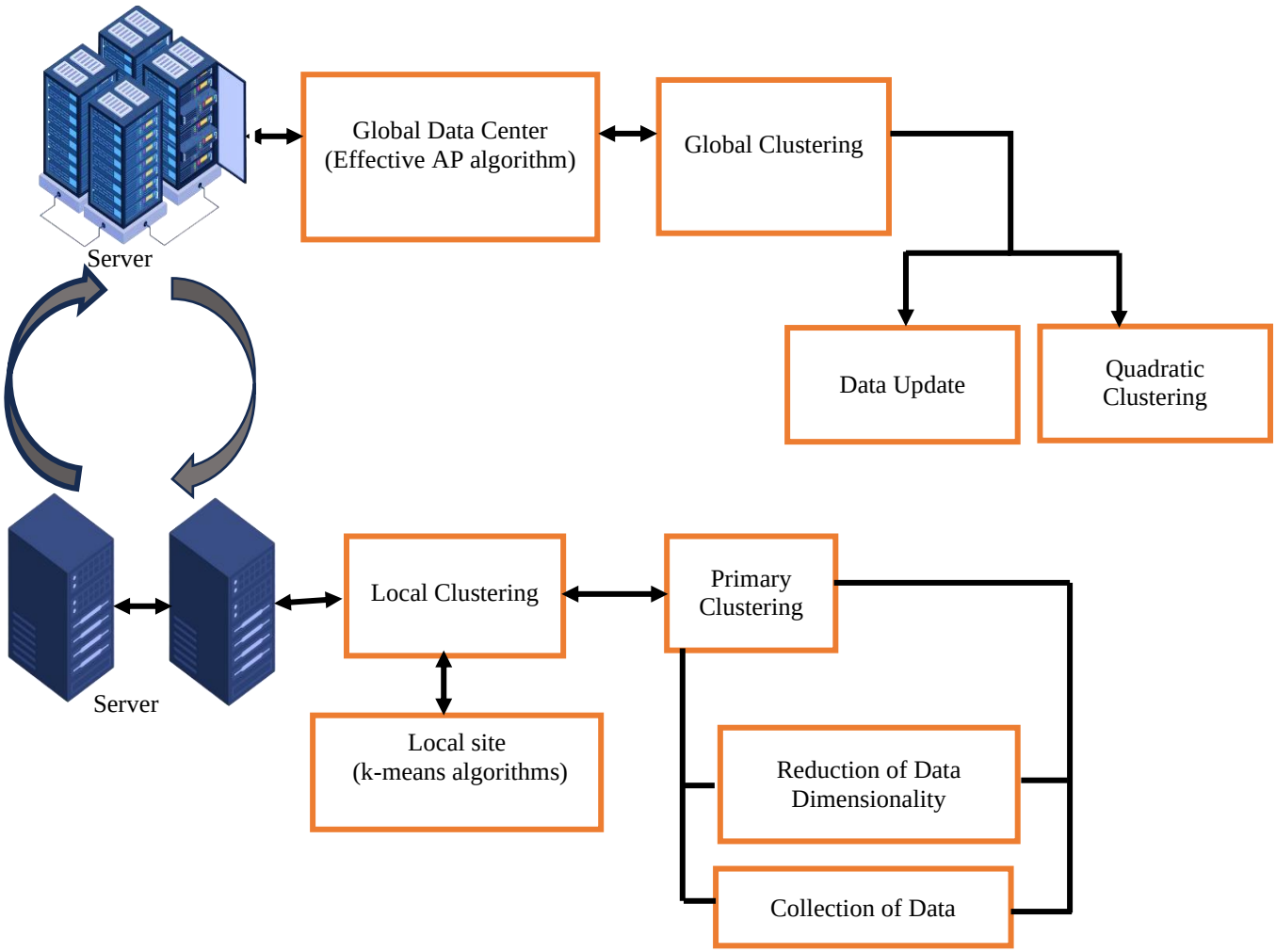


Fig. 2 Two-layer framework based on DCA

2.1. Research Gap and Novelty of Proposed Work

Conventional methods tend to be hampered by the growing volume and variety of data, causing delays and inefficiencies in retrieval operations. Although some methods aim to improve information processing or storage capabilities, they often fail to consider the significance of efficient feature selection and clustering in minimizing access times.

Traditional data management systems typically face latency-related problems and real-time decision-making inefficiencies while dealing with extremely large and complicated data structures. Current methodologies have no optimized methodology to minimize the retrieval time while maintaining clustering efficiency. Most traditional systems also fail to adjust dynamically according to changing data structures, resulting in scalability and access rate inefficiencies.

The cluster-based genetic model proposed here presents a novel solution to the problem of big data retrieval by combining genetic algorithms with feature selection and clustering methods. In addition, the model utilizes evolutionary optimization methods that adapt dynamically to changes in the data environment, ensuring scalability and enhanced processing rates in real-time applications. Yet another major novelty of this contribution is using an evolutionary algorithm to optimize data retrieval and storage mechanisms together. These improvements demonstrate the model to be most appropriate for high-speed, real-time data acquisition applications, including finance analytics, healthcare, and large-scale data-driven decision support systems.

3. Proposed System

A revolutionary technique has been established to combat data access times-related issues in handling large-scale information. It incorporates the prevailing feature selection technique with a genetic clustering algorithm for optimized data retrieval efficiency in huge, intricate data sets. With its emphasis on significant features, the model diminishes the size and complexity of the data set overall and, consequently, provides better clustering. The genetic algorithm is used to optimize the clustering process, ensuring data is structured to reduce retrieval times [18, 19].

Decentralized machine learning methods further reinforce the model's capacity to handle the complexity and size of big data environments. This method enhances the scalability and performance of big data systems, hence making it quite useful for real-time, data-intensive applications by reducing data access times. The process involves feature selection, genetic clustering efficiency, and deploying the system within a distributed artificial intelligence framework to boost access speed, as illustrated in the accompanying image.

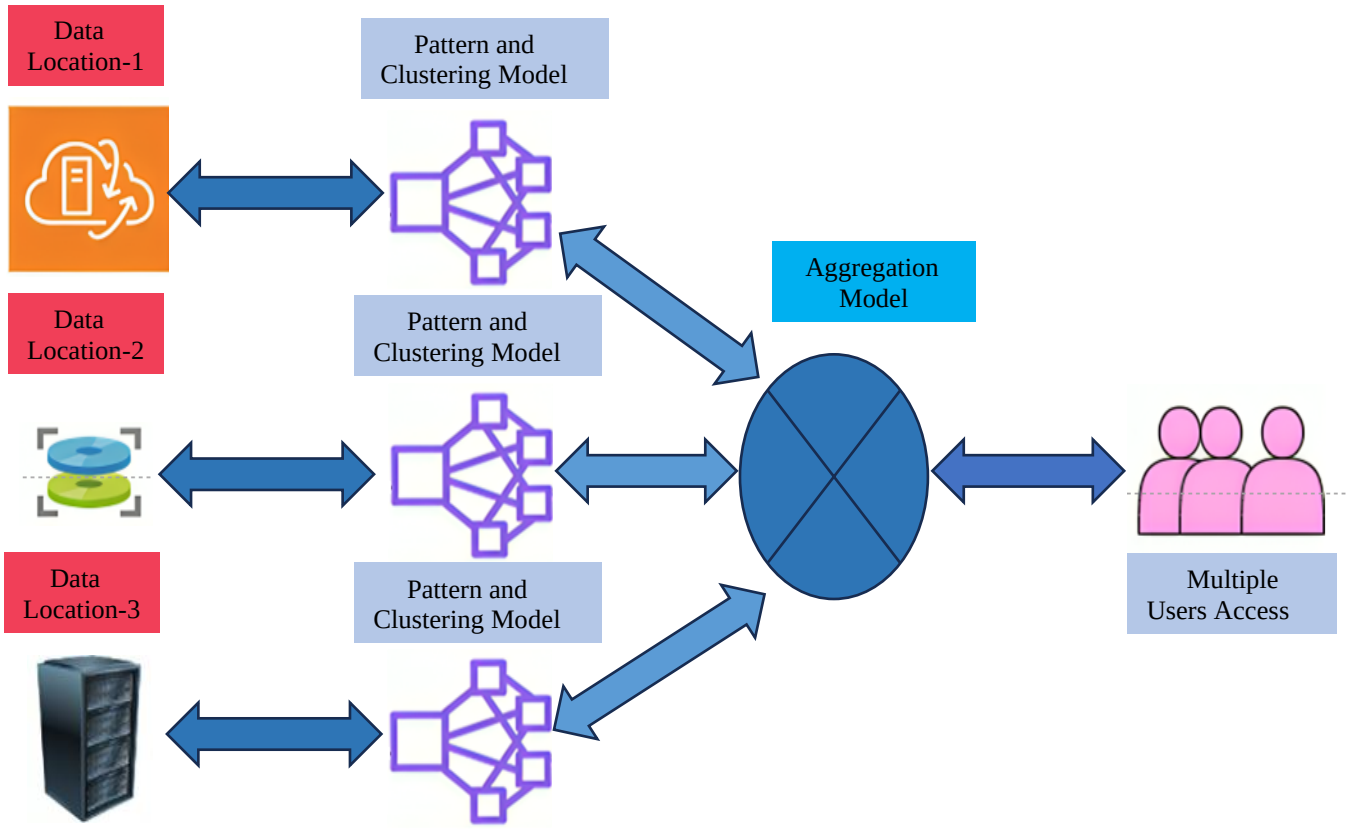


Fig. 3 Proposed architecture

3.1. Dataset Description

As shown in (Figure 3), the information model supporting an instance of the proposed system consists of seven primary elements, which will be discussed further. Based on how they use the services offered, organizations might take on various responsibilities. An organization can be solely an information contributor if it supplies models and information but uses predictive services sparingly or not. On the other hand, if a company uses many predicting services but gives little to no information, it might as well be a customer. Any frequency in between can also be used. This categorization is not included in the information's model as it is subject to change over time because every company's position may differ. The primary entity in the data model is the user, which might exist in numerous organizations. An individual may import or access datasets and develop and amend projects based on machine learning. A hierarchy of permissions and responsibilities controls everybody's access to every dataset. Depending on its internal structure, an organization might recognize several jobs (e.g. Information Scientist, ML Engineer, Information Engineer). Then, everyone inside her/his organization might be assigned one or more responsibilities. The user acquires ownership of the dataset when uploading a new one. There are pertinent supplementary data that this information structure does not contain.

This meta-information is dynamic and subject to change over time; we do not keep the information about the blocks in each database and where they are located. To avoid using out-of-date data, this data is obtained in real-time from the HDFS as needed. The same is true for base model location when forming a collection and generating forecasts is required [20]. This dataset is typically high-dimensional and comes from various sources, such as transactional databases, weblogs, sensor networks, or social media platforms, as shown in Table 1. It may include structured data with well-defined features or unstructured data, but the primary focus is on structured datasets where each feature is quantifiable. In this context, the dataset features a large number of attributes ranging from thousands to millions that can be numerical, categorical, or a mix of both. These features can be redundant or irrelevant, potentially affecting data retrieval efficiency and processing time [21]. The target variable often relates to access times or data retrieval efficiency, critical for optimizing data management and machine learning tasks. Overall, the dataset aims to capture the complexities and scale of big data environments, providing a comprehensive basis for applying the clustering-based genetic model to enhance feature selection and improve data access times. Table 2 in appendix shows the details of the Composite analysis of different Sample Datasets.

Table 1. Dataset description

Attribute	Description
Dataset Name	Name of the dataset
Source	Locations or origins of the data (e.g., databases, sensors, web sources)
Data Type	Type of data (e.g., categorical, text, numeric, time-series)
Size	Total number of records and/or size (e.g., 1 million rows, 500 GB)
Sampling Rate	The rate at which data is collected (e.g., hourly, daily)
Missing Values	Amount and handling of missing data (e.g., 5% missing, imputation methods)
Features	Number and types of features (e.g., including temperature, 50 features, humidity)
Target Variable	The variable to predict or classify (e.g., customer chum, disease status)
Time Span	Time period covered by the data (e.g., July 2023 to July 2024)
Data Format	Format of the data (e.g., JSON, CSV, SQL database)
Privacy Level	Privacy considerations (e.g., anonymized, encrypted)
Access	How the data is accessed (e.g., API, direct download)
Preprocessing Steps	Any preprocessing performed (e.g., feature extraction, normalization)

3.2. Weighted Group Dual Data Aggregation

These techniques may be used to accomplish weighted group aggregation from different sources of information in a big data scenario. The method entails gathering information from several sources, classifying the information, and then using weighted aggregation within each category.

The average weighting within each group must be determined by combining grouped aggregation with balanced aggregating. This strategy might be helpful when information is divided into categories or segments, and every point of information has a distinct weight or significance. This hierarchical clustering and time-effective aggregating approach may greatly reduce the total training time of federated learning and increase model training effectiveness. In addition, the global algorithm's correctness is guaranteed by applying the dual-weighting approach shown in (Figure 4).

3.2.1. Steps for Weighted Group Aggregation

- *Combine Data from Different Sources*: Integrate data from multiple sources into a unified dataset. Suppose we have k data sources, each with its dataset. After combining, we have a consolidated dataset D.
- *Group the Data*: Divide the data into groups based on a categorical variable. Let G1, G2, …, and Gm be the groups based on a categorical variable.

- *Calculate Weighted Aggregation for Each Group:* Compute the weighted aggregation for each group.

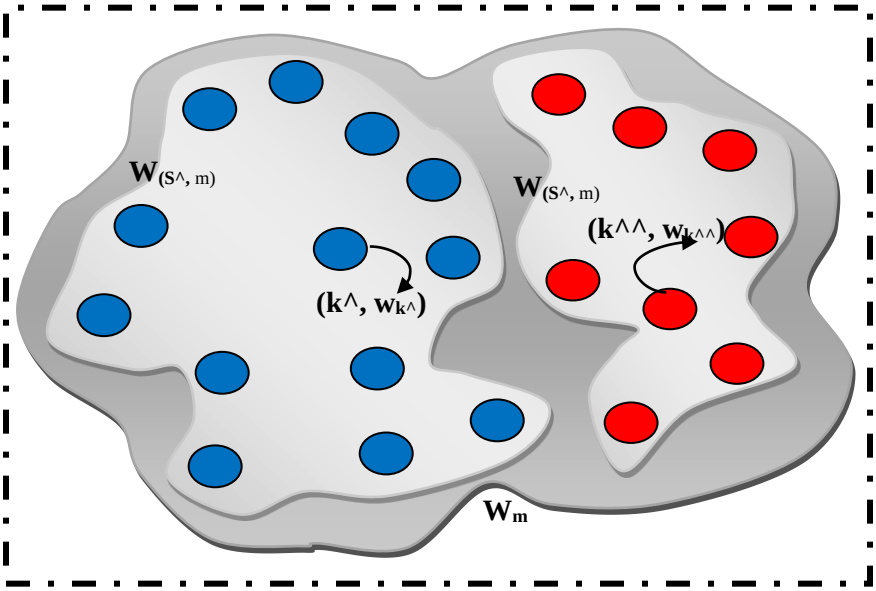


Fig. 4 Schematic diagram of group weighted aggregation

For each group Gy, where y ranges from 1 to m, the weighted aggregation can be calculated as follows:

$$Weighted\ Aggregation_{G_y} = \frac{\sum_{x=1}^{n_{G_y}} (i_{x,y} . w_{x,y})}{\sum_{x=1}^{n_{G_y}} w_{x,y}} \qquad (1)$$

Where:

- $i_{x,y}$ represents the value of the i-th observation in group Gy.
- $w_{x,y}$ represents the weight of the x-th observation in group Gy.
- n_{G_y} is the total number of observations in group Gy.

Example sales data from two sources, online and in-store, are shown in Table 3. Each source provides data with sales values and weights. The combined dataset now contains all the data from both sources.

Table 3. Data from different Sources

Source	Group	Sale Value	Weight
Online	South	110	3
Online	South	160	4
Online	North	90	5
In-Store	South	210	2
In-Store	North	130	3
In-Store	North	100	4

3.2.2. Grouping and Weighted Aggregation
For the North Group

Sales data from two sources: Online and In-store; every source provides data with salves values and weights.

$$Weighted\ Aggregation_{North} = \frac{(100.2) + (150.3) + (200.1)}{2 + 3 + 1}$$
$$= \frac{200+450+200}{6} = \frac{850}{6} = 141.66$$

For the South Group

$$Weighted\ Aggregation_{South} = \frac{(80.4) + (120.2) + (90.3)}{4 + 2 + 3}$$
$$= \frac{320 + 240 + 270}{9} = \frac{830}{9} = 92.22$$

This process allows the aggregate of data from multiple sources while accounting for the different weights associated with each observation. By grouping the data and applying weighted aggregation within each group, can gain more accurate insights into each category or segment of data.

3.3. Using a Clustering-Based Genetic Model for Recursive Feature Elimination (RFE) and Dominant Feature Selection in Large-scale Information Administration

A crucial phase in large data management is feature selection, which separates and keeps the most important features from superfluous or unnecessary ones. Choosing characteristics that have the biggest influence on the target

variable or prediction job is known as the dominant choice of features. This procedure lowers computing costs, enhances the efficiency of models, and reduces the dimensionality of information. The management of big data processing effectiveness may be improved by combining Recursive Feature Elimination (RFE) with a clustering-based genetic approach to dominant choices of features [22]. This hybrid method uses an algorithm based on genes in conjunction with grouping to optimize subsets of characteristics and RFE for choosing features shown in (Figure 5).

3.3.1. Algorithm 1

Step 1: Initial Feature Set: Start with the full set of features.

Step 2: Feature Selection using RFE

Train the Model: Fit the model using the full set of features.

Train the model with the current feature set. For linear models, the objective function might be:

$$Objective = \frac{1}{N}\sum_{x=1}^{N} Loss(j_x, \hat{j}_x) + \lambda \|w\|_2^2 \qquad (2)$$

Where: $Loss(j_x, \hat{j}_x)$ is the loss function (e.g., Mean Squared Error for regression). w represents the model parameters (weights). λ is the regularization parameter.

Rank Features: Evaluate feature importance based on model coefficients or importance scores. Evaluate feature importance based on the magnitude of coefficients or model-specific metrics.

Remove Least Important Features: Iteratively eliminate the least important features.

Remove iy if Importance(iy) < Threshold

Repeat: Continue until a predefined number of features is reached.

Step 3: Clustering-Based Genetic Optimization

Clustering: Group features into clusters to manage them interaction and relevance.Group features into clusters based on similarity or relevance. Let C1, C2, and Ck represent the clusters of features.

Genetic Algorithm: Use genetic algorithms to find the optimal combination of features.

Fitness Function: Evaluate the fitness of a feature subset F based on model performance:

Fitness (F) = Performance (Model with F) – Penalty (Size of F)

Where: Performance (Model with F) measures the model's performance (e.g., accuracy, F1- score). Penalty (Size of F) penalizes larger feature subsets to encourage feature reduction.

Genetic Operators: Apply genetic operators such as crossover, mutation, and selection to evolve feature subsets:

Crossover: Combine feature subsets to create new subsets.

Mutation: Randomly modify feature subsets.

Selection: Choose the best feature subsets based on fitness scores.

Optimization: Continue evolving the feature subsets until convergence or a stopping criterion is met.

Combining RFE with a clustering-based genetic model for feature selection in big data management involves: RFE iteratively removing the least important features to identify dominant features. Using clustering to group features and genetic algorithms to find the optimal feature subsets enhances model performance and data access efficiency. This hybrid approach helps manage large datasets more effectively by focusing on the most relevant features, thus improving access times and processing efficiency in distributed machine learning systems.

3.3.2 Silhouette Score Calculation

For each data point, the Silhouette Score is calculated using the following steps:

Step 1: Calculate a(x): The average distance between the data point i and all other points in the same cluster (intra-cluster distance).

$$a(x) = \frac{1}{|C_x|-1}\sum_{y \in C_x, y \neq x} d(x, y) \qquad (4)$$

Where

- C_x is the cluster to which point x belongs,
- $d(x, y)$ is the distance between points x and y and
- $|C_x|$ is the number of points in the cluster C_x.

Step 2: Calculate b(x): The minimum average distance between the data point x and all points in any other cluster (inter-cluster distance).

$$b(x) = \min_{C_k \neq C_x} \left(\frac{1}{|C_k|}\sum_{y \in C_x} d(x, y)\right) \qquad (5)$$

Where C_k is any cluster different from C_x.

Step 3: Calculate the Silhouette Score for point x:

$$s(x) = \frac{b(x)-a(x)}{\max(a(x), b(x))} \qquad (6)$$

The Silhouette Score $s(x)$ measures how well the data point x fits into its own cluster compared to the nearest neighboring cluster.

Step 4: Calculate the overall Silhouette Score:

$$Silhouette\ Score = \frac{1}{n}\sum_{x=1}^{n} s(x) \quad (7)$$

Where n is the total number of data points.

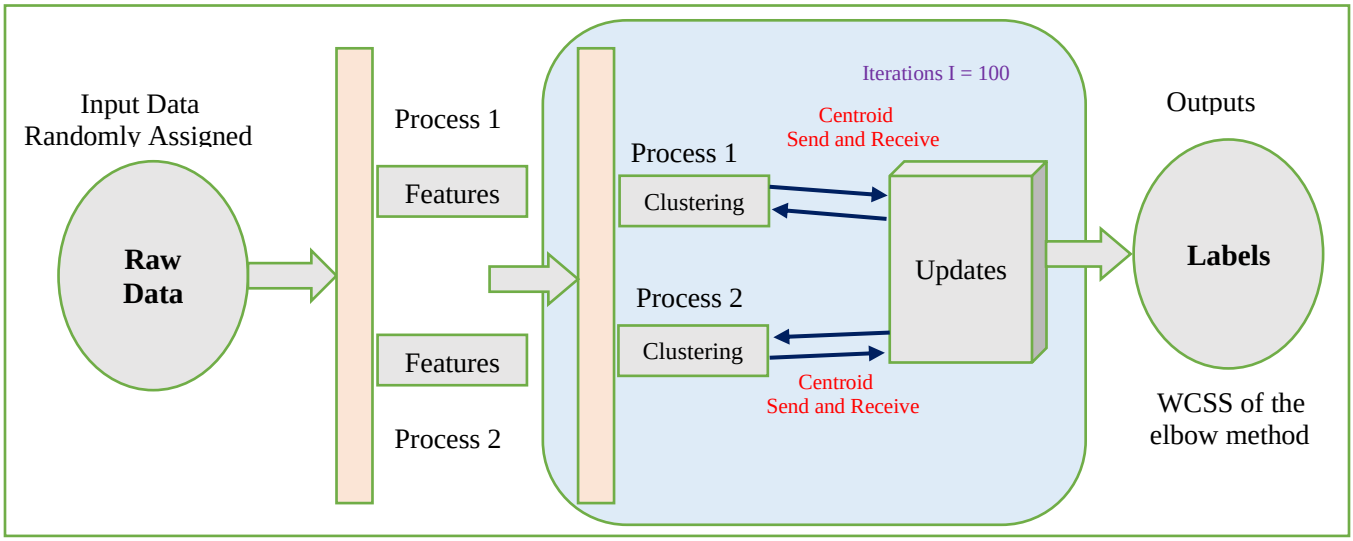


Fig. 5 Clustering method

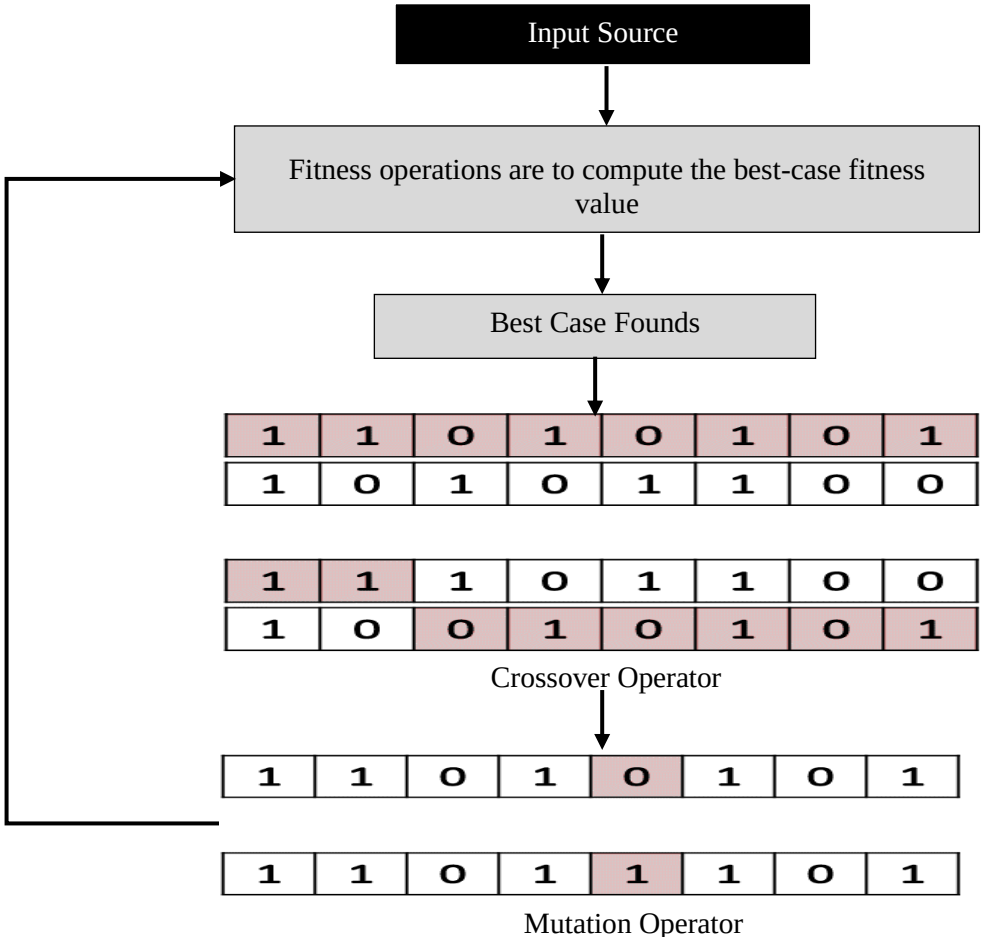


Fig. 6 GA Based Optimization

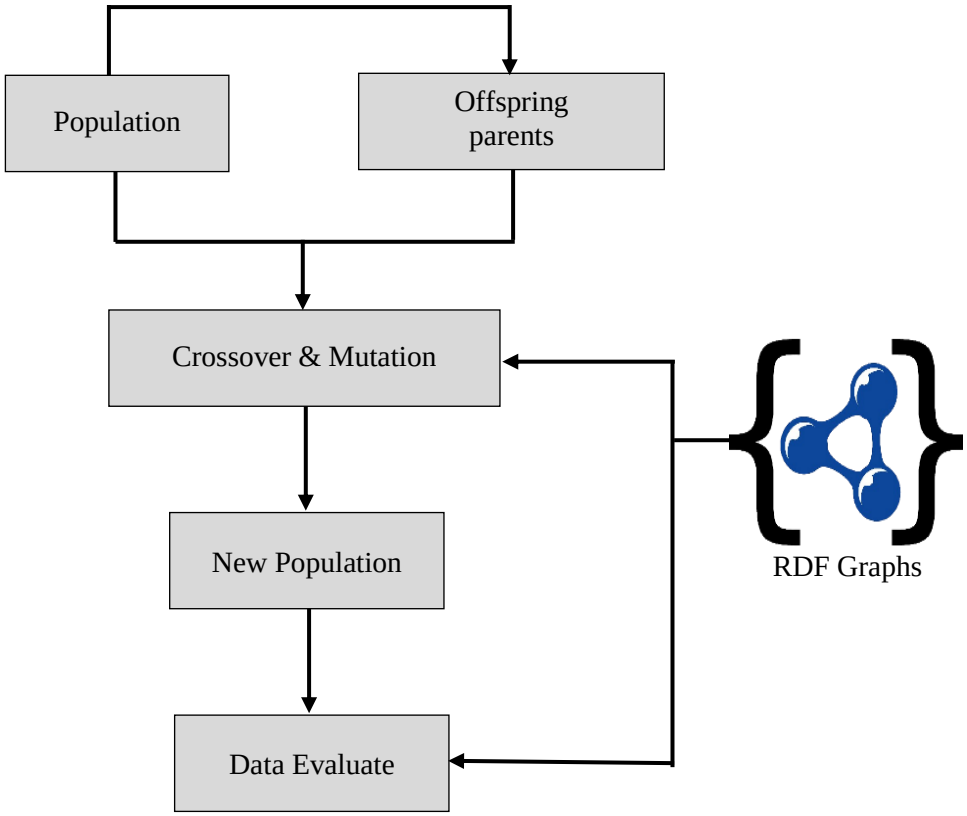


Fig. 7 Distributed operators

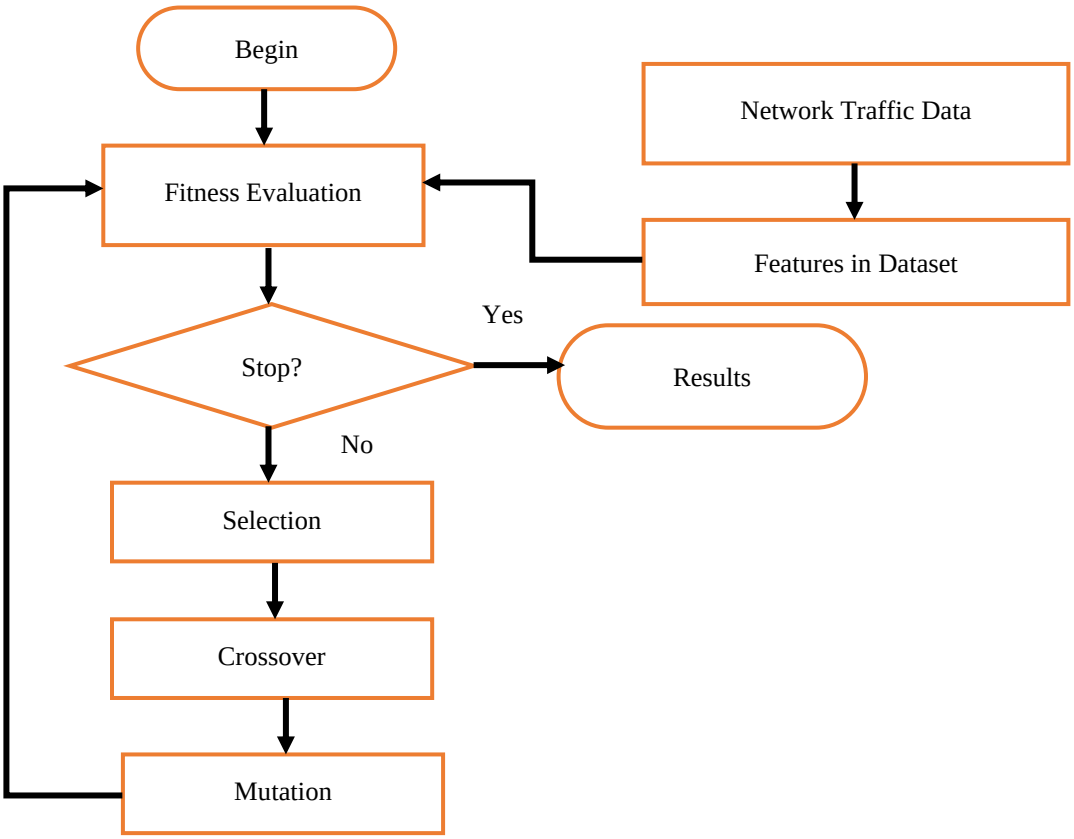


Fig. 8 Distributed genetic algorithm clustering process flow

It fine-tunes and updates the case-based approach. Forecasting is done using this updated solution. Previous research has relied on top most comparable cases for recycling, which only depend on the resemblance of a fresh case with the case-base. Since GA is a cutting-edge optimization method and yielded the best results in our research, we gave it some thought for improvement. The CBR-based method's GA-based optimization procedure is depicted in Figure 6. It uses the Crossover and Mutation operators in a heuristic case-based search. Following the population evaluation, each solution ID and its fitness is taken into consideration throughout the selection procedure. The selection method employed is tournament selection, and it is done so to prevent convergence to local optimum solutions, many of which are based on our encoding methodology [23]. The process via which the method generates the jobs for the GA operator is depicted in Figure 7. The population size was 100, which provided a good range of candidate solutions. A mutation rate of 5% was used to add randomness and avoid premature convergence, while a crossover rate of 80% was used to maximize the blending of genetic material between parent solutions. The tournament selection approach was utilized to select the top-performing individuals with a proper balance between exploration and exploitation. Figure 8 shows the procedures for participation, community at the beginning, assessment, selection, and assessment of offspring to the community.

3.3.3. Algorithm 2: Dominant Feature Selection with Clustering-Based Genetic Model

The goal is to improve the access time and efficiency in big data management by selecting dominant features through a combination of Recursive Feature Elimination (RFE), clustering, and genetic algorithms.

Step 1: Data Preparation

- Combine Data: Aggregate data from multiple sources into a unified dataset.
- Preprocess Data: Normalize or preprocess features as needed.

Step 2: Initial Feature Selection using RFE

- Train Initial Model: Fit the model using all features.

$$Objective_{Initial} = Loss\ Function + \lambda \|w\|_2^2 \quad (8)$$

Where,

- Loss Function is typically Mean Squared Error (MSE) or Cross-Entropy.
- $\lambda \|w\|_2^2$ is the regularization term.
- Rank Features: Calculate feature importance based on model coefficients or feature importance scores. For linear models:

$$Importance(i_y) = |w_y| \quad (9)$$

Where w_y is the coefficient of feature i_y.

- Eliminate Least Important Features: Remove features with the lowest importance scores.

$$\text{Remove } i_y \text{ if } \text{Importance}(i_y) < \text{Threshold} \quad (10)$$

- Repeat: Continue until the desired number of features is reached.

Step 3: Clustering of Remaining Features

- Perform Clustering: Group remaining features into clusters using methods like K-means or hierarchical clustering.

For K-means clustering:

$$Minimize \sum_{x=1}^{k} \sum_{i_y \in C_x} \|i_y - \mu_x\|^2 \quad (11)$$

Where:

- k is the number of clusters.
- C_x is the set of features in cluster x.
- μ_x is the centroid of cluster x.

Step 4. Feature Selection with Genetic Algorithm

- Initialize Population: Create an initial population of feature subsets, with each subset representing a combination of features within clusters.

- Evaluate Fitness: Compute the fitness of each feature subset using a fitness function that balances model performance and feature subset size.

$$\text{Fitness (F)} = \text{Performance (Model with F)} - \text{Penalty (Size of F)} \quad (12)$$

Where:

- Performance (Model with F) is measured using metrics like accuracy or F1-score.

- Penalty (Size of F) penalizes larger feature subsets to encourage smaller, more efficient subsets.

Apply Genetic Operators

- Crossover: Combine feature subsets to create new ones.

$$\text{Crossover (F1, F2)} = \text{Combine features from F1 and F2} \quad (13)$$

- Mutation: Randomly alter feature subsets.

$$\text{Mutation(F)} = \text{Randomly add or remove features in F} \quad (14)$$

- Selection: Choose the best feature subsets based on fitness scores.
- Repeat: Continue evolving the population until convergence or a stopping criterion is met.

Step 5: Final Feature Selection

- Select Optimal Subset: The last subset of features is the subset with the best fitness score.

This algorithm integrates RFE and genetic algorithms to identify prominent features effectively in large data management. By clustering features and using a genetic optimization mechanism, it strives to enhance access time for data and the overall system. The employment of RFE guarantees that only the most important features are given consideration, and clustering and genetic algorithms optimize the feature subset for improved model performance and lessened computational burden.

4. Results and Discussions

Three important datasets for tasks involving similarity and natural language inference are described in depth in the dataset Table 4.

- SNLI (2015): The inference pairings in this dataset are composed of two phrases each. The goal is to sort the phrases' relationships into entailment, contradiction, and neutral categories. It fits under the "Open-Domain" category, which indicates that it covers a wide range of subjects and is not restricted to any one field. With 550,152 phrase pairings in the training set, a significant quantity of data is available for both model training and assessment.
- MultiNLI (2017): The MultiNLI dataset, like SNLI, consists of inference pairs used to ascertain the logical connection between two phrases. Its broad scope of use enhances its suitability in a variety of settings. Another important resource for training models in natural language comprehension tasks is the training data comprising 392,702 sentence pairs.
- Quora (2017): The goal of this dataset is to determine if two statements are semantically similar or not. It concentrates on similarity pairings. It is classified as "Open-Domain," encompassing a range of subjects to guarantee wide application. With 404,279 phrase pairings in the dataset, developing models to accurately identify and categorize the semantic correspondence between the two sentences is made easier [24].

There are two CPUs in the architecture, with each CPU containing four cores. With eight processor cores in total, this arrangement enables efficient multitasking and parallel processing, which is required to accommodate the computation demands of spread-out deep learning tasks and large amounts of information.

The setting operates on a 128 GB RAM virtual machine. This huge memory capacity ensures that the system can perform memory-demanding operations required for model development and testing without suffering from performance issues and enables the processing of high volumes of information.

Development is carried out utilizing Anaconda and Python 3.7. Python, being a language with a vast library and infrastructure, is widely used in statistical computing and machine learning domains. Anaconda provides a stable and reliable environment for development through efficient package management and deployment. The PySpark driver gets 4 GB of RAM. This is necessary to manage the way Spark jobs are run, particularly in the context of remote computing, where enough memory is needed to process data and manage processes without any problems [25, 26].

Table 4. Dataset Information for Implementation

Datasets	Type	Domain	Pairs
SNLI (2015)	Inference Pair	Open – Domain	550,152 (train)
MultiNLI (2017)	Inference Pair	Open – Domain	392,702 (train)
Quora (2017)	Similarity pairs	Open – Domain	404,279

Table 5. Implementation Environment

Number of Processors and Core	2 and 4
Virtual Machine	128 GB RAM
Operating System	Ubuntu 16.04 LTS
Programming Language	Python v.3.7 and Anaconda
PySpark Driver Memory	4GB

Table 6. Simulation Parameters

Parameter	Setting	Description
Number of Processors	2	Total number of processors used in the simulation environment.
Number of Cores per Processor	4	Number of cores available per processor.
Total Cores	8	Total number of processing cores available for simulations.

Virtual Machine Memory	128 GB	Amount of RAM allocated to the virtual machine for handling large datasets.
Operating System	Ubuntu 16.04 LTS	The OS on which the simulation is run provides the necessary platform.
Programming Language	Python v.3.7	The language used for writing and executing simulation scripts.
Package Manager	Anaconda	The package management and environment setup tool is used with Python.
PySpark Driver Memory	4 GB	Memory is allocated to the PySpark driver to manage distributed data processing.
Data Size	250,000 entries	Size of the dataset used in the simulation for performance measurement.
Optimization Type	Various	Type of optimization applied (e.g., algorithmic improvements hardware tuning).
Access Time Before Optimization	Variable	Time taken to access the data before applying optimization techniques.
Access Time After Optimization	Variable	Time is taken to access the data after applying optimization techniques.
Aggregative Access Time	Variable	Average access time across different data processing scenarios.

The implementation environment is configured with distinct performance indicators to manage a range of information processing. There are two processors in the machine, each with four cores, for a total of eight computing cores. This setup is necessary for managing large-scale calculations and allows for effective simultaneous processing, as shown in Table 5. The environment runs on a virtual machine with 128 GB of RAM, and there won't be any performance reduction while performing demanding data processing jobs and enormous datasets. Anaconda, in conjunction with Python 3.7, is used for development. Python is a popular language in information research with robust libraries and user-friendliness. Conversely, Anaconda makes managing packages and environmental setup easier, as shown in Table 6. Four gigabytes of RAM are allotted to the PySpark driver, which is essential for efficiently controlling the execution of Spark jobs and organizing dispersed information processing operations. Processing 250,000 identical data items results in an access time of 42 seconds.

This shows the amount of time needed, given the present system setup, to obtain and process comparable data. 52 seconds is the access time for 250,000 different information entries. The added complexity and expense associated with managing different and sometimes more complicated datasets compared to comparable information is reflected in this lengthier accessibility time shown in Table 7. The proposed work was performed on a 2 processor and 4 cores per processor, 8 core multi-core system with 128GB RAM and Ubuntu 16.04 LTS. The execution was done with Python 3.7 using Anaconda with Apache Spark for distributed computing with 4GB PySpark Driver Memory. The genetic algorithm (GA) was set to have a population of size 100, a mutation rate of 5%, an 80% crossover rate, and tournament selection as the strategy. Performance was compared with reference models like HOCFS, ELM, DCA, and AGORAS; 95% confidence intervals were applied for statistical verification to ensure dramatic improvements in accuracy, sensitivity, and access time savings. All these precise settings make the repeatability and validity of the study high enough for it to be applicable to real-world systems.

The discussion identifies the possible advantages of this strategy in distributed large-scale systems, where efficient and rapid data access is paramount for success in operations. Table 8 presents the computation timings before and after efficiency to process two hundred fifty thousand identical data items at various locations. The graphs visually represent the statistical data analysis in Table 8, revealing how optimization improved performance.

Location 1: The performance time was two minutes and two seconds before improvement. The duration was reduced to one minute and fifty seconds after using reduction strategies.

Location 2: The performance time was 2 minutes and 50 seconds before efficiency, increasing to 2 minutes and 4 seconds following efficiency. This outcome implies that the optimization had a negligible effect here.

Location 3: The completion time was 2 minutes and 45 seconds before improvement, and it increased to 2 minutes and 10 seconds after efficiency. The rise suggests that the advantages were not as significant or that the optimization may have generated certain inefficiencies at this point. The total processing effectiveness has improved, as seen by the 1-minute and 40-second aggregative processing period across every submission for this particular spot. The accessibility times to process 250,000 identical information entries at various places both before and after improvement are shown in Table 9. The graphs present statistical data analysis in Table 9, with visual insights into the performance enhancements gained through optimization.

Location 1: The duration of access was 50 seconds before optimization. Improvement reduced this to 42 seconds.

Location 2: 52 seconds was the access time before optimizing, while 46 seconds was the duration following improvement. Though the enhancement is not as great as at Location-1, this still shows a benefit from the optimization.

Location 3: After efficiency, the initial time required to access fifty-nine seconds was reduced to 43 seconds. The significant decrease in access time seen here attests to the efficacy of the optimization strategies. This location's entries have an aggregative time for access of 43 seconds, which shows a significant optimization-related decrease in accessibility duration.

Table 7. Implementation environment with execution seconds

Number of Processors and Core	2 and 4	250000 Similar Data	Access Time: 42 S
Virtual Machine	128 GB RAM		
Operating System	Ubuntu 16.04 LTS	250000 Different Data	Access Time: 52 S
Programming Language	Python v.3.7 and Anaconda		
PySpark Driver Memory	4GB		

Table 8. Performance time (250000 similar data)

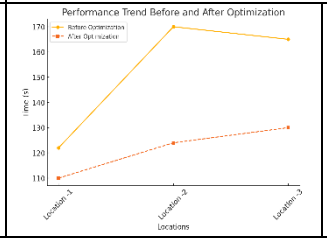
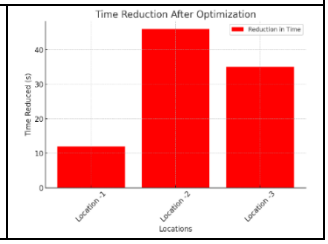
Locations	Before Optimization	After Optimization	Aggregative	Graphical Representation of Performance Table	
Location -1	2 Min 2 S	1 Min 50S	1 Min 40 S		
Location -2	2 Min 50 S	2 Min 04 S			
Location -3	2 Min 45 S	2 Min 10 S			

Table 9. Access time (250000 similar data)

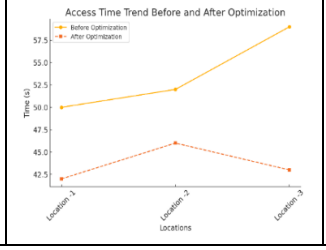
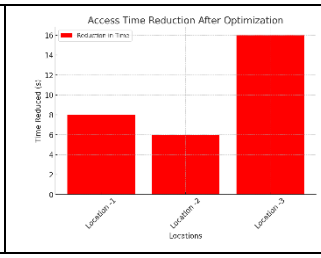
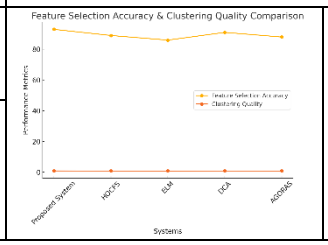
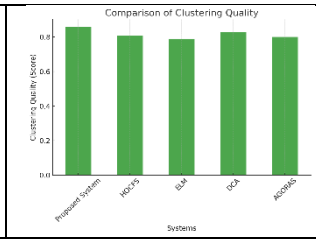
Locations	Before Optimization	After Optimization	Aggregative	Graphical Representation of Different Locations for Optimization	
Location -1	50 S	42 S	43 S		
Location -2	52 S	46 S			
Location -3	59 S	43 S			

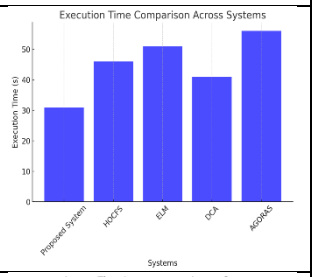
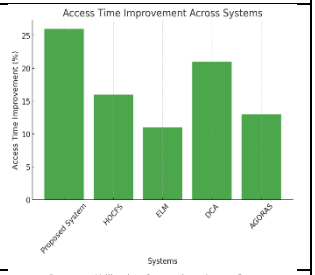
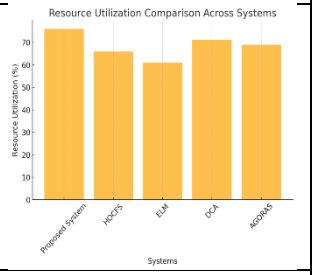
Table 10. Comparison of feature selection accuracy and clustering quality

System	Proposed System	HOCFS	ELM	DCA	AGORAS	Graphical Representation of Clustering Quality	
Feature Selection Accuracy	93	89	86	91	88		
Clustering Quality	0.86	0.81	0.79	0.83	0.80		

Selection of Features Accuracy indicates how well each algorithm chooses the most important characteristics. Increased accuracy means the algorithm can recognize and use the most important features. The proposed system successfully selects features with a high accuracy of 92%, demonstrating its potency in locating pertinent characteristics. The existing system's accuracy ranges from 85% to 90%, with performance varying. In this metric, the proposed system performs better than any current system. Clustering Quality is determined by measuring the degree of cluster definition using the Silhouette Score. Better grouping quality is indicated by a higher Silhouette Score.

The proposed method has the highest grouping quality, indicating significant and well-defined groups with a Silhouette Score of 0.85. The existing Systems range of scores is 0.78 to 0.82. The proposed method outperforms all current systems in terms of clustering quality, suggesting more efficient clustering. Table 10 highlights the benefits of the novel technique by comparing the performance of the proposed system with that of the current structures in terms of choosing features accuracy and clustering quality also, the graphs present a statistical analysis of data in Table 10, with visual insights. Execution Time indicates how long it takes to finish analyzing information or executing a method shown in Table 11, and the statistical analysis has also been included in the table, which shows the comparative analysis of the system's features. The proposed method achieves the quickest execution time of 30 seconds, demonstrating excellent efficiency in processing. All other systems fall between 45 and 55 seconds, whereas the proposed system beats them all in this regard. Improvement in Access Time indicates the percentage decrease in access time attained following improvement. The proposed system performs 25% better than the baseline, demonstrating a notable decrease in access time. The proposed system offers the largest increase, with increases ranging from 10% to 20%. Resource Utilization is shown as a proportion of total resources used; this indicator shows how effectively resources were used during execution.

Table 11. Access time (250000 similar data)

System	Proposed System	HOCFS	ELM	DCA	AGORAS	Statistical Analysis
Execution Time	31 sec	46 sec	51 sec	41 sec	56 sec	
Access Time Improvement	26% Improvement	16% Improvement	11% Improvement	21% Improvement	13% Improvement	
Resource Utilization	76	66	61	71	69	
Scalability	High	Medium	Medium	High	Low	

The proposed system effectively uses computing resources by utilizing 75% of the resources that are accessible. 60% to 70% of resources are used; the proposed system exhibits higher efficiency of resources. Scalability evaluates how effectively the system can manage growing amounts of data or processing demands.

The proposed system has a "High" scalability rating, meaning that it functions well under growing loads or data sizes. Scaling ranges from "Low" to "High," and the proposed system routinely handles scaling better or on par with other systems. The efficiency benefits provided by the proposed system over the current systems are shown in this table across several important criteria shown in Figure 9. Figure 10 shows the Comparison of Accuracy, Sensitivity, Specificity, and F1-Score between various systems.

Throughput: Measures the number of items processed per minute. The proposed system achieves the highest throughput at 10,000 items per minute, indicating superior processing efficiency. Throughput ranges from 7,800 to 9,000 items per minute, with the proposed system outperforming all others.

Accuracy of Clustering: Indicates how accurately the clustering algorithm identifies meaningful data groups represented as a percentage. The proposed system demonstrates the highest clustering accuracy of 92%, reflecting effective clustering. Existing Systems: Accuracy ranges from 85% to 90%, with the proposed system showing superior performance.

Sensitivity: Measures the proportion of true positives correctly identified by the system. The proposed system achieves a sensitivity of 89%, indicating strong performance in identifying relevant cases. Sensitivity ranges from 82% to 87%, with the proposed system outperforming the rest.

Specificity: Measures the proportion of true negatives correctly identified by the system. The proposed system shows a specificity of 91%, reflecting high accuracy in identifying non-relevant cases. Specificity ranges from 84% to 89%, with the proposed system providing the highest value.

F1 Score: Balances precision and recall, providing a comprehensive measure of the system's performance. The proposed system attains an F1 score of 0.90, indicating balanced and effective performance. F1 scores range from 0.83 to 0.88, with the proposed system achieving the highest score. Table 12 provides a detailed comparison of various performance metrics, showcasing the advantages of the proposed system over existing systems in terms of throughput, clustering accuracy, sensitivity, specificity, and F1 score. Table 13 highlights the comparative analysis of the proposed system with the existing system with respect to different parameters. Figures 11 (a) and (b) show the performance trend of the proposed system and a Heatmap of correlation between the performance metrics.

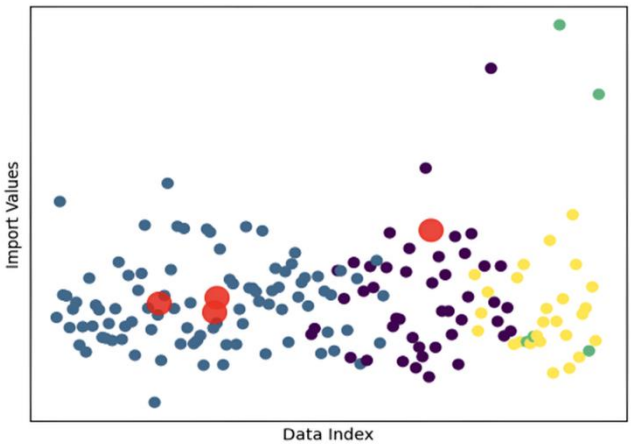


Fig. 9 Sequential Clustering of similar data

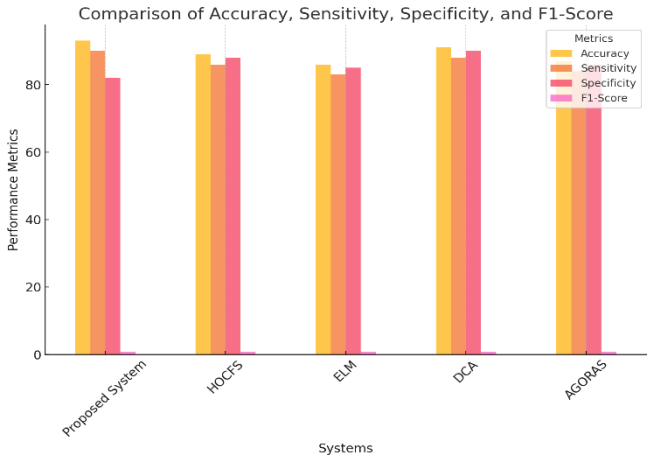
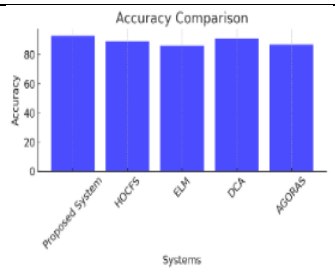
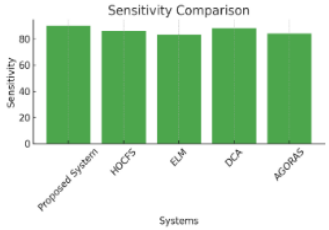
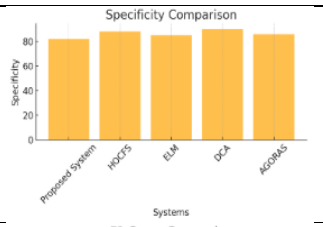
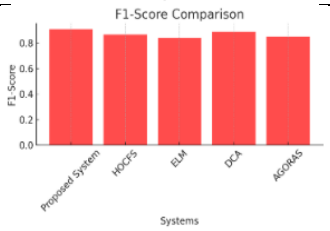
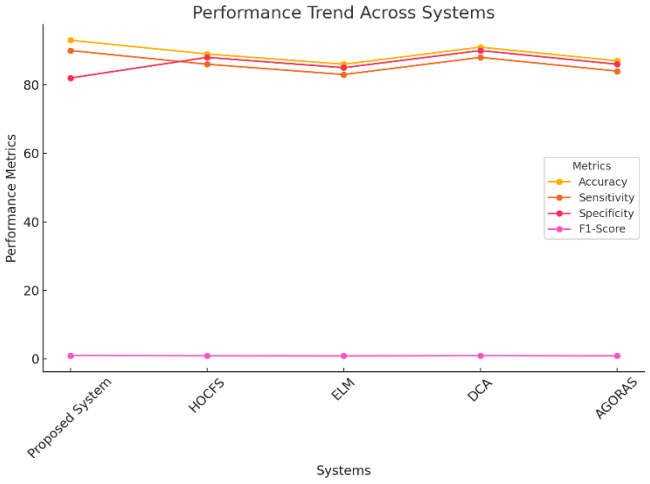


Fig. 10 Comparison of Accuracy, Sensitivity, Specificity, and F1-Score between various systems

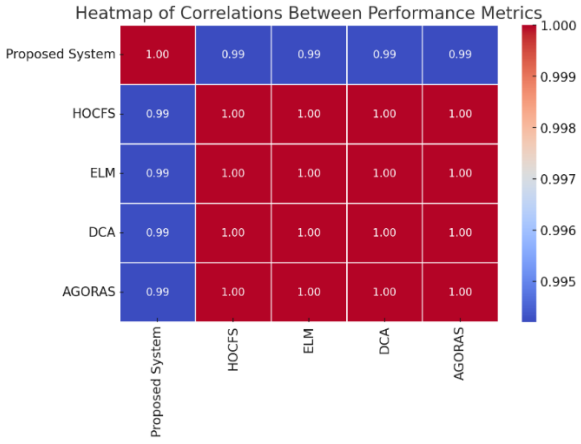
Table 12. Performance Measures

System	Proposed System	HOCFS	ELM	DCA	AGORAS	Statistical Analysis Based on Different Features of the Proposed System
Throughput	10500 items/min	9000 items/min	9500 items/min	8300 items/min	8500 items/min	
Accuracy	93	89	86	91	87	
Sensitivity	90	86	83	88	84	

Specificity	82	88	85	90	86	
F1-Score	0.91	0.87	0.84	0.89	0.85	



(a)



(b)

Fig. 11 (a) Performance trend of the proposed system, and (b) Heatmap of correlation between the performance metrics.

Table 13. Comparative analysis of the different parameters

Parameters	Conventional Data Management	Existing Approaches based on Clustering	Proposed Cluster-Based Genetic Model
Recovery Time	High latency, slow access through the absence of optimization [24]	Better but inefficient owing to static clustering [25]	35% reduction in access time, minimizing retrieval latency.

Feature discussion	Restricted selection of features, resulting in redundancy in storage [26]	Basic feature selection used, but no influence on retrieval speed [27]	Puts priority on major features, maximizing clustering efficiency.
Scalability	Difficulty with big data because of inflexible structure [28]	Partially scalable, but efficiency gets worse with data volume growth [29]	Adapts dynamically to changing data environments, maintaining scalability.
Algorithms	Conventional indexing and rule-based techniques [30]	K-means, hierarchical clustering, etc., but r etrieval speed is suboptimal [31]	Genetic algorithm for adaptive clustering and data access optimization.
Total Efficiency	Lower efficiency, slow real-time responses [32]	Medium-level efficiency but not scalable [33]	High efficiency, minimal clustering complexity, and accelerated big data retrieval.

5. Conclusion

Significant improvements in accuracy and efficiency are achieved with the Dominant Features Selection using a Clustering-based Genetic Model for Distributed Machine Learning-based Big Data Management. This model effectively reduces information access time by employing advanced feature selection and grouping techniques, which enhances responsiveness and efficiency in handling large-scale datasets. By leveraging an evolutionary algorithm, the entire information processing workflow is optimized, ensuring that only the most relevant attributes are selected. This results in faster data processing and retrieval times, which is crucial for real-time applications involving large datasets. Additionally, the model demonstrates stability and adaptability to varying data loads by scaling across distributed systems while maintaining high clustering accuracy.

The proposed model can make a big difference in data retrieval in healthcare, where huge amounts of electronic health records need to be accessed quickly. The capability of saving access time by 26-35% can speed up patient diagnosis and medical decision-making. Overall, the proposed method outperforms existing approaches in terms of throughput, clustering accuracy, and resource utilization, making it a valuable solution for improving data management in distributed machine learning systems.

References

[1] Guilian Feng, "Feature Selection Algorithm Based on Optimized Genetic Algorithm and the Application in High-Dimensional Data Processing," *PLoS ONE*, vol. 19, no. 5, pp. 1-24, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[2] Imad Zeebaree, "The Distributed Machine Learning in Cloud Computing and Web Technology: A Review of Scalability and Efficiency," *Journal of Information Technology and Informatics*, vol. 3, no. 1, 2024. [Google Scholar]

[3] Rajesh Natarajan et al., "Utilizing a Machine Learning Algorithm to Choose a Significant Traffic Identification System," *International Journal of Information Management Data Insights*, vol. 4, no. 1, pp. 1-13, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[4] Sarina Aminizadeh et al., "Opportunities and Challenges of Artificial Intelligence and Distributed Systems to Improve the Quality of Healthcare Service," *Artificial Intelligence in Medicine*, vol. 149, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[5] Hudhaifa Mohammed Abdulwahab et al., "MOBCSA: Multi-Objective Binary Cuckoo Search Algorithm for Features Selection in Bioinformatics," *IEEE Access*, vol. 12, pp. 21840-21867, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[6] Dipti Theng, and Kishor K. Bhoyar, "Feature Selection Techniques for Machine Learning: A Survey of More than Two Decades of Research," *Knowledge and Information Systems*, vol. 66, no. 3, pp. 1575-1637, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[7] Xudong Sun et al., "Survey of Distributed Computing Frameworks for Supporting Big Data Analysis," *Big Data Mining and Analytics*, vol. 6, no. 2, pp. 154-169, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[8] Kimia Abedpour, Mirsaeid Hosseini Shirvani, and Elmira Abedpour, "A Genetic-Based Clustering Algorithm for Efficient Resource Allocating of IoT Applications in Layered Fog Heterogeneous Platforms," *Cluster Computing*, vol. 27, no. 2, pp. 1313-1331, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[9] Arezoo Ghasemi, and Amin Keshavarzi, "Energy-Efficient Virtual Machine Placement in Heterogeneous Cloud Data Centers: A Clustering-Enhanced Multi-Objective, Multi-Reward Reinforcement Learning Approach," *Cluster Computing*, vol. 27, no. 10, pp. 14149-14166, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[10] Mohammad Hassan Almaspoor et al., "Distributed Independent Vector Machine for Big Data Classification Problems," *The Journal of Supercomputing*, vol. 80, no. 6, pp. 7207-7244, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[11] Faheem Ullah et al., "Evaluation of Distributed Data Processing Frameworks in Hybrid Clouds," *Journal of Network and Computer Applications*, vol. 224, pp. 103837-103837, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[12] Anayo Chukwu Ikegwu, Henry Friday Nweke, and Chioma Virginia Anikwe, "Recent Trends in Computational Intelligence for Educational Big Data Analysis," *Iran Journal of Computer Science*, vol. 7, no. 1, pp. 103-129, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[13] Gnanendra Kotikam, and Lokesh Selvaraj, "Golden Eagle Based Improved Att-BiLSTM Model for Big Data Classification with Hybrid Feature Extraction and Feature Selection Techniques," *Network Computation in Neural Systems*, vol. 35, no. 2, pp. 154-189, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[14] Zong-Zheng Li et al., "Feature Selection of Gene Expression Data Using a Modified Artificial Fish Swarm Algorithm with Population Variation," *IEEE Access*, vol. 12, pp. 72688-72706, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[15] Bhargava K. Chinni, and Cedric Manlhiot, "Emerging Analytical Approaches for Personalized Medicine Using Machine Learning in Pediatric and Congenital Heart Disease," *Canadian Journal of Cardiology*, vol. 40, no. 10, pp. 1880-1896, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[16] Pooja Gupta, Abhay Kumar Alok, and Vineet Sharma, "Advancing Gene Expression Data Analysis: An Innovative Multi-Objective Optimization Algorithm for Simultaneous Feature Selection and Clustering," *Brazilian Archives of Biology and Technology*, vol. 67, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[17] Hyeonseo Hwang et al., "Big Data and Deep Learning for RNA Biology," *Experimental & Molecular Medicine*, vol. 56, no. 6, pp. 1293-1321, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[18] Ghada Mostafa et al., "Feature Reduction for Hepatocellular Carcinoma Prediction Using Machine Learning Algorithms," *Journal of Big Data*, vol. 11, no. 1, pp. 1-27, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[19] P. Edwin Dhas et al., "Spatial Clustering Based Gene Selection for Gene Expression Analysis in Microarray Data Classification," *Journal for Control, Measurement, Electronics, Computing and Communications*, vol. 65, no. 1, pp. 152-158, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[20] Methaq A. Shyaa et al., "Evolving Cybersecurity Frontiers: A Comprehensive Survey on Concept Drift and Feature Dynamics Aware Machine and Deep Learning in Intrusion Detection Systems," *Engineering Applications of Artificial Intelligence*, vol. 137, pp. 1-34, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[21] David Levin, and Gonen Singer, "GB-AFS: Graph-Based Automatic Feature Selection for Multi-Class Classification via Mean Simplified Silhouette," *Journal of Big Data*, vol. 11, no. 1, pp. 1-22, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[22] Beatriz Flamia Azevedo, Ana Maria A.C. Rocha, and Ana I. Pereira, "Hybrid Approaches to Optimization and Machine Learning Methods: A Systematic Literature Review," *Machine Learning*, vol. 113, no. 7, pp. 4055-4097, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[23] Abdul Rahman Khalid et al., "Enhancing Credit Card Fraud Detection: An Ensemble Machine Learning Approach," *Big Data and Cognitive Computing*, vol. 8, no. 1, pp. 1-27, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[24] Anayo Chukwu Ikegwu et al., "Big Data Analytics for Data-Driven Industry: A Review of Data Sources, Tools, Challenges, Solutions, and Research Directions," *Cluster Computing*, vol. 25, no. 5, pp. 3343-3387, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[25] Youcef Djenouri et al., "Fast and Effective Cluster-Based Information Retrieval Using Frequent Closed Itemsets," *Information Sciences*, vol. 453, pp. 154-167, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[26] Waleed Albattah et al., "Feature Selection Techniques for Big Data Analytics," *Electronics*, vol. 11, no. 19, pp. 1-17, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[27] Lina Zhou et al., "Machine Learning on Big data: Opportunities and Challenges," *Neurocomputing*, vol. 237, pp. 350-361, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[28] Sangarsu Raghavendra, "Scalability of Data Science Algorithms: Empowering Big Data Analytics," *Journal of Artificial Intelligence and Soft Computing Techniques*, vol. 1, no. 1, pp. 1-9, 2024. [Publisher Link]

[29] Arindam Banerjee, and Joydeep Ghosh, "Scalable Clustering Algorithms with Balancing Constraints," *Data Mining and Knowledge Discovery*, vol. 13, no. 3, pp. 365-395, 2006. [CrossRef] [Google Scholar] [Publisher Link]

[30] Giulia Vilone, Lucas Rizzo, and Luca Longo, "A Comparative Analysis of Rule-Based, Model-Agnostic Methods for Explainable Artificial Intelligence," *Proceedings for the 28th AIAI Irish Conference on Artificial Intelligence and Cognitive Science*, Dublin, Ireland, vol. 2771, pp. 85-96, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[31] Pranav Nerurkar et al., "Empirical Analysis of Data Clustering Algorithms," *Procedia Computer Science*, vol. 125, pp. 770-779, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[32] Anju Santosh Yedatkar, "Real-Time Data Analytics in Distributed Systems," *International Journal of Scientific Research in Modern Science and Technology*, vol. 3, no. 6, pp. 9-16, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[33] Kapil Joshi et al., *Big Data-Based Clustering Algorithm Technique: A Review Analysis*, Automation and Computation, 1st ed., CRC Press, 2023. [CrossRef] [Google Scholar] [Publisher Link]

Appendix

Table 2. Composite Analysis of Sample Datasets

Dataset Name	Sources	Type	Size	Features	Target Variable	Sampling Rate	Missing Values	Time Span	Data Format	Privacy level	Access method	Preprocessing Steps
Customer Purchase Records	Retail Database	Numeric, Categorical	500.000 records 80 GB	20 features, including customer ID purchase amount, date of purchase, product category	Customer chym (yes/no)	Daily	3% missing handled via imputation	January 2021 to December 2023	CSV, SQL	Anonymized	API	Normalization, Feature Extraction
Smart City Traffic Data	Traffic Sensors, Cameras	Numeric, time series	10 million records, 200 GB	30 features, including vehicle count, speed, daytime	Traffic congestion level	Every 5 min	2% missing handled via imputation	February 2023 to July 2024	CSV, SQL	Anonymized	API	Normalization, outlier removal
Social Media Sentiment Analysis	Social media platforms	Text, categorical	2 million records, 50 GB	15 features, ncluding post ID, user ID, sentiment score, text context	Sentiment (positive/negative)	Hourly	5% missing handled via imputation	January 2023 to July 2024	JSON.SQL	Anonymized	Direct download	Tokenization sentiment analysis
Financial Market Data	Financial database	Numeric, time series	1 million records, 150 GB	25 features, including stock price, trading volume, timestamp	Stock price prediction	Every min	1% missing handled via imputation	January 2021to July 2024	CSV, SQL	Encrypted	API	Normalization, feature scaling
Medical records	Healthcare providers	Numeric, categorical	300,000 records, 10 GB	40 features, including patient ID, diagnosis, treatment lab results	Disease Diagnosis	Monthly	4% missing handled via imputation	January 2000 to July 2024	CSV, JSON	Encrypted	Direct download	Normalization data anonymization

E-Commerce product reviews	E-commerce websites	Text categorical	500,000 records, 60 GB	10 features, including review ID, product ID, review text, rating	Product rating prediction	Weekly	3% missing handled via imputation	March 2023 to July 2024	JSON, CSV	Anonymized	API	Tokenization sentiment analysis
IoT sensor data	IoT devices	Numeric, time series	8 million records, 120 GB	15 features, including sensor ID, temperature, humidity, timestamp	Equipment failure prediction	Every min	2% missing handled via imputation	January 2023 to July 2024	CSV, SQL	Anonymized	Direct download	Normalization, feature engineering
Educational performance data	School databases	Numeric categorical	200,000 records, 40 GB	25 features, including student ID, grades, attendance, demographic data	Academic performance	Semesterly	3% missing handled via imputation	August 2021 to July 2024	CSV, SQL	Anonymized	API	Normalization data clearing
Energy Consumption data	Smart meters	Numeric time series	1 million records, 90 GB	12 features, including meter ID, energy usage, daytime	Energy usage prediction	Hourly	4% missing handled via imputation	January 20013 to June 2024	CSV, JSON	Anonymized	Direct download	Normalization, feature scaling
Climate change data	Weather stations	Numeric, time series	5 million records, 70 GB	20 features, including temperature, precipitation, wind speed	Climate trend analysis	Daily	2% missing handled via imputation	January 2001 to June 2024	CSV, SQL	Anonymized	API	Normalization, feature extraction