International Journal of Advancement in Electronics and Computer Engineering (IJAECE)
Volume 13, Issue 11, November. 2024, pp. 108-114 ISSN 2278 -1412

Optimizing DBSCAN Density-Based Clustering Algorithm For Enhanced Performance in Data Mining

Uvaish Akhter[1], Mr. Jeetendra Singh Yadav[2]
[1]M. Tech., Scholar, uvaishakhter0@gmail.com, CSE Bhabha University, Bhopal, India
[2]Assis. Prof., jeetendra2201@gmail.com, RKDFCE, Bhopal, India

Abstract *– Data mining techniques play a crucial role in extracting valuable insights from large datasets, with clustering methods being among the most widely used. The Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm is notable for its ability to identify clusters of varying shapes while effectively handling noise. However, DBSCAN faces limitations with high-dimensional data and varying density clusters, which restrict its performance in complex datasets. This thesis investigates methods to enhance the performance of DBSCAN, focusing on optimizing parameters, improving computational efficiency, and addressing density variations within clusters. We propose an advanced DBSCAN framework that integrates adaptive parameter selection and novel density-based heuristics to improve accuracy and scalability in high-dimensional data mining applications. Experimental results demonstrate that the enhanced DBSCAN algorithm achieves superior clustering accuracy, reduced computational time, and improved noise resilience compared to the traditional DBSCAN. These findings highlight the enhanced DBSCAN's potential as a robust clustering solution for real-world data mining tasks, particularly in scenarios involving large, complex datasets.*

Keywords: Data Mining, Clustering Algorithms, DBSCAN (Density-Based Spatial Clustering of Applications with Noise), Performance Optimization, Density Variations, High-Dimensional Data, Noise Handling, Parameter Tuning, Adaptive Clustering

I. INTRODUCTION

Clustering is a fundamental technique in data mining that organizes data into groups, or clusters, based on similarities within the data. Among the various clustering algorithms, the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm is widely recognized for its ability to handle complex datasets, particularly those with varying densities and noise. Unlike traditional clustering methods such as k-means, DBSCAN does not require pre-defining the number of clusters, making it suitable for applications with unpredictable or unstructured data patterns.

Despite its popularity, DBSCAN has certain limitations, particularly in high-dimensional and large datasets. The algorithm's reliance on two critical parameters—epsilon (neighborhood radius) and minPts (minimum points per cluster)—is sensitive to data characteristics and can result in suboptimal clustering when not configured correctly. Furthermore, the high computational complexity of DBSCAN poses challenges for scalability, limiting its efficiency in handling extensive datasets in real-time applications. As the demand for efficient clustering techniques grows across diverse fields—such as anomaly detection, image segmentation, and spatial analysis—enhancing the performance and adaptability of DBSCAN is a pressing need.

This thesis focuses on the enhancement of DBSCAN's performance in the context of data mining, addressing key issues related to parameter sensitivity, computational efficiency, and noise resilience. By introducing optimized methods for parameter tuning and adaptive clustering, we aim to make DBSCAN more effective in handling diverse density variations and complex data structures. Our proposed improvements are validated through extensive testing on synthetic and real-world datasets, illustrating the effectiveness of the enhanced DBSCAN in terms of clustering accuracy, scalability, and computational speed.

II. UNDERSTANDING DATA MINING

Data mining is a sophisticated analytical process used to discover patterns, correlations, and insights from large volumes of data. This field, which intersects with

Figure 1 Categorization of data mining techniques.

DOI Member 236.24.428

statistics, machine learning, and database systems, involves extracting valuable information from datasets to support decision-making and enhance business intelligence. As organizations increasingly rely on data for strategic and operational decisions, the importance of data mining has grown significantly, transforming raw data into actionable knowledge.

III. METHOD

The proposed methodology for enhancing the performance of the DBSCAN clustering algorithm is structured into several key phases. Each phase focuses on addressing the limitations of the existing DBSCAN algorithm, ensuring that the enhancements are systematic and evidence-based. Advanced DBSCAN (Density-Based Spatial Clustering of Applications with Noise) clustering algorithm, including its key concepts, equations, and enhancements to improve its performance. DBSCAN is a density-based clustering algorithm that groups together points that are closely packed together (high density) while marking points that lie alone in low-density regions as outliers. The algorithm is defined by two key parameters:

- **Epsilon (ε)**: The maximum distance between two points for them to be considered as part of the same neighborhood.
- **MinPts**: The minimum number of points required to form a dense region.

Algorithm Steps

1. **Neighbor Identification**: For each point p in the dataset, identify all points within the distance ϵ\epsilonϵ using the equation:

$$N(p, \epsilon) = \{q \in D | \text{distance}(p, q) \leq \epsilon\}$$

where D is the dataset and distance(p,q) is a distance metric (commonly Euclidean distance).

2. **Core Points**: A point ppp is considered a core point if the number of points in its ε-neighborhood is at least MinPts:

$$\text{CorePoint}(p) = \begin{cases} \text{True} & \text{if } |N(p, \epsilon)| \geq \text{MinPts} \\ \text{False} & \text{otherwise} \end{cases}$$

3. **Cluster Formation**:
   - If p is a core point, create a new cluster and add p to this cluster.
   - For each point q in N(p,ϵ):
     - If q is not yet assigned to any cluster, assign it to the current cluster.
     - If q is a core point, expand the search and repeat the process.
4. **Noise Handling**:
   - Points that are neither core points nor directly reachable from core points are classified as noise.

Advanced DBSCAN Enhancements

To improve the performance of the basic DBSCAN algorithm, several enhancements can be implemented, leading to what we can call Advanced DBSCAN. These enhancements focus on dynamic parameter selection, improved distance calculations, and more efficient neighborhood searches.

Dynamic Parameter Selection

Instead of using static values for ε and MinPts, we can use adaptive methods to calculate these parameters based on the dataset characteristics.

- **Adaptive Epsilon**: Compute ε based on the k-nearest neighbors of each point. For each point p, the distance to the k^{th} nearest neighbor can be used to define ε:

$$\epsilon_p = \text{distance}(p, NN_k(p))$$

- **Adaptive MinPts**: Define MinPts as a function of the local density:

$$\text{MinPts}_p = \lceil \alpha \cdot \text{density}(p) \rceil$$

where α is a tunable parameter, and density can be calculated based on the number of points in the local neighborhood.

Improved Distance Calculation

Utilizing advanced distance metrics can enhance clustering accuracy, especially for high-dimensional data.

- **Cosine Similarity**: For points p and q:

$$\text{similarity}(p, q) = \frac{p \cdot q}{\|p\| \|q\|}$$

3. Efficient Neighborhood Search

Utilizing spatial indexing structures such as R-trees or KD-trees can significantly reduce the computational overhead associated with neighbor searches.

- **R-Tree**: A hierarchical data structure that allows for efficient spatial access methods, enabling rapid retrieval of points within the ε-neighborhood.
- **Parallel Processing**: Implement a parallelized version of the algorithm using multi-threading or distributed computing frameworks, allowing for simultaneous clustering of data partitions.

Procedure:

- **Initialization**: We start by initializing a `Cluster ID`, lists for `Clusters` and `Noise`, and a `Visited` array to keep track of which points have been processed.

- **Main Loop**: We iterate through each point in the dataset:

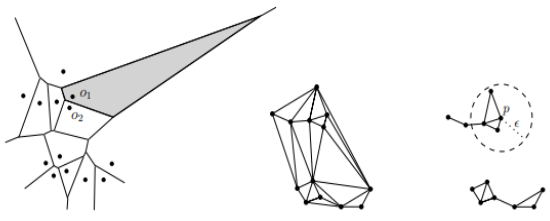
  - If the point has already been visited, we skip it.
  - We perform a region query to find all neighboring points within the defined epsilon radius.
  - If the number of neighbors is less than `minPts`, we classify the point as noise.

- **Cluster Formation**: If a point has enough neighbors:

  - We increment the cluster ID and start forming a new cluster.
  - We iterate through each neighbor, marking it as visited and expanding the neighborhood if it meets the criteria for density.

- **Cluster Filtering**: After identifying clusters, we can filter out any clusters that are smaller than `minPts`, which helps in removing noise or less significant clusters.

- **Region Query Function**: This helper function calculates the distance between the point and others in the dataset to find neighbors.



(a)Voronoi diagram (b)Delaunay graph (c) Remainder graph after edge removal

Figure 2 Step-2 Algorithm

Algorithm:

Advanced_DBSCAN
Input:
Dataset D (points in n-dimensional space)
ε (epsilon) - maximum radius of neighborhood
minPts (minimum points) - minimum number of points to form a dense region

Output:
Clusters (list of clusters), Noise (list of noise points)
1. Initialize:
- Cluster ID = 0
- Create an empty list for Clusters and Noise
- Create a boolean array Visited[] to keep track of visited points, initialized to False

2. For each point P in D:
a. If Visited[P] is True, continue to the next point
b. Mark Visited[P] as True
c. Neighbors = RegionQuery(P, ε) // Find neighbors within ε distance from P

d. If |Neighbors| < minPts:
- Add P to Noise // Point P is considered noise
continue
e. Cluster ID += 1
f. Create a new cluster C with Cluster ID
g. Add P to C

h. For each point Q in Neighbors:
i. If Visited[Q] is False:
- Mark Visited[Q] as True
- Neighbors_Q = RegionQuery(Q, ε) // Find neighbors of Q

ii. If |Neighbors_Q| >= minPts:
- Add Neighbors_Q to Neighbors // Expand the neighborhood

iii. If Q is not in any cluster:
- Add Q to C // Add Q to the current cluster

3. For each cluster C:
a. If |C| < minPts:
- Remove C from Clusters // Remove small clusters that do not meet criteria

4. Return Clusters, Noise

Function RegionQuery(P, ε):
- Initialize empty list of neighbors
- For each point O in D:

```
if Distance(P, O) <= ε:
    Add O to neighbors
return neighbors
```

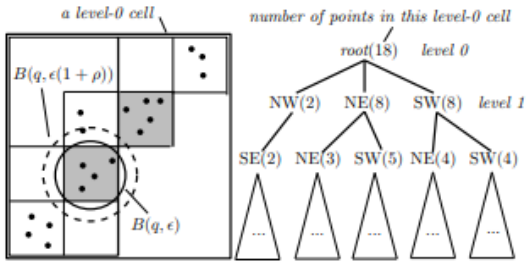


Figure 3 Range Counting

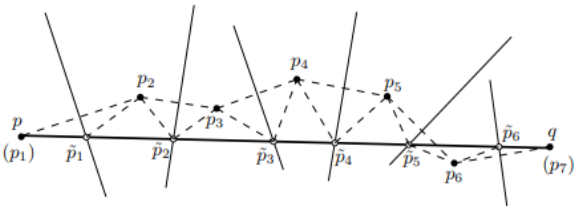
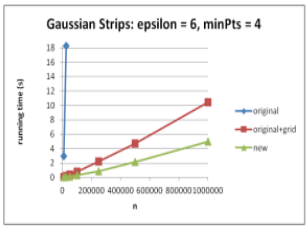


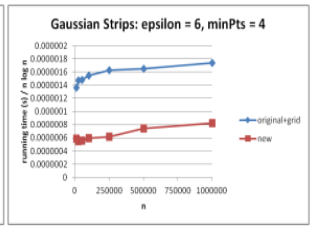
Figure 4 Correctness proof of our Step-2 algorithm

IV. RESULT

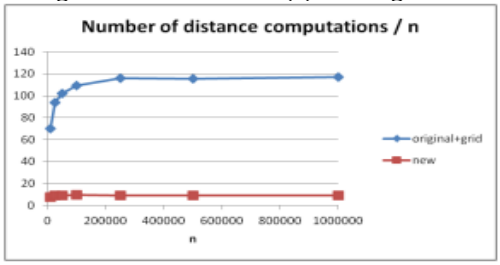
The performance of the algorithm is evaluated based on a series of metrics that reflect its efficacy in identifying clusters within the given dataset.



(a) running time



(b) running time / n log n

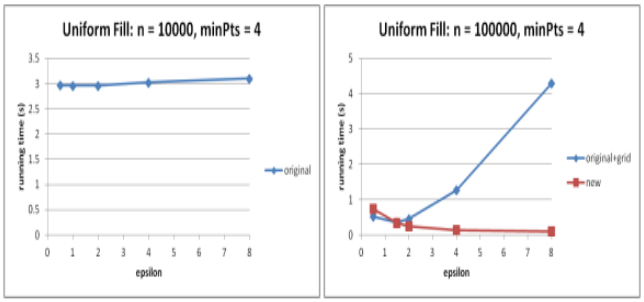


(c) number of distance computations / n

Figure 5: Gaussian strips, variable n

The results are compared with those from the traditional DBSCAN to highlight the enhancements achieved through the proposed modifications. For the strip datasets, which include both uniform and Gaussian variations, we determined the parameters ϵ and minPts to ensure the algorithm accurately identifies each strip as a distinct cluster while recognizing noise as outliers. Next, we aim to evaluate the performance of the algorithms by varying the value of

ϵ. The results of this experiment using the uniform fill dataset are presented in Figure.

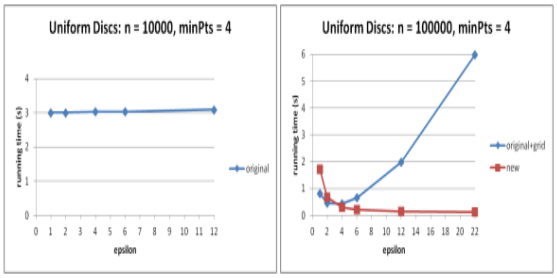


(a) original (b) improved

Figure 6: Uniform fill, variable ε

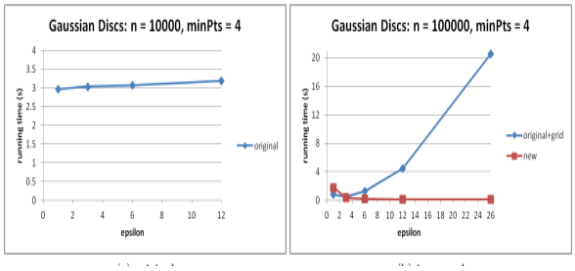
As illustrated in Figure 6 a, the performance of the original DBSCAN algorithm remains largely unaffected by changes in the value of ϵ\epsilonϵ. This is due to the fact that the original algorithm conducts a complete scan of the entire dataset for each point during the range query, regardless of the ϵ value.

In contrast, Figure 6b reveals more intriguing results, demonstrating that both the new algorithm and the original algorithm with grid are highly sensitive to variations in ϵ. Typically, the original grid-based algorithm benefits from a smaller ϵ value, as a larger ϵ\epsilonϵ leads to an increased number of points within each grid cell, thereby making the range query more computationally intensive. Conversely, the new algorithm benefits from a larger ϵ, as having more points within a grid cell increases the likelihood of locating a grid cell containing more than the minimum number of points, minPts. The running times of both algorithms converge when ϵ\epsilonϵ is around 1.5. However, at this value, the algorithm tends to identify multiple small clusters, which is not the intended outcome. When ϵ is set to 2 or higher, we achieve the desired result of recognizing all points as a single cluster. At this point, the new algorithm demonstrates superior performance compared to the original grid-based version. For the other data sets figure 7, 8,9 and 10.



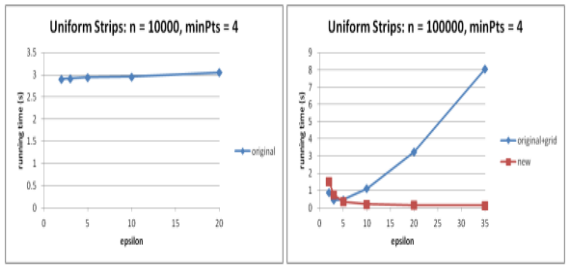
(a) original (b) improved

Figure 7: Uniform discs, variable ε



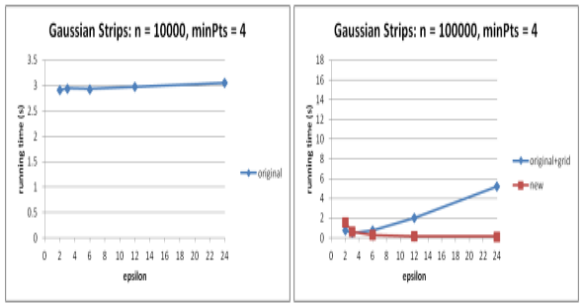
(a) original (b) improved

Figure 8: Gaussian discs, variable ε



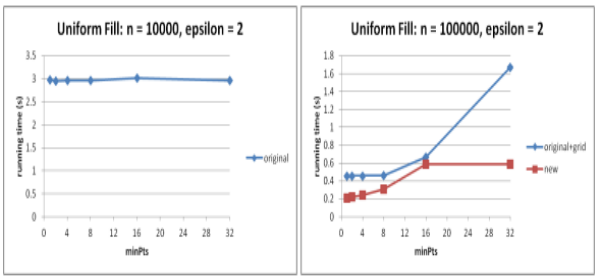
(a) original (b) improved

Figure 9: Uniform strips, variable ε



(a) original (b) improved

Figure 10: Gaussian strips, variable ε

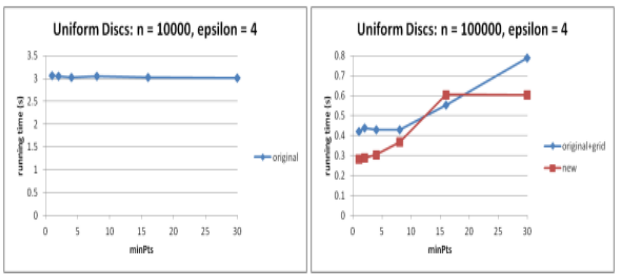


(a) original (b) improved

Figure 5.11: Uniform fill, variable minP ts

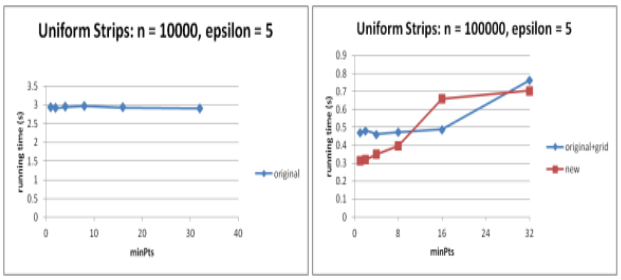
In the following experiment, we aimed to evaluate the algorithms' performance with varying values of minPts. The results for the uniform fill dataset are presented in Figure 12. Consistent with the observations in Figure 13a, Figure 13a indicates that the original DBSCAN algorithm shows minimal sensitivity to changes in minPts.

Figure 13b illustrates that the grid-based version of the original algorithm remains fairly stable for minPts≤8. However, when 8<minPts ≤32, the algorithm experiences a slowdown as the minPts value increases. While we cannot pinpoint the exact cause of this behavior, we note that for minPts≤8, the algorithm tends to yield a single large cluster. In contrast, when 8<minPts ≤32, the number of resultant clusters increases while their sizes decrease. The new algorithm also experiences increased processing time with higher values of minPts, although it remains stable once minPts reaches 16. One of the advantages of this new approach is its ability to bypass certain distance computations when the number of points within a cell exceeds minPts. As minPts increases, the number of such cells diminishes until it eventually reaches zero. Additionally, it is important to note that when examining points in neighboring cells, the new algorithm terminates its search once it has identified at least minPts points within a distance of ϵ\epsilonϵ. Therefore, if minPts is set too high, the algorithm's responsiveness to changes in minPts may diminish.



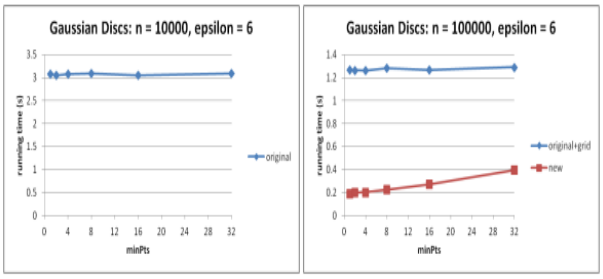
(a) original (b) improved

Figure 12: Uniform discs, variable minPts



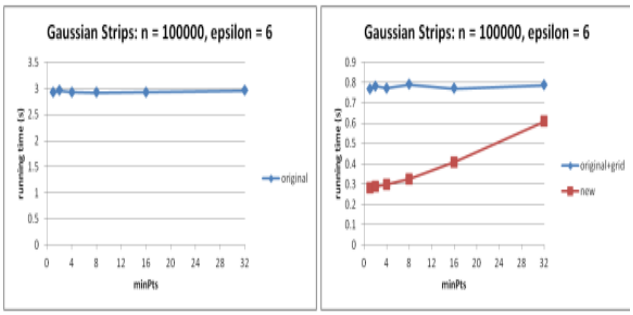
(a) original (b) improved

Figure 13: Uniform strips, variable minPts



(a) original (b) improved

Figure 14: Gaussian discs, variable minP ts



(a) original (b) improved

Figure 15: Gaussian strips, variable minP ts

V. Conclusion

This paper compared the performance of the newly developed algorithm against the "original DBSCAN with grid" approach, focusing specifically on their running times. The findings indicate that the new algorithm consistently outperforms the original grid-based version across various test scenarios. Notably, as the value of ϵ\epsilonϵ increases, the efficiency of the new algorithm improves, suggesting that starting with a larger ϵ\epsilonϵ may enhance clustering performance. It is advisable to first explore higher values of ϵ\epsilonϵ and then gradually reduce it to identify the optimal clustering outcome.

Additionally, our experiments revealed that lower values of minPts correspond to faster execution times for the new algorithm. Consequently, an effective strategy for clustering would be to initiate the process with a smaller minPts value and progressively increase it until a satisfactory clustering result is achieved.

References

[1] Yang, Chen Qian, Haomiao Li, Yuchao Gao, Jinran Wu, Chan-Juan Liu & Shangrui Zhao, “An efficient DBSCAN optimized by arithmetic optimization algorithm with opposition-based learning”, Volume 78, pages 19566–19604, (2022), Springer.

[2] Yuxian Duan; Changyun Liu; Song Li, “Battlefield Target Grouping by a Hybridization of an Improved Whale Optimization Algorithm and Affinity Propagation”, IEEE Access (Volume: 9), 2021, DOI: https://doi.org/10.1109/ACCESS.2021.3067729.

[3] Stephen Akatore Atimbire, Justice Kwame Appati & Ebenezer Owusu, “Empirical exploration of whale optimisation algorithm for heart disease prediction”, Scientific Reports volume 14, Article number: 4530 (2024).

[4] Shaoyuan Weng, Zimeng Liu, Zongwen Fan & Guoliang Zhang, “A whale optimization algorithm-based ensemble model for power consumption prediction”, 2024, Springer.

[5] Rami Sihwail, Mariam Al Ghamri, Dyala Ibrahim, “An Enhanced Model of Whale Optimization Algorithm and K-nearest Neighbors for Malware Detection”, Vol.17, No.3, 2024, International Journal of Intelligent Engineering and Systems, DOI: 10.22266/ijies2024.0630.47.

[6] K M Archana Patel & Prateek Thakral, “The best clustering algorithms in data mining”, ISBN:978-1-5090-0396-9, 2016, IEEE, DOI: 10.1109/ICCSP.2016.7754534.

[7] Manish Verma, Mauly Srivastava, Neha Chack, Atul Kumar Diswar, Nidhi Gupta, “A Comparative Study of Various Clustering Algorithms in Data Mining”, ISSN: 2248-9622, Vol. 2, Issue 3, 2012, IJERA.

[8] Ashish Dutt, Saeed Aghabozrgi, Maizatul Akmal Binti Ismail, and Hamidreza Mahroeian, “Clustering Algorithms Applied in Educational Data Mining”, Vol. 5, No. 2, March 2015, International Journal of Information and Electronics Engineering, DOI: 10.7763/IJIEE.2015.V5.513.

[9] G. Biswas; J.B. Weinberg; D.H. Fisher, “ITERATE: a conceptual clustering algorithm for data mining”, Volume: 28, Issue: 2, 2002, IEEE, DOI: 10.1109/5326.669556.

[10] Grabmeier and JRudolph A(2019)Techniques of Cluster Algorithms in Data MiningData Mining and Knowledge Discovery10.1023/A:10163084046276:4 (303-360)Online publication date: 1-Jun-2019. DOI: https://dl.acm.org/doi/10.1023/A%3A1016308404627

[11] Syed Thouheed Ahmed, S. Sreedhar Kumar, B. Anusha, P. Bhumika, M. Gunashree & B. Ishwarya,

"A Generalized Study on Data Mining and Clustering Algorithms", pp 1121–1129, Springer Link.

[12] Dingsheng Deng, "DBSCAN Clustering Algorithm Based on Density", ISBN:978-1-7281-9628-2, 2021, IEEE, DOI: 10.1109/IFEEA51475.2020.00199.

[13] Albasheer Fawzia Omer, H. Ahmed Mohammed, M. Ahmed Awadallah, Zia Khan, Said Ul Abrar & Mian Dawood Shah, "Big Data Mining Using K-Means and DBSCAN Clustering Techniques", SBD,volume 111, pp 231–246, 02 September 2022.

[14] Kawtar Sabor, Damien Jougnot, Roger Guerin, Barthélémy Steck, Jean-Marie Henault, Louis Apffel, Denis Vautrin, "A data mining approach for improved interpretation of ERT inverted sections using the DBSCAN clustering algorithm", Geophysical Journal International, Volume 225, Issue 2, May 2021, Pages 1304–1318, DOI: https://doi.org/10.1093/gji/ggab023.

[15] Fang Huang, Qiang Zhu, Ji Zhou, Jian Tao, Xiaocheng Zhou, Du Jin, Xicheng Tan and Lizhe Wang, "Research on the Parallelization of the DBSCAN Clustering Algorithm for Spatial Data Mining Based on the Spark Platform", Volume 9, Issue 12, 2017, 9(12), 1301, MPDI, DOI: https://doi.org/10.3390/rs9121301.

[16] Cheng-Fa Tsai; Chun-Yi Sung, "DBSCALE: An efficient density-based clustering algorithm for data mining in large databases", ISBN:978-1-4244-7969-6, 2010, IEEE, DOI: 10.1109/PACCS.2010.5627040.

[17] Suresh kurumalla, P srinivasa rao, "K-Nearest Neighbor Based Dbscan Clustering Algorithm For Image Segmentation", 31st October 2016. Vol.92. No.2, ISSN: 1992-8645, Journal of Theoretical and Applied Information Technology.