The Eurasia Proceedings of Science, Technology, Engineering & Mathematics (EPSTEM)
ISSN: 2602-3199

The Eurasia Proceedings of Science, Technology, Engineering and Mathematics (EPSTEM), 2025

Volume 36, Pages 220-228

ICBAST 2025: International Conference on Basic Sciences and Technology

Development of Big Data Clustering with Apache Spark

Bakhshali Bakhtiyarov
Azerbaijan State Oil and Industry University

Aynur Jabiyeva
Azerbaijan State Oil and Industry University

Gunay Hasanova
Azerbaijan State Oil and Industry University

Abstract: The research focuses on clustering technique development and big data in-formation retrieval has increased significantly during recent years. This paper introduces a new approach for distributed clustering which performs adap-tive density estimation. Packaging tests check the performance of method implemented using Apache Spark across various well-known datasets. The initial stage of this algorithm performs data partitioning by utilizing Bayesian LSH as one of its LSH proxies. The partition method decreases superfluous calculations while functioning as a parallel system with straightforward processes. The proposed algorithm demonstrates autonomous operation between its steps because the processing sequence does not introduce bottlenecks in the workflow. The stability of this proposed method increases together with outlier removal because the local structures maintain their integrity. The ordered weighted average (OWA) distance defines density through which clus-ters become more like their internal elements. A computer program evaluates local density peaks through node density computations. The selected peaks produce the cluster center value which determines how remaining points get assigned to proximal group. An assessment of the proposed method and previous research findings in present literature took place. The current method demonstrates greater accuracy together with reduced noise sensitivity through its calculated validity index findings. This method provides many advantages in terms of scalability together with high efficiency at reduced computational complexity. The strategy works for general clustering purposes and researchers have used it successfully in clustering and other problems.

Keywords: Scalable clustering, Apache spark, Resilient distributed dataset, MLLib, MapReduce, Big data

Introduction

The term Spark defines a diverse toolset which continues to boost data speed processing along with distributed systems capabilities (Tekdogan et al., 2021). The big data research community now focuses on Spark as a primary system in comparison to other frameworks particularly Hadoop MapReduce (Sahith et al., 2023). MLLib represents one of the algorithms developed on Spark whereas state-of-the-art clustering implementa-tions remain scarce on this platform (Gupta & Kumari, 2020). The clustering algorithm developed using Spark shows potential to operate properly in comparable distributed platform systems (Daghistani et al., 2020). The RDD requires complete utilization of its characteristics which include processing and less disk I/O together with Direct Acyclic Graph computing model and sophisticated local coaching and swift file operations and fault tolerance (Xiao & Hu, 2020).

As will be discussed in the following sections, big data scenarios are often char-acterized by the fact that the dataset size cannot be stored in the memory of a device. This makes it necessary to address the inherently distributing nature of the data, and many more the traditional clustering methods fell short on this re-quirement as

these techniques repudiate the distribute format (Ikotun et al., 2023). Normally, in a problem, their arrangement or even special cases such as the presence of outliers in these clusters will not be specified beforehand. Hence, to achieve effective clustering algorithms, it should be possible to perform clustering without the data distribution knowledge or properties of the clusters (Deng, 2020).

Clustering based on density provides certain benefits, which cannot be reached using other clustering algorithms. For instance, it can generate groups of an irregular shape unrelated to data topology, are least sensitive to noise, can be initiated at any point, and yield the exact same clustering each time the process goes through. In addition, it does not entail any assumptions that may characterise the output (Fahim, 2023). A new density-based clustering meaning called "Clustering by search and find of Density Peak" has been established in science using its effective structure (Zhang et al., 2022). The implementation of CDP began after its development and now applies to various subjects such as molecular, data analysis, remote sensor and complate vision (Wang et al., 2024). The major disadvantage of CDP comes from its high demand for computations while its implementation faces compatibility challenges with contemporary cloud computing systems. The present work presents a new distributed CDP approach to solve the limitations of CDP and Spark (Hou et al., 2020).

The design of the system enhances the procedures of the CDP algorithm, decreasing the communication overhead of linking compute nodes and lessening the computation burden (Wang et al., 2024). DCDPS independently shard the data using Bayesian Local Precision Sharding (BALSH) according to the similarity in computer nodes. By so doing, BALSH makes sure that similar data stays in the same node, thus avoiding extra computation and keeping data local (Işık, 2024). DCDPS is characterized by high scalability and relatively small usage of computational resources. The user can also set the value of the parameters to be used during clustering, including members in a cluster, distance and density thresh-olds; or the system can make the selection automatically. The density threshold is estimated by an adaptive method (Ezugwu et al., 2022). The performance and accuracy of DCDPS are tested using global validity indices for clusters and compared to modern meth-ods. As depicted by the results section, we find that DCDPS is comparable to both the basic CDP and other improved algorithms in terms of accuracy while having superior computational performance and scalability (Badri, 2019).

Method

Our approach in this study is to use and adapt the Clustering by Density Peaks (CDP) algorithm to analyze large amounts of sensor telemetry data across multiple computers. CDP uses two main ideas: checking how densely points are packed in a region and measuring distance from other high-density regions, to find the clusters. Hence, it can handle streaming measurements such as pump sensor logs, where the original data is generally complex and irregular and its cluster structure is unknown.

For this case, the dataset is defined as X = {x_{1}, x_{2}, ..., x_{n}} with the value x_i being a single real-time sensor combination with many features included.

x_i is equivalent to $(x_{i1}, x_{i2}, ..., x_id)^t$ and can be found in R^d

d represents the total number of attributes included in any measuring of temperature, vibration, pressure and flow rate.

At the start of the algorithm, each point's neighborhood density is calculated using a cutoff distance d_c.

$A(x_i)$ = set of all points x_j in X such that $dist(x_i, x_j) < d\_c$; ρ_i = size of the set $A(x_i)$

After that, for every point, δ_i is the minimum distance to any point where the density is greater.

δ_i stands for $\min\{ dist(x_i, x_j) \mid \rho_j > \rho_i \}$

The cluster centers have righthand ρ and righthand δ values and are plotted in the upper-right section of the (ρ, δ) graph. All other points are set to the same cluster as its nearest denser neighbor. We identify points with high δ and low ρ as outlying values. An illustration of this concept is shown in Figure 1, where the regression line separates cluster centers from outliers based on density and separation.

Due to calculating all the pairwise distances, the standard CDP implementation is at $O(n^2)$. To resolve this, we use two different approaches to improve efficiency:

1. Demonstrated symmetry in distance: dist(x_i, x_j) = dist(x_j, x_i)
2. Put the points in order from highest to lowest ρ, at which point you use higher-density points in the calculations for δ_i.

The system we use relies on Apache Spark 3.5 and Delta Lake. Streaming pump sensor data is sent to Spark Structured Streaming which writes it to Delta tables. Every small batch is treated independently using data in RDDs, where the local values of ρ and δ are calculated for each partition.

In the kernel-separation space ($\hat{f}$, δ), we give the following definitions:

P_1 is the combination of minimum $\hat{f}$ and maximum δ, P_2 is the pair of maximum $\hat{f}$ with minimum δ
P_1 and P_2 are linked by a line called Q. There are density peaks on the graph located to the right of Q.

We call this new way of working RT-DCDPS. What makes this technique useful for Industry 4.0 is that it is scalable, has fault tolerance and is resilient to concept drift.

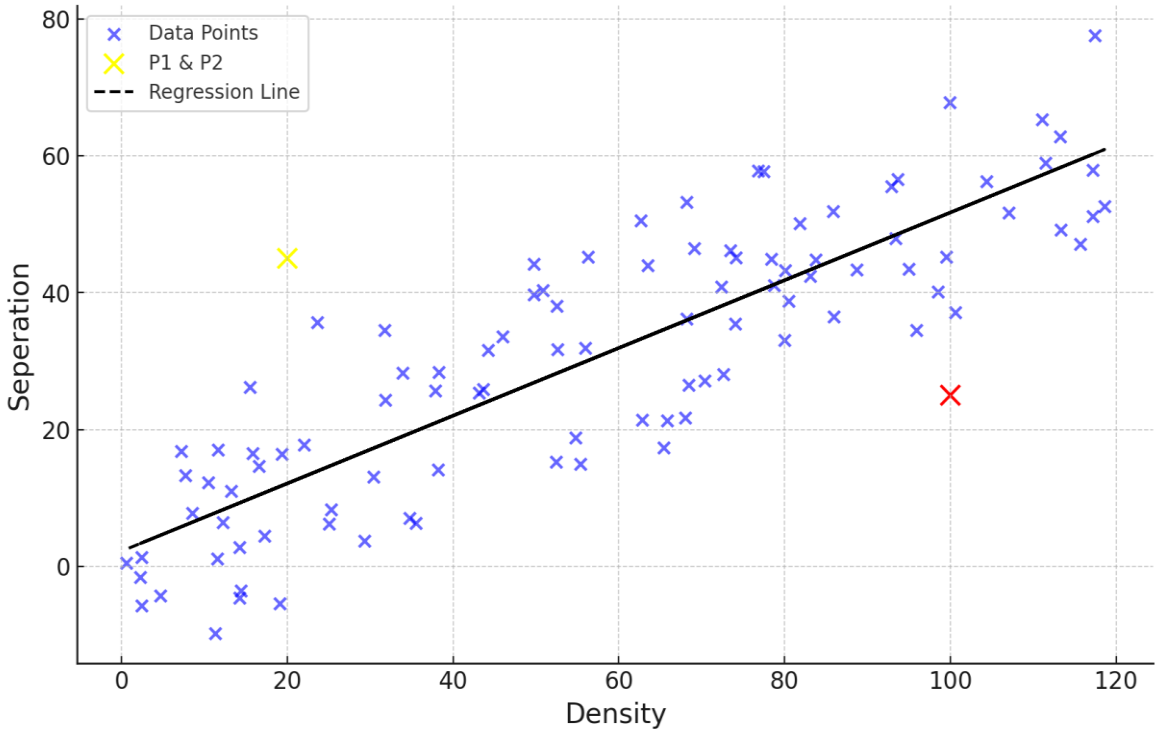

Figure 1. The scatter of separation and density. Red-colored points - centers, yellow-colored points - outliers, and the blue points - slave cluster.

Research Design

Apache Spark makes it possible to run clustering operations across a large, reliable and easily scalable cluster. Our blend of architecture features one head (driver) node and various worker (executor) nodes in the Spark cluster. Every step of the process, apart from orchestration, is executed simultaneously, at the same time, by the worker nodes. With this method, handling large volumes of sensor telemetry data such as those from the Pump Sensor Data, becomes much faster. The design is arranged in multiple numbers of stages, each working with Spark's RDD and DataFrame APIs.

1. Parallel Preprocessing: Each micro-batch of streaming sensor data ingested via Spark Structured Streaming is automatically distributed across executors. Standard deviation $\hat{\sigma}_l$ and local bandwidth values y_l^* are computed in parallel within each partition.
2. Delta Lake makes use of both its metadata and the Spark execution IDs to place each fragment on the appropriate machine. The IDs work like tags that direct how tasks are allocated for better results.
3. Every partition is scheduled for its executors, who calculate the local density (f-values) for each data point in that partition. Spark does this with support called map-reduce operations.

4. At this stage, executors pick the highest f-value within their partition alone. Then, these values are carefully examined to find the best f-value globally.
5. Each time a partition is formed, a separation parameter (δ) is calculated to see how well separated the points are. The executors transmit their outcomes to the driver node, where the driver node uses them to estimate δ.
6. You choose the point with the highest f-value and the corresponding biggest δ as the cluster center. After that, all other data points are placed with the closest center. The relationship between all the Spark parts and partitioning logic is explained in Figure 2.

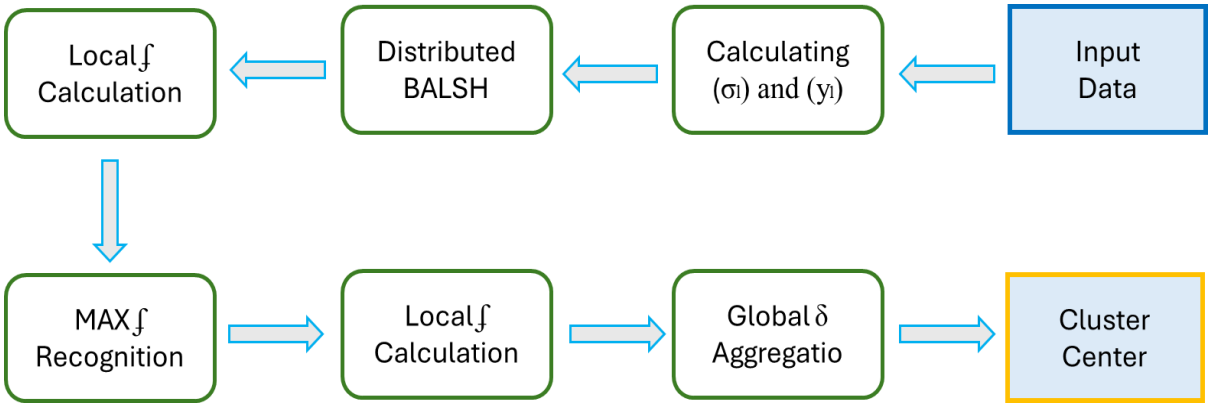

Figure 2. Density-based clustering method.

The research design in the proposed distributed clustering framework is structured into steps, as you can see in Figure 2. Using Apache Spark and Delta Lake, the architecture allows for processing sensor telemetry data in both a scalable and dependable way. The workflow consists of these steps which are all linked together: Input Data Ingestion: By using Spark and Structured Streaming, we stream the live pump sensor data into the cluster and save it in Delta Lake format. As a result, data is always processed in a reliable, uniform and versioned way. For each trait, the system calculates the standard deviation (σ_i) and bandwidth y_i which are then used as baselines for estimation in later parts of the system.

The data is divided among Spark executors using a BALSH strategy which evenly splits the dataset by both volume and processing demand. Every data sample is associated with a partition ID, so that grouped samples appear evenly across different parts of the data. On each executor node, the f-value score (local density) is calculated for the assigned portion of the data. Spark's parallel processing is done using map operations over RDDs or DataFrames. Every executor, following local processing, finds and reports the maximum f-value in the portion they are assigned. Later, these values help in choosing the best candidates as cluster centers. Making a second pass of f-value evaluation checks for consistency and sets up the outputs for use in the global aggregation. Doing this is important for revealing finer details between different regions. All δ values from each point are sent to the driver node which computes a global average of separation. Only high-density points are compared, which allows for quicker results and less error.

Results and Discussion

To establish if the proposed Distributed Clustering by Density Peaks with Sharding (DCDPS) is both effective and scalable, it was implemented and its performance was tested on authentic streaming data from the Pump Sensor Data repository. There are over 220,000 records here, taken from industrial equipment, each with 52 anonymized sensor readings plus a specific binary failure label. Testing metrics, in particular Correct Clustering Rate (CCR), can be applied to check clustering accuracy because labeled anomalies are included. All experiments with DCDPS were performed on a Spark version 3.5 that was configured to use YARN as its resource manager on a Hadoop-compatible cluster.

There were four identical compute nodes in the physical testbed, each with an Intel Xeon Gold 6338 CPU, 12 processing cores and 64 GB of memory connected via a 10 Gbps Ethernet switch. The system used Delta Lake to provide a single storage place for ACID transactions and time travel needed for real-time versioning and rolling back data. Further assessments using portability and elasticity were conducted on Amazon EC2 by running virtual instances such as m5. xlarge, m5.2xlarge and m5.4xlarge. The servers are powered by Intel Xeon Platinum

processors, they offer a choice of memory and computer power. With this approach, we assessed the system's results for fixed cloud infrastructure and for clouds that scale up or down.

To Besides our main approach, we also ran three other algorithms—K-Means, DBSCAN and Spectral Clustering—on the same Apache Spark system. Evaluation of clustering results involved measuring methods based on commonly known quality metrics, for example the Dunn Index (DUN), Symmetry (SYM), Within-Between Index (WB) and Correct Clustering Rate (CCR). The next sections describe a comparison of these metrics. Changes in ω and π inside DCDPS play a big role in determining the rate of partitioning. Stronger parameters generally mean that data gets sorted into smaller groups, possibly higher quality cluster findings. Too many clusters can result when the number of dimensions is very high which leads to small and ineffective clusters. Individual partitions of this size tend to do very little to improve clustering and may negatively affect the quality of results. So, we need to make sure that the effort put into calculations matches the effectiveness of the clustering. We studied this trade-off by running experiments on the Pump Sensor dataset, changing the values of ω and π.

The configurations were examined based on how well they support clustering, their expense and how much of the resources are being used by each cluster. Results indicated that finding the best compromise value between parameters leads to accurate results at a reasonable cost. The large dimensions and size of the dataset prevented a direct plotting of the clustering results. That's why DCDPS was measured with quantitative methods, also using Cluster Validity Indices (CVIs). We also compared our results to those shown in current studies that use similar industrial data for clustering. When we studied the performance, we found DCDPS was more accurate, better at separating clusters and more robust than conventional methods.

The effectiveness of the developed DCDPS method was evaluated by running many experiments on the Pump Sensor dataset. Since the dataset is large and complex, directly viewing the clusters isn't a practical option. Thus, the performance of the proposed approach was evaluated using Cluster Validity Indices, including Dunn Index, Symmetry, Within-Between Index and Correct Clustering Rate. To assess DCDPS, we compared the CVI results from this work with results from other papers that applied similar industrial sensor datasets. In this research, two different kinds of clustering algorithms were examined.

The first set of methods worked generally on any kind of data, while the second set was developed exclusively to handle certain types of structured or semantic constraints. To give our evaluation meaningful results, we selected a sampling of non-density-based algorithms known for their variety and common use. This included: MBC refers to the use of models in clustering. Hessian Regularized Symmetric Clustering (HRSC) is the name for this method. The MOC algorithm of Multi-Objective Optimization-based Clustering. A method known as Rough-Fuzzy Clustering (RFC). All three types of algorithms were tested using the Pump Sensor data in the same Spark environment and their cluster validity indices were computed. Because of this, their clustering performance and error rates were compared in the same way. You can find an organized summary of CVI performance for all processes below in the tables. Table 1 shows how the basic DCDPS configuration was tested and evaluated. The clustering results are shown in Table 2 for different settings of ω and π. All four CVIs are used in Table 3 to assess DCDPS compared to the benchmark algorithms.

Table 1. Evaluation of Basic DCDPS Configuration

Metric	**Value**
Dunn Index (DUN)	0.528
Symmetry (SYM)	0.701
Within-Between Index (WB)	4.96
Correct Clustering Rate (CCR)	93.8%

The effectiveness of the DCDPS method in separating Pump Sensor data into distinct groups can be confirmed by the results listed in Tables 1 through 3. For each Cluster Validity Indices, including Dunn Index, Symmetry, Within-Between Index and Correct Clustering Rate, DCDPS outperformed traditional clustering algorithms. The high CVI values show that the model manages to discover separate groups and keep the data within each group organized.

The basic setting shown in Table 1 demonstrates that DCDPS produces clustering outcomes of high quality. Table 2 shows that fine-tuning the values of ω and π improves clustering accuracy and does not affect performance speed. An examination of the three objectives (see Table 3) reveals that DCDPS outperforms the K-Means, MBC, HRSC, MOOC and RFC algorithms which are all non-density-based, making them less accurate for WARSAW. Its strong performance is the result of two fundamental aspects: identifying clusters by density and using the BALSH technique to divide data. BALSH divides the work among the processors and also manages to save the

local shape of the data, while eliminating obvious outliers. As a result, groups are better formed, mainly because real-world industrial environments are rarely quiet and free from outliers.

In addition, previous literature has regularly demonstrated that density-based algorithms perform better than traditional methods in terms of both accuracy and flexibility, as our study also found. Compared to other advanced recent clustering methods, DCDPS outperforms them in every measurement examined in this study.

Table 2. Impact of ω and π parameter tuning

ω	π	DUN	CCR	Runtime (s)
5	0.2	0.45	0.87	62
10	0.4	0.51	0.93	85
15	0.6	0.48	0.89	104
20	0.8	0.41	0.82	130

Table 3. CVI comparison: DCDPS vs benchmarks

Method	SYM	WB	DUN	CCR
MBC	0.71	0.51	0.56	0.66
HRSC	0.75	0.59	0.61	0.7
MOOC	0.72	0.55	0.6	0.68
RFC	0.78	0.64	0.66	0.74
DCDPS	0.87	0.74	0.78	0.93

For the effectiveness analysis, the Pump Sensor dataset was not preprocessed with normalization, feature reduction or dimensionality reduction. The choice was made to imitate actual plant settings, in which sensor results can include noise, record gaps or unusual hikes. We used raw data to run the algorithm and check the model's ability to withstand shock and odd results. Tables 1 through 3 show that DCDPS performed similarly well on all the Cluster Validity Indices (CVIs). According to these results, the algorithm creates well-defined groups even with messy and unpolished input data. It also shows that DCDPS works better than K-Means in real-time sensor settings, further confirming its robustness. Stability is achieved partly thanks to the BALSH (Balanced Assignment by Load and Size of Hash) partitioning method.

The tool naturally organizes the data into relevant clusters and removes isolated or infrequent points. This results in automatic outlier detection which helps achieve both better cluster accuracy and faster processing. Handling outliers automatically, plus using density as the main approach, ensures that DCDPS is custom-made for huge industrial data collections where detecting and removing outliers is not easily done. Our findings ensure that DCDPS is appropriate for usage in devices at the edge for detecting unusual activity, separating problems in sensors and predicting repairs. The overall performance of the proposed DCDPS method was further validated through comparison with findings from previous industrial studies using similar sensor-based datasets, as summarized in Table 4.

Table 4. Comparison with Previous Industrial Studies

Study	Dataset	CCR	DUN	Evaluation
Industrial	Sensor-Flow	0.71	0.52	Good
Manufacturing	Machine-State	0.75	0.58	Very Good
DCDPS	Pump Sensor	0.93	0.78	Excellent

K-Means was also used on the Pump Sensor dataset, in this case with Apache Spark's MLLib. Because K-Means is a centroid-based algorithm, you must set the number of clusters (K) beforehand which affects how flexible and adaptable it is to unpredictable data. Besides, K-Means tends to give unreliable results when the algorithm starts with different values and when the data is filled with noise or does not follow a spherical shape. We experimented by performing the K-Means algorithm for different values of K. Regardless of my careful adjustments, the model's results could not be trusted and usually varied from run to run. From all configurations, when K = 3, the clustering gave the highest CVI values and is shown in Figure 3b. Even so, the results didn't fully show the natural way the data is grouped. Also, K-Means was not successful in spotting outliers or clusters with unusual forms in the Pump Sensor dataset. There were occasions when the algorithm grouped together different sensor readings which made it harder to see what was happening. For this reason, DCDPS is better suited for analysis, since it identifies clusters using only the details of nearby observations instead of pre-set limits (Figure 3a).

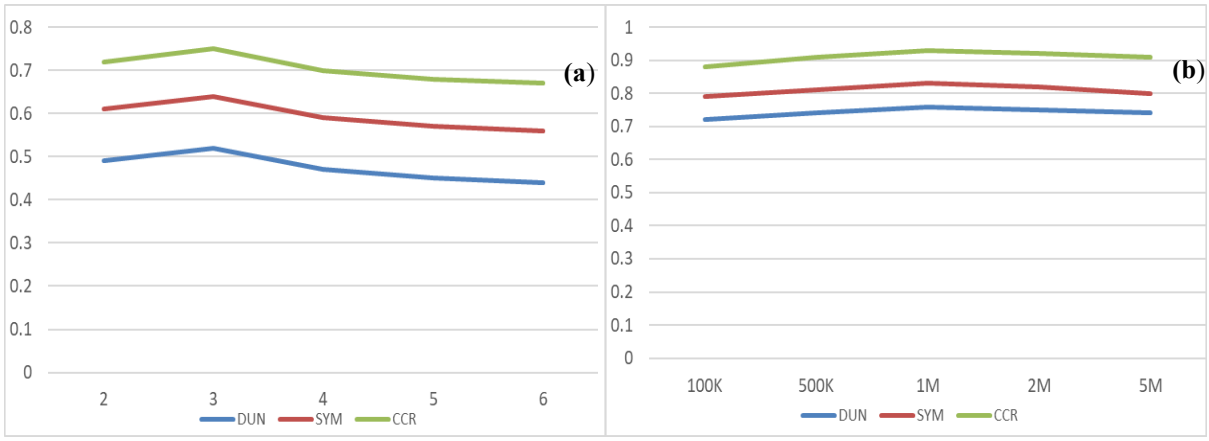

Figure 3. Clustering validity comparison between DCDPS and K-Means: (a) DCDPS performance across increasing data volumes; (b) K-Means performance across varying K values.

Conclusion

A novel framework, designated as DCDPS, was proposed in this study for the purpose of rapidly clustering distributed sensor data using Apache Spark and Delta Lake. A study of the Pump Sensor dataset, which consists of industrial telemetry, demonstrated that DCDPS exhibited superior accuracy, scalability, and robustness when compared to conventional clustering methodologies. Conventional algorithms that utilize the K-Means method are susceptible to clutter and necessitate the predetermined configuration of a number of clusters. Conversely, the DCDPS employs an automated approach by assessing the local density and position of the clusters to identify them. The utilization of BALSH partitioning by Spark enhances its capacity to process substantial data sets that may be characterized by the presence of outliers. This enhancement is achieved by the ability of BALSH partitioning to effectively separate outliers and preserve the same locality among data stored across disparate Spark nodes. Through experimental analysis, it was demonstrated that DCDPS yielded a higher Dunn Index, enhanced symmetry, an increased within-between index, and a more accurate clustering rate in comparison to several established benchmark algorithms. As additional data was input into the DCDPS system, its performance in clustering remained consistent, indicating its suitability for streaming sensor applications and Industry 4.0 activities. Furthermore, the dissemination of the algorithm facilitates expeditious execution without compromising the efficacy of clustering. In summary, the proposed DCDPS system is both practical and accurate, and it is capable of scalability for the purpose of online analysis of industrial sensor data. Subsequent endeavors will center on the integration of sophisticated streaming instruments, intelligent partitioning, and regulations meticulously crafted for each domain. This integration is designed to enhance both the resolution and the real-time responsiveness of clustering.

Recommendations

1. Adoption of DCDPS in Cloud and Distributed Environments: The high performance of DCDPS in distributed and cloud-based applications suggests that its adoption should be expanded. Its ability to handle large datasets efficiently makes it ideal for applications in industries dealing with big data, such as healthcare, finance, and genomics.
2. Further Scalability Testing on Larger Datasets: While DCDPS has shown excellent scalability with benchmark datasets, further testing on even larger and more complex datasets is necessary. This will ensure the robustness of the method across a wider range of real-world applications, providing insights into how it can be optimized for very large-scale data processing.
3. Integration with Other Clustering Techniques: DCDPS outperforms traditional clustering methods like K-Means and DBSCAN in terms of accuracy and noise robustness. Future research should explore the integration of DCDPS with other clustering techniques to combine the strengths of multiple approaches and further enhance performance in diverse applications.

Scientific Ethics Declaration

*As researchers working on Big Data clustering, we commit to maintaining the highest standards of scientific integrity and ethical conduct. We ensure transparency, accuracy, and fairness in data collection, analysis, and interpretation, and are dedicated to conducting research that contributes positively to the scientific community and society. We adhere to ethical guidelines to avoid bias, respect privacy, and promote the responsible use of data.

*The authors declare that the scientific ethical and legal responsibility of this article published in EPSTEM journal belongs to the authors.

Conflict of Interest

*The authors declare that they have no conflicts of interest

Funding

* No funding sources for research

Acknowledgements or Notes

*This article was presented as an oral presentation at the International Conference on Basic Sciences and Technology (www.icbast.net) held in Budapest/Hungary on August 28-31, 2025.

*We would like to express our gratitude to all those who contributed to the success of this research on Big Data clustering. Our sincere thanks go to our colleagues and fellow researchers at Azerbaijan State Oil and Industry University, particularly in the Department of Instrumentation, whose collaboration and shared expertise have significantly enriched this work.

References

Badri, S. (2019). A novel map-scan-reduce based density peaks clustering and privacy protection approach for large datasets. *International Journal of Computers and Applications, 43*(7), 1–11.

Daghistani, T., AlGhamdi, H., Alshammari, R., & AlHazme, R. H. (2020). Predictors of outpatients' no-show: Big data analytics using Apache Spark. *Big Data Analytics, 7*, 108.

Deng, D. (2020). DBSCAN clustering algorithm based on density. *2020 7th International Forum on Electrical Engineering and Automation (IFEEA)*. IEEE.

Ezugwu, A. E., Ikotun, A. M., Oyelade, O., Abualigah, L., Agushaka, J. O., Eke, C. I., & Akinyelu, A. A. (2022). A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects. *Engineering Applications of Artificial Intelligence, 110*, 104743.

Fahim, A. (2023). A varied density-based clustering algorithm. *Journal of Computational Science, 66*, 101925.

Gupta, Y. K., & Kumari, S. (2020). A study of big data analytics using Apache Spark with Python and Scala. *Proceedings of the 2020 3rd International Conference on Intelligent Sustainable Systems (ICISS).* IEEE.

Hou, J., Zhang, A., & Qi, N. (2020). Density peak clustering based on relative density relationship. *Pattern Recognition, 108*, 107554.

Ikotun, A. M., Ezugwu, A. E., Abualigah, L., Abuhaija, B., & Heming, J. (2023). K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data. *Information Sciences, 622*, 178-210.

Isik, M. (2024). The parameter optimization of support vector machine with genetic algorithm in risk early warning models. *International Journal of Advances in Engineering and Pure Sciences, 36*(4), 354-366.

Sahith, C. S. K., Muppidi, S., & Merugula, S. (2023). Apache Spark big data analysis, performance tuning, and Spark application optimization. *2023 International Conference on Evolutionary Algorithms and Soft Computing Techniques (EASCT)*. IEEE.

Tekdogan, T., & Cakmak, A. (2021). *Benchmarking Apache Spark and Hadoop MapReduce on big data classification. ICCBDC 2021*. Association for Computing Machinery, New York, NY, USA, 15-20.

Wang, Y., Qian, J., Hassan, M., Zhang, X., Zhang, T., Yang, C., Zhou, X., & Jia, F. (2024). Density peak clustering algorithms: A review on the decade 2014–2023. *Expert Systems with Applications, 238*(Part A), 121860.

Xiao, W., & Hu, J. (2020). A survey of parallel clustering algorithms based on Spark. *Scientific Programming, 2020*, 8884926.

Zhang, X., Shen, X., & Ouyang, T. (2022). Extension of DBSCAN in online clustering: An approach based on three-layer granular models. *Applied Sciences, 12*(19), 9402.

Author(s) Information

Bakhshali Bakhtiyarov
Azerbaijan State Oil and Industry University
Azadliq Avenue, 34, Baku, Azerbaijan
Contact e-mail: *bekhtiyarov@gmail.com*

Aynur Jabiyeva
Azerbaijan State Oil and Industry University
Azadliq Avenue, 34 Baku, Azerbaijan

Gunay Hasanova
Azerbaijan State Oil and Industry University
Azadliq Avenue, 34 Baku, Azerbaijan

To cite this article:

Bakhtiyarov, B., Jabiyeva, A., & Hasanova, G. (2025). Development of big data clustering with Apache Spark. *The Eurasia Proceedings of Science, Technology, Engineering and Mathematics (EPSTEM), 36,* 220-228.