COMPUTATIONAL INTELLIGENCE AND MACHINE LEARNING
e-ISSN: 2582-7464

Computational Intelligence and Machine Learning
Volume 6, Issue 2, October 2025

Enhancement of Density-Based Spatial Clustering of Application with Noise for Fire Risk Assessment and Warning in Metro Manila

Franchezka S. P. Flores[1*], Pinky Mae O. De Leon[2], Raymund M. Dioses[3], Vivien A. Agustin[4]

[1, 2]Student, Computer Science, Pamantasan ng Lungsod ng Maynila, Manila, Philippines
[3, 4] Professor, Computer Science, Pamantasan ng Lungsod ng Maynila, Manila, Philippines
[*]Corresponding Author Email: floresfranchezka22@gmail.com

Abstract

This study focuses on applying an enhanced Density-Based Spatial Clustering of Application with Noise for fire risk assessments and warning in Metro Manila. Unlike other clustering algorithms, DBSCAN is known for its ability to identify arbitrary shaped clusters and its resistance to noise. However, its performance diminishes when handling high dimensional data, wherein it can read the noise points as relevant data points. Also, the algorithm is dependent on the parameters (eps & minPts) set by the user, choosing the suboptimal parameters can greatly affect its clustering result. To overcome these challenges, the study proposes three key enhancements, first is to utilize multiple MinHash and Locality-Sensitive Hashing to decrease the dimensionality of the dataset, second is to implement Jaccard Similarity before applying the parameter Epsilon to ensure that only similar data points are considered neighbors, and third is to use the concept of Jaccard Neighborhood along with the parameter MinPts to improve in classifying core points and identifying noise in the dataset. The results show that the modified DBSCAN algorithm outperformed three other clustering methods, achieving fewer outliers, which facilitated a clearer identification of fire-prone areas, high Silhouette score, indicating well-separated clusters that distinctly identify areas with potential fire hazards and exceptionally achieved a low Davies-Bouldin Index and a high Calinski-Harabasz score, highlighting its ability to form compact and well-defined clusters, making it an effective tool for assessing fire hazard zones. This study is intended for assessing areas in Metro Manila that are most prone to fire risk.

Keywords

Cluster, DBSCAN, Enhancement, Fire, Jaccard Similarity.

INTRODUCTION

Advanced approaches like clustering algorithms can be used to analyze data and identify patterns in several incidents, including crimes and disasters. These types of algorithms are versatile because they can be applied across various areas to assess large datasets, grouping them based on their attributes and similarities. An example of such an algorithm is the Density-Based Spatial Clustering algorithm, or DBSCAN, which identifies spatial patterns based on the density of data points in a dataset. It is known for its ability to form clusters with arbitrary shapes, which makes it more effective than other clustering algorithms [1]. This makes it suitable for handling spatiotemporal data when assessing a crowded area like Metro Manila because it specializes in this type of data. In the context of risk evaluation, it analyzes areas based on factors such as the number of incidents that have occurred. The regions where several incidents are recorded will be clustered and labeled as areas at high risk for the said incident.

Fire incidents are one of the most common disasters that occur, especially in tropical countries. Metro Manila is known for having a high population rate and closely built houses and buildings. It also has a high record of fire incidents, mainly in areas that have several informal settlers, making the city prone to these accidents. One way to address this issue is implementing a fire assessment in all the areas within the city, wherein all the data such as the date and time of incident, location, and cause of fire incident are collected to produce outcomes and predictions.

Implementing DBSCAN can be advantageous, but just like other algorithms, it has problems and limitations that can affect its performance. However, it has shortcomings like issues like parameter tuning and noise sensitivity and can be computationally expensive that can result in generating sub-optimal results [2]. DBSCAN requires the setting of two parameters: epsilon (ε) and minPts to identify the minimum distance and number of data points for it to be clustered. While being computationally expensive requires a huge number of computational resources and time to process and generate results. If these factors are not addressed, they can lead to having inaccurate clusters.

To deal with these issues, the proponents propose an enhanced Density-Based Spatial Clustering algorithm by incorporating methods and techniques like density reduction, Jaccard similarity, and Jaccard neighbor. By having a density reduction first before the clustering, it can lessen the data points that need to be analyzed, decreasing the complexity. The Jaccard Similarity can decrease the reliance of the algorithm to the parameter epsilon. It works by using the features or similarity of data points, rather than their spatial distance, as basis for clustering. While the concept of Jaccard neighbors can help the minPts in classifying the core points. By implementing these enhancements, it can significantly improve the performance of DBSCAN in fire assessment in

Metro Manila, thereby providing warnings to the citizens of Metro Manila.

The performance of DBSCAN can be affected by several factors, including parameters, noise, and complexity, which can lead to suboptimal clustering. The algorithm still depends on manually setting the threshold for Jaccard similarity. Furthermore, the study is limited to analyzing data from areas within Metro Manila, which can restrict the scope of fire assessment and warning implementation. This study focuses on three main objectives and does not aim to address all existing aspects of the DBSCAN algorithm. The fire incident dataset used in this study was obtained from the Bureau of Fire Protection (BFP) and contains a total of 18,738 recorded fire incidents. The implementation is done in Python, relying on open-source tools and available libraries within the VSCode environment, and is not yet optimized for large-scale systems.

METHODS AND METHODOLOGY

Figure 1(a) shows the process behind the enhanced Density-Based Spatial Clustering Algorithm (DBSCAN) proposed in this study. There are three (3) techniques inside the green box, which are MinHash signature generation, Locality-Sensitive Hashing, and Core Point Classification (Jaccard Similarity), applied in DBSCAN's clustering process to improve its performance.

The process starts by gathering data, then it will undergo preparation through binarization. Next, it will go through dimensionality reduction by applying MinHash signature generation and Locality-Sensitive Hashing, wherein each data point has its own compact signature for quick comparison. Afterward, LSH will group the data points based on signatures to easily identify similar ones, speeding up the process. Then, the concept of Jaccard Similarity is applied in core point classification and neighbor validation, ensuring the data points are compared and clustered based on their features. After identifying the core points, each of them expands based on the minPts. The remaining data points that are not part of a cluster will be labeled as noise. The process ends with the result of data points clustered based on the similarity of their features, which can be used for fire risk assessment.

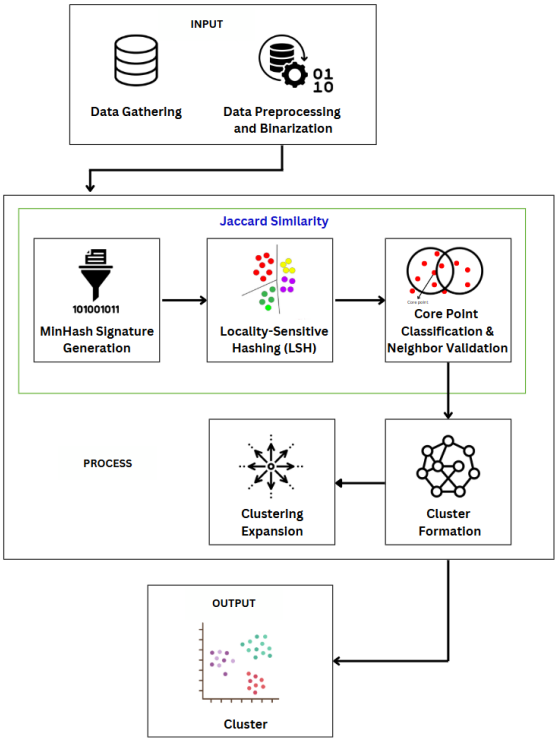

Figure 1(a): Proposed System Architecture

The proposed DBSCAN algorithm intergrates new methods aimed at enhancing clustering performance by addressing the existing problems and limitations of the original algorithm. The techniques MinHash and Locality-Sensitive Hashing (LSH) are used to make DBSCAN more effective and efficient in handling high-dimensional data. Jaccard Similarity is implemented with the epsilon parameter to help identify similar data points, while the concept of Jaccard Neighbors is employed along with the minPts parameter to help classify core, border, and noise points.

For Objective 1, MinHash and Locality-Sensitive Hashing (LSH) are utilized to handle high-dimensional data. After converting each data point into a binary format, MinHash generates a unique hash signature for each one. This is done by applying 128 different hash functions or permutations, resulting in a 128-length signature that improves the accuracy of similarity classification. For Objective 2, Jaccard Similarity is used alongside the epsilon (ε) parameter to identify valid neighbors. The enhanced algorithm computes the Jaccard Similarity between a given data point and the other data points in the same bucket. If the similarity score is greater than or equal to the epsilon, the compared data point is considered a valid neighbor. Otherwise, it is excluded. For Objective 3, the concept of Jaccard neighbors is used in classifying core, border, and noise points. After determining the valid neighbors of each data point, the algorithm checks how many neighbors each point has. If the number of valid neighbors is greater than or equal to the minPts parameter, the data point is classified as a core point. If the number of valid neighbors of a data point is less than minPts but greater than zero, it will be classified as a border point.

1. **Input**
   1.1. High-dimensional categorical dataset.
   1.2. Parameters:
      1.2.1. *eps* (Jaccard similarity threshold).
      1.2.2. *minPts* (minimum number of points to form a core point).
      1.2.3. *num_perm* (number of hash functions for MinHash).
      1.2.4. *lsh_threshold* (similarity threshold for LSH).
2. **Data Preprocessing**
   2.1. Convert the categorical dataset into binary format using **one-hot encoding**.
   2.2. Store the binary matrix as *X_binarized*.
3. **MinHash Signature Generation**
   3.1. For each data point in *X_binarized*, generate a **MinHash signature**:
      3.1.1. Apply *num_perm* hash functions to generate the MinHash signature for each data point.
   3.2. Store the generated MinHash signatures for future use.
4. **LSH Index Construction**
   4.1. Initialize a **Locality-Sensitive Hashing (LSH)** index.
   4.2. For each MinHash signature: **Objective 1**
      4.2.1. Insert the MinHash signature into the LSH index.
5. **Neighbor Search Using LSH**
   5.1. For each data point:
      5.1.1. Query the LSH index to retrieve a list of **candidate neighbors** based on similar MinHash signatures.
   5.2. Filter the candidate neighbors:
      5.2.1. Compute the **Jaccard similarity** between the data point and each candidate neighbor. **Objective 2**
      5.2.2. Retain only neighbors with Jaccard similarity greater than or equal to *eps*.
   5.3. Store the valid neighbors for each data point.
6. **Core Point Classification**
   6.1. For each data point: **Objective 3**
      6.1.1. If the number of valid neighbors is greater than or equal to *minPts*, classify the point as a **core point**.
      6.1.2. If the number of valid neighbors is less than *minPts* but greater than zero, classify the point as a **border point**.
      6.1.3. If the point has no valid neighbors, classify it as **noise**.
7. **Clustering Process**
   7.1. Initialize a cluster label counter, starting at 0.
   7.2. For each unvisited data point:
      7.2.1. If the point is a **core point**:
         7.2.1.1. Assign it the current cluster label.
         7.2.1.2. Expand the cluster by recursively visiting all its valid neighbors.
      7.2.2. For each neighbor of the core point:
         7.2.2.1. If the neighbor is a core point and unvisited, add it to the cluster and visit its neighbors.
         7.2.2.2. If the neighbor is a border point and unvisited, add it to the cluster.
   7.3. If the point is not a core point, skip to the next unvisited point.
   7.4. After expanding the cluster, increment the cluster label counter.
8. **Post-processing**
   8.1. Label all remaining **noise points** as -1.
   8.2. Return the cluster labels for all data points.
9. **Output**
   9.1. The algorithm outputs the clusters formed, with noise points labeled as -1.

Standard Density Based Spatial Clustering of Application with Noise

DBSCAN is a clustering algorithm known for its capability to form clusters regardless of the shape. It works by clustering data points based on their distance to other data points. DBSCAN starts by setting the parameters epsilon and minPts, which are the basis for determining if a data point will be classified as a core point, border point, or noise. After setting the parameters, it will randomly pick an unvisited data point as a starting point. If the number of neighbors within the radius (epsilon) is greater than or equal to minPts, it will be identified as a core point. Next, the cluster will start to expand by adding the neighbors of the core point as part of the cluster. Each neighbor can be classified as either a core point or a border point. If it's classified as a core point, its neighbors will be added to the current cluster; otherwise, it will be marked as a border point, at which the expansion of the cluster stops. The process repeats until all the data points are visited. It ends with the clusters being formed, and the remaining data points that don't belong to any cluster will be identified as noise.

Euclidean Distance

As previously mentioned, DBSCAN works by grouping data points based on their distance to the surrounding data points. The Euclidean distance is used to measure the spatial proximity of the data points in a straight line, making it a good method in identifying dense areas in the dataset. The formula computes the distance between data points p and q, where n is the number of dimensions, while p_i and q_i represents the values of data points' feature.

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \quad (1)$$

After computing the distance, the result is used in data point classification and neighborhood formation.

Epsilon Neighborhood

Once the distance between the datapoints is calculated, the clustering will start based on the epsilon radius, ε. Any distance that is equal to or less than ε will be considered part of the ε-neighborhood of a point. If a point has minPts or more neighbors within its ε-neighborhood, it is classified as a core point. Data points that have a distance greater than ε are either classified as border points or noise. Border points lie within the ε-neighborhood of a core point but don't have enough neighbors to be considered core points. Points that do not lie within the ε-neighborhood of any core point and have fewer than minPts neighbors are classified as noise.

$$d(p,q) \leq \in \quad (2)$$

Core Point Condition

A core point is crucial for clustering, given that it is considered to be the starting point from which the expansion of the cluster will begin. If a data point has a number of neighbors that is greater than or equal to minPts, it is considered as a core point.

$$Number\ of\ neighbors \geq minPts \quad (3)$$

Literature Review

According to Sharma, A. (2024), DBSCAN is known for its robust and reliable clustering because it can precisely represent clusters based on its density, along with its ability to identify & separate the noise points from the data [3]. Each datapoint can be classified as core, border, or noise point; this algorithm uses local density to group data points, while outliers are labelled as noise. Also, it can deal with different levels of density within datasets, making it useful in real-world scenarios where data is allocated unevenly. Its robustness and reliability make DBSCAN an effective algorithm for clustering data with several noise points, and

even when there are more outliers than the actual data.

DBSCAN also has limitations. Its performance heavily relies on two critical parameters: MinPts (minimum number of points in a cluster's neighborhood) and Eps (neighborhood radius). Choosing inappropriate values for these parameters can significantly impact the clustering results (Liu et al., 2020) [4]. Furthermore, DBSCAN can struggle with datasets where clusters have significantly different densities. Clusters with lower density might be missed altogether (Yi et al., 2020) [5]. Finally, DBSCAN's effectiveness can deteriorate in high-dimensional spaces due to the "curse of dimensionality" (Yan et al., 2023) [6]. In conclusion, DBSCAN provides a robust approach for clustering spatial data with various shapes and noise. However, its effectiveness depends on choosing appropriate parameters and its suitability for datasets with varied densities and high dimensionality.

Data-driven approaches, like those utilizing Jaccard similarity, offer promising solutions to these challenges. Jaccard similarity captures the ratio of shared features between data points, making it robust to feature scaling and less sensitive to the absolute values within the data (Huang, 2018) [7]. By leveraging the inherent structure of the data itself, Jaccard similarity allows the clustering algorithm to learn an appropriate similarity measure, potentially leading to more accurate and interpretable clusters (Calmon & Albi, 2020) [8]. This data-driven approach can be particularly advantageous for complex datasets where pre-defining parameters might be difficult or misleading.

Design Methodology

This section discusses the techniques added to DBSCAN to improve its clustering performance. There are three approaches: Locality-Sensitive Hashing (LSH), MinHash, and Jaccard Similarity. The first two techniques are applied for better handling of high-dimensional data. LSH creates a compact signature for each data point based on their features, while MinHash groups these signatures based on their similarity, allowing quicker comparison of the data points. On the other hand, Jaccard Similarity is used in core point classification and neighborhood validation by evaluating the similarity between data points and clustering them accordingly.

MinHash and Locality-Sensitive Hashing

MinHash is applied multiple times to each data point to create a unique hash signature based on its features, making it easier for DBSCAN to assess the similarity of high-dimensional data. LSH is used to organized and group the hash signatures based on their similarity. The LSH threshold indicates if two data points belong to the same cluster. By utilizing MinHash and LSH, the data points are pre-clustered before the main clustering process starts, speeding up computations and enhancing DBSCAN's ability to handle high-dimensional data efficiently.

Jaccard Similarity

The concept of Jaccard Similarity is implemented in core point classification and neighbor validation, which are crucial parts in the clustering process. Jaccard Similarity measures the similarity of data points by computing the size of their intersection divided by the size of their union. With this, instead of solely relying on the defined epsilon, the clustering will be based on the actual data points, resulting in better clustering of DBSCAN. Jaccard neighbors indicates whether two data points are neighbors based on their similarity. Meanwhile, Jaccard Similarity also plays a role in neighbor validation by verifying if two data points are considered neighbors based on the computed Jaccard Similarity along with the epsilon.

Testing

The proponents developed 2 programs, one for visualization of the clustering and another for comparing clustering performances and application for fire risk assessment. There are two distinct datasets used for each system. For the first system, synthetic data is used by using the function make_blobs from the sklearn.datasets module in the scikit-learn library. The data generated applied in both dbscan and enhanced dbscan hav

e the same value of dimensions and data points. There are 5050 data points from make_blobs. The second dataset used is from BFP wherein all the data from 2019 to 2023 are combined, having a total of 18738 data points and 5 dimensions which are Date_of_Fire, City_or_Municipality, Property_Type_General, Property_Type_Sub, Cause, Day_of_Week, and Time_of_Alarm_Range_24H. It is saved in a csv file where it is uploaded to the program.

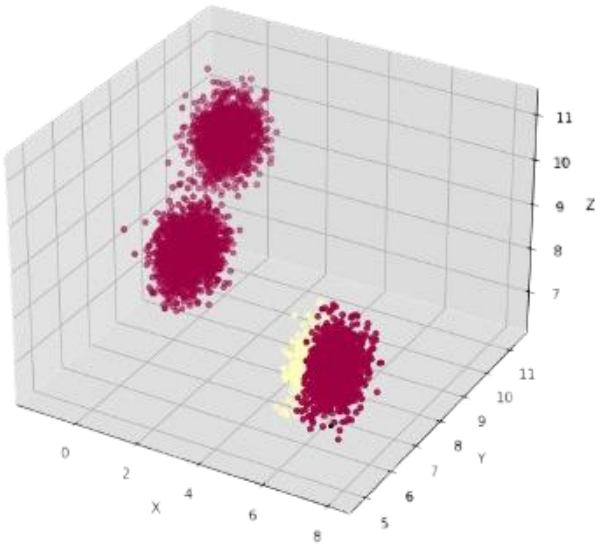

Figure 1(b): Original DBSCAN's Clustering Results

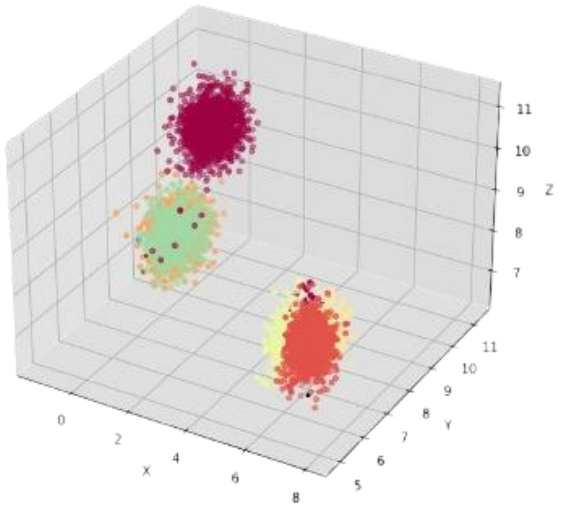

Figure 1(c): Enhanced DBSCAN's Clustering Results

Comparing the two figures above, Figure 1(b) has distinct and clear clusters, and most data points are grouped, however, the clusters are simplistic and uniform. Noise points (in yellow) are dispersed around the major clusters. Figure 1(c) shows clearer clusters revealing finer details, shown by multiple colors. It seems to have found more sub-clusters and offers less noise points than the first. Based on this, it suggests that Figure 1(c) shows more precise and nuanced clustering results.

The proponents developed the second program to focus on the performance metrics comparison and actual application of the original and enhanced DBSCAN in fire risk assessment by using the dataset from the Bureau of Fire Protection. The dataset includes all the records of fire incidents in Metro Manila from the year 2019 to 2023.

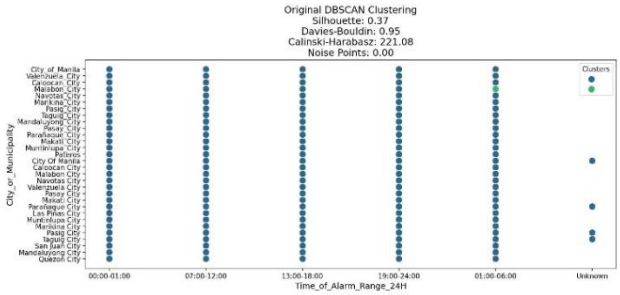

Figure 1(d): Standard DBSCAN Applied in Fire Risk Assessment

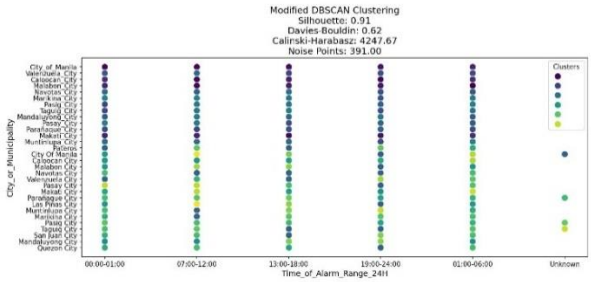

Figure 1(e): Enhanced DBSCAN Applied in Fire Risk Assessment

RESULTS

Table 1 highlights the performance comparison between the Original and Enhanced DBSCAN using two data sets. The first data set has 7 dimensions, while the second has 50 dimensions, with both containing 18738 data points. Additionally, both data sets have the same values for epsilon (0.85) and MinPts (20), so it centers on how the two algorithms handle data sets with low and high dimensions.

Table 1: Clustering Performance Metrics using Data Sets with Low and High Dimensions

	Method (Eps: 0.85, MinPts: 20)	**Silhouette Score**
Objective 1: Enhancing DBSCAN with MinHash and LSH for High-Dimensional Data	Original DBSCAN (7D)	0.37
	Enhanced DBSCAN (7D)	0.91
	Original DBSCAN (50D)	Not Applicable
	Enhanced DBSCAN (50D)	0.74
Legend:	**The closer the Silhouette Score is to 1, the more compact and well-separated the clusters are.**	

In Objective 1, which enhanced the ability of DBSCAN to handle high-dimensional datasets, the Silhouette Score was chosen as the performance metric because it measures how similar data points are within each cluster. A score close to 1 indicates that the data points are well clustered, while scores near -1 tells that there's a huge possibility that the data points are assigned to the wrong cluster. This metric is useful for validating the performance of clustering algorithms. The result shows that the original DBSCAN (7 dimensions) has a silhouette score of 0.37, while the enhanced DBSCAN (7 dimensions) achieved a higher SC of 0.91, clearly suggesting that the clusters are well separated. Furthermore, in the data set with 50 dimensions, the enhanced DBSCAN also has a higher SC (0.74) than the original which resulted in "Not Applicable" due to not forming any clusters. These results show that adding techniques such as LSH and MinHash improved the enhanced DBSCAN's performance in handling high dimensional data sets. This reflects the efficiency of the enhancements in creating well-defined and non-overlapping clusters

Table 2 shows the performance comparison between the Original and Enhanced

Table 2: Clustering Performance Metrics with Low and High Epsilon

	Method (MinPts: 20)	**DBI**
Objective 2: Using Jaccard Similarity to Optimize	Original DBSCAN ($\varepsilon = 0.20$)	0.85
	Enhanced DBSCAN ($\varepsilon = 0.20$)	0.77

	Method (MinPts: 20)	DBI
Epsilon Selection in DBSCAN	Original DBSCAN (ε = 0.85)	0.95
	Enhanced DBSCAN (ε = 0.85)	0.62
Legend:	**The lower the DBI, the better the clustering quality—indicating compact, well-separated clusters.**	

For Objective 2, which addressed the sensitivity of DBSCAN to the parameter epsilon, the Davies-Bouldin Index (DBI) was selected as the performance metric because it examines the two important factors affected by epsilon: cluster separation and compactness. The Davies-Bouldin Index assesses the clusters' separation and compactness based on their density and highlights overlapping clusters, which is best used in density-based methods. Having an index of 0 ensures the formed clusters are compact and far apart. The lower the index, the better the ratio of separation and compactness of the clusters. In the first test with low epsilon (0.20), the original DBSCAN had a DBI of 0.85, while it improved to 0.77 with the enhanced DBSCAN. In the second test with high epsilon (0.85), it still shows that enhanced DBSCAN performed better than the original because its DBI decreased to 0.62 from 0.95 Both results show that the enhanced DBSCAN produced indexes closer to 0, making the formed cluster less dense. This indicates that the enhanced DBSCAN achieves better performance than the original because it has optimal clustering performance with no overlapping clusters. Such a result demonstrates a significant improvement in distinguishing and separating the data points.

Table 3 displays a comparison between the performance of the Original and Enhanced DBSCAN using two data sets. The first data set has a small MinPts value of 9, while the second data set has a value of 20. However, both have the same epsilon value of 0.20 to emphasize the algorithms' sensitivity to MinPts.

Table 3: Clustering Performance Metrics with Low and High MinPts

	Method (Eps: 0.20)	CHS
Objective 3: Using Jaccard Neighbor Concept to Optimize Epsilon Selection in DBSCAN	Original DBSCAN (MinPts = 9)	221.08
	Enhanced DBSCAN (MinPts = 9)	5903.73
	Original DBSCAN (MinPts = 20)	221.08
	Enhanced DBSCAN (MinPts = 20)	4247.67
Legend:	**The higher the CHS, the better the clustering performance—indicating well-defined, dense clusters.**	

In Objective 3, which resolves the sensitivity of DBSCAN to another parameter called MinPts, the performance metric used is the Calinski-Harabasz Score (CHS), which evaluates clustering performance by comparing the variance of a pair of clusters to the variance of another pair. Higher CHS value tends to have better separated and more compact clusters, while lower values indicate less effective clustering performance because the clusters are not properly separated or compact. For MinPts = 9, the original DBSCAN had a CHS of 221.08, while the enhanced DBSCAN increased to 5903.73. For MinPts = 20, the enhanced DBSCAN (4247.67) also resulted into having a significantly higher CHS than the original (221.08). The results suggest that the enhanced DBSCAN latter has a better clustering performance since the data points in each cluster are related to each other, and each of the formed clusters is distinct from the others.

Table 4.5 shows the top 5 clusters based on fire incidents in Metro Manila from 2019 to 2023. The results revealed a distinct pattern regarding fire causes, peak occurrence time (month and day), and high-risk-prone cities in Metro Manila.

Table 4: Cluster Profile Summary of Fire Incidents using Enhanced DBSCAN

C	Year	Total incidents	Top fire cause	Peak month	Peak time	Poperty type	Property type sub	Most affected location
3	2020	1546	Electrical ignition caused by arcing	5	Evening	Non_structural	Electrical pole	Quezon city
43	2021	1474	Electrical ignition caused by arcing	7	Evening	Non_structural	Electrical pole	City of manila
19	2020	1324	Electrical ignition caused by arcing	6	Evening	Non_structural	Electrical pole	Quezon city
33	2021	1249	Electrical ignition caused by arcing	5	Evening	Non_structural	Electrical pole	Quezon city
48	2023	633	Electrical ignition caused by arcing	10	Morning	Residential	Single and two family dwelling	Quezon city
44	2023	549	Overheated home appliances	12	Morning	Residential	Single and two family dwelling	City of manila

The most common fire cause across multiple clusters was electrical ignition due to arcing, which affected non-structural properties like electrical poles. Quezon City and Manila, the two most densely populated cities in Metro Manila, are prominent which have a record the highest number of fire incidents. In fact, high population density in an area increases the demand for electricity. Residential fires were also significant, showing an increase in 2023 in fire incidents in single and two-family dwellings. This is connected to urbanization and informal settlements inside Metro Manila, as electrical overloading is also more rampant.

Regarding seasonal aspects in correlation with fire incidents, the peak was observed from May to July and December, corresponding to extreme weather conditions in Metro Manila and increased electricity usage. Cooking-related fires were most frequent in March and December. Furthermore, electrical fires were most frequent in the evening. These patterns highlight the need for differentiated fire prevention and seasonal fire safety campaigns, especially in densely populated and high-risk areas.

CONCLUSION

After applying the new methods; MinHash, Locality-Sensitive Hashing, and Jaccard Similarity, DBSCAN's performance improved in handling high-dimensional data, parameter dependency, and clustering performance. The Enhanced DBSCAN outperforms the original clustering, as shown by a Silhouette Score of 0.91, which indicates that the clusters are well defined.

With a small distance between points of the same cluster and a higher distance between different clusters, without overlapping. By comparison, the standard version of DBSCAN has a lower Silhouette Score of 0.37, indicating more diffuse clusters. The Davies-Bouldin Index (DBI) of the Enhanced DBSCAN (0.62) is slightly lower than that of the Original DBSCAN (0.95); nevertheless, the improvement is still significant. The original DBSCAN's DBI being close to 1 suggests that its clustering quality is not well-defined. Additionally, the Enhanced DBSCAN's Calinski-Harabasz Score (CHS) of 4247.67 is much higher than that of the original (221.08), meaning the clusters are very well separated. In the aspect of cluster identification, the original DBSCAN algorithm identified only 2 clusters and no noise points, while the Enhanced DBSCAN identified 6 clusters and 391 noise points. This indicates that it is better able to discover unique structures and adapt to the complexity of the dataset. Although it handles marginally more noise, the significant enhancements in cohesiveness, cluster separation, and pattern detection demonstrate that the Enhanced DBSCAN is the superior algorithm in this context.

Thus, the Enhanced DBSCAN proves to be a more reliable and efficient approach for handling complex clustering tasks, especially when dealing with higher-dimensional data and noise.

Based on the findings and limitations of this study, it is recommended that future researchers explore additional clustering enhancements beyond MinHash, Locality-Sensitive Hashing, and Jaccard Similarity to further improve DBSCAN's performance with high-dimensional data. Investigating parameter-free techniques may also enhance clustering accuracy and adaptability. The enhanced DBSCAN can be applied to broader domains such as disaster risk reduction, healthcare analytics, and environmental monitoring to validate its effectiveness. Moreover, integrating this method into a web-based GIS dashboard with real-time IoT sensor data can support timely, data-driven decisions for urban risk assessment and management.

Acknowledgement: We thank Pinky Mae De Leon and Franchezka Flores for their contributions to this work. Special thanks to Sir Raymund M. Dioses and Ma'am Vivien A. Agustin for their assistance.

Funding Statement: The authors did not receive financing for the development of this research.

Data Availability: There are two datasets used in this study: one generated by utilizing the make_blobs function in python, and the other is the actual fire incident records (2019-2023) from the Bureau of Fire Protection-National Capital Region.

Conflict of Interest: The authors declare that there is no conflict of interest.

Miscellaneous: There are no other figures, tables, supplements, appendices, or annexes placed after the reference.

REFERENCES

[1] Deng, D. (n.d.). DBSCAN Clustering Algorithm Based on Density | IEEE Conference Publication | IEEE Xplore. Retrieved January 05, 2025 from https://ieeexplore.ieee.org/document/9356727.

[2] Hu, L., Liu, H., Zhang, J., & Liu, A. (2021). KR-DBSCAN: A density-based clustering algorithm based on reverse nearest neighbor and influence space. Retrieved January 06, 2025 from https://www.researchgate.net/publication/354120988_KR-DBSCAN_A_density-based_clustering_algorithm_based_on_reverse_nearest_neighbor_and_influence_space.

[3] Sharma, A. (2024, February 8). How to master the popular DBSCAN Clustering algorithm for machine Learning. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/09/how-dbscan-clustering-works/.

[4] Liu, Y., Wu, Z., & Xiong, T. (2020). A parameter selection method for DBSCAN algorithm based on grid clustering. Journal of Big Data, 7(1), 1-13. https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0236-x.

[5] Yi, S., Zhang, C., & Qin, L. (2020). A New DBSCAN Algorithm with Improved Performance for Datasets with Varied Densities. IEEE Access, 8, 149222-149232.

[6] Yan, X., Xu, J., & Tian, Y. (2023, March). A Survey of Density-Based Clustering Algorithms for High-Dimensional Data. In 2023 International Conference on Big Data and Smart Computing (BigDSC) (pp. 261-268). IEEE.

[7] Huang, A. (2018). Similarity measures for text analysis. Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), 67–73. https://aclanthology.org/W05-1203.pdf.

[8] Calmon, A. P., & Albi, G. (2020). Cluster analysis for knowledge discovery: A review of recent unsupervised learning methods. Frontiers in Artificial Intelligence, 3, 16. https://www.frontiersin.org/articles/10.3389/fmed.2021.764934 8.