TMJM THE MINDANAWAN JOURNAL OF MATHEMATICS
OFFICIAL JOURNAL OF THE MINDANAO STATE UNIVERSITY - ILIGAN INSTITUTE OF TECHNOLOGY

Vol. 6 (2024), no. 2, pp. 113–133 ISSN: 2094-7380 (Print) | 2783-0136 (Online)
https://doi.org/10.62071/tmjm.v6i2.530

Enhancing Density-Based Spatial Clustering of Applications with Noise (DBSCAN) through Kernel Density Estimation

Mary Jean O. Eniola*, Aljo Clair P. Pingal, and Catherine R. Caño

Department of Mathematics and Statistics
MSU-Iligan Institute of Technology, 9200 Iligan City, Philippines
maryjean.eniola@g.msuiit.edu.ph, aljoclair.pingal@g.msuiit.edu.ph
catherine.cano@g.msuiit.edu.ph

Received: 29th April 2024 Revised: 9th January 2025

Abstract

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is one of the prominent methods that are efficient at uncovering clusters of various shapes, however, it faces limitations when dealing with datasets containing clusters of varying densities. To address this limitation, this study integrates kernel density estimation into the DBSCAN algorithm to enhance its capacity to capture density variations and handle irregularly shaped clusters. Specifically, we employ Kernel Density Estimation (KDE) using Epanechnikov as the kernel function and the grid search method with cross-validation for the bandwidth selection, along with the added density threshold. The simulation study shows that the proposed procedures were able to specify the number of clusters even for varying densities correctly. Moreover, empirical results show that the proposed clustering procedure enhanced the DBSCAN algorithm and gave meaningful results.

1 Introduction

Big data may be characterized as a term for huge or complex data collection where traditional data processing applications are insufficient. Due to the various features of big data, such as its volume, variety, variability, value, velocity, and complexity, it is exceedingly challenging to analyze data and extract information using conventional data mining approaches [39]. The discovery of interesting characteristics and patterns in large spatial databases is known as spatial data mining. It has many applications, some of which include seismology (clustering seismic events that are concentrated along faults), minefield detection (grouping mines in a minefield), and astronomy (grouping of stars in galaxies) [9]. In data analysis and data mining, cluster analysis is a technique used to classify objects or data points into groups called clusters. Clustering algorithms are machine learning algorithms and may be divided into four broad classifications: a) the partition-based algorithms or partitional clustering methods, b) density-based algorithms, c) hierarchical-based algorithms, and d) grid-based algorithms [43]. One of the most well-known clustering algorithms used in data mining is the DBSCAN (Density-Based Spatial Clustering of Applications with Noise). DBSCAN is a clustering algorithm designed to discover clusters

*Corresponding author
2020 Mathematics Subject Classification: 62G07, 62H30
Keywords and Phrases: DBSCAN, kernel density estimation, grid search, Epanechnikov
This research is supported by the DOST-ASTHRDP Student Research Support Fund

of varying shapes and noise in a spatial database. The algorithm works based on the density of objects, it requires two parameters to build a dense region: a) the epsilon (ϵ) which defines the neighborhood around a data point, and b) the minimum number of neighbors ($MinPts$), which defines the minimum number of data points required to form a distinct cluster [10]. DBSCAN defines a cluster by starting with an arbitrary point p in the database then its surrounding area is calculated using radius ϵ. If there are at least $MinPts$ in the neighborhood, then p is designated as the *core point*, and a cluster formation begins by retrieving all density-reachable points from p with respect to ϵ and $MinPts$. If no points are density-reachable from p, then p becomes a *border point*. If p is neither a core point nor a border point, p becomes a *noise point* or an *outlier* and DBSCAN proceeds on to the next point of the database. This process is repeated until all points have been visited and processed. DBSCAN has also been demonstrated to work in practice, and received the SIGKDD test-of-time award in 2014 [37]. It is widely used due to its efficiency and ability to discover clusters of arbitrary shapes. However, the original algorithm has limitations when dealing with datasets that contain clusters of varying densities [28];[24]. Since the DBSCAN algorithm relies on two key parameters, the ϵ and $MinPts$ to form a cluster it can assume that clusters are dense regions divided by areas of lesser density, which may not necessarily be the case in real-world datasets with irregular shapes and densities.

Kernel Density Estimation (KDE) is a non-parametric approach used to estimate the probability density function (PDF) of a continuous random variable [4]; [35]; it is a powerful and most widely-used non-parametric estimation technique of density-based spatial point patterns [21]. The kernel function and the coefficient of smoothness or bandwidth are two fundamental concepts in kernel estimation. Samiuddin and El-Sayyad in 1990 [36], introduced the idea of inadmissible kernels. A kernel is inadmissible if another kernel gives uniformly a small mean squared error (MSE). In their paper, they assume a large sample for the mean squared error of kernel estimators, they show that some kernels are inadmissible in the sense that estimates based on them can always be uniformly improved and that an Epanechnikov kernel is the only kernel that is admissible. Scott in 2015 [38], developed a thorough analysis showing that the Epanechnikov kernel is the most efficient when pdf estimation is at stake. The Epanechnikov kernel function has the property of being optimal in the sense that it produces or minimizes the mean square error (MSE) [42]; [6]. This means it tends to produce estimates with lower overall error compared to other kernels and is not dependent on any type of data. The GridSearchCV is a function in Scikit-learn (or SK-learn) model selection packages. It is a method that methodically constructs and evaluates a model for each combination of algorithm parameters specified in a grid [32]. Due to its strength and exhaustive search [41], it will be utilized for the identification and defining the coefficient of smoothness or bandwidth parameter.

This study proposed a spatial clustering algorithm called DBSCAN via Kernel Density Estimation (DBSCAN-KDE). This paper focuses on incorporating kernel density estimation into the DBSCAN algorithm making it ideal for capturing density variations to handle datasets with clusters of different densities and irregular shapes and to adapt it to seismic data by adding a density threshold that can improve the algorithm in identifying areas of high seismic activity. Specifically, we employ Epanechnikov as the kernel function. It is chosen because it has many successful applications that produce better performance and results [45]; [15]; [27] and since it is shown in [42]; [6]; [3] to give minimal mean square error (MSE) when being compared to other kernels. Moreover, the GridSearchCV is used in this study to identify the optimal bandwidth. The methodology and algorithms are designed to handle datasets that contain spatial attributes.

The structure of this paper is as follows: Section 2 presents the Related Literature, Section

3 discusses the Methodology, Section 4 outlines the Results obtained and their discussions, and finally, Section 5 concludes with the Summary and Recommendations of the proposed work.

2 Related Works

Density-Based Clustering

Density-based clustering is one of the areas that has been investigated by developing algorithms that can be implemented and used to analyze real-world datasets. These algorithms have evolved through time to become more efficient and effective clustering methods. Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a prominent density-based clustering technique developed by Martin Ester, Hans-Peter Kriegel, Jöorg Sander and Xiaowei Xu in 1996 [10]. It is one of the most widely used density-based clustering algorithms in data mining. Several studies have been conducted to extend and enhance the DBSCAN algorithm, [22], introduced a new algorithm called VDBSCAN (Varied Density Based Spatial Clustering of Application with Noise) to handle varied-density datasets. The basic idea of VDBSCAN is that it uses a k-dist plot to select suitable values of the ϵ parameter for different densities. However, the k parameter in VDBSCAN is a user dependent and it can reduce the efficiency of the permanent ϵ value. [11], presented in their study a method to enhance the DBSCAN algorithm by automatically selecting the input parameters and finding density-variable clusters. The key concept is that ϵ needs to be identified before using the traditional DBSCAN algorithm. A k-dist graph needs to be created for every point to detect the various ranges of ϵ values automatically and to identify the number of clusters of varying densities including noise. The goal is to determine the *"knees"* for estimating the set of ϵ parameters. Then DBSCAN algorithm is adopted for each ϵ value to ensure that all clusters with respect to corresponding density are clustered. The same authors published an article that same year that reviewed the proposed method and its implementation. They discovered that the value k still required a user to enter its value and suggested that future research would determine the value of k internally, making the entire process automatic [12]. In 2010, [31] proposed a density varied DBSCAN called DVBSCAN (Density Variation Based Spatial Clustering of Applications with Noise) which is capable of handling local density variation within the cluster. The algorithm begins by selecting core objects, which are points that have a minimum number of points within a specified ϵ. These core objects are used to form clusters. Nevertheless, similar to any algorithm, DVBSCAN is sensitive to parameter selection, therefore the choice of selecting appropriate values of these parameters can pose a challenge. In 2013, [5] and colleagues, presented a new clustering approach for classifying focal mechanisms from large moment tensor catalogs, intending to automatically detect families of earthquakes having comparable source geometry, identify the direction of most active faults, and detect temporal variations of the rupture processes. Within their article, they limit the discussion on the application of DBSCAN and integrate it with the definition of different metrics that can be applied to determine the distance between source models based on various moment tensor representations. Furthermore, clustering techniques were primarily been employed in seismology to uncover patterns in seismic events and distinguish foreshocks and aftershocks based on their spatial location [29]; [18]; [20].

Kernel Density Estimation and Its Application

Kernel density estimation or KDE is a nonparametric density estimator that does not require the underlying density function to be from a parametric family. KDE will automatically learn the shape of the density from the data. Because of its nonparametric character, KDE is a preferred technique for data obtained from a complicated distribution [7]. Several kernel functions may be found in the relevant literature, including the Gaussian kernel, the Epanechnikov kernel, the Biweight kernel, the Triangular kernel, the Rectangular kernel, the Inverse Gaussian kernel, and others. In the study of [44] in 2018, the two common methods used for bandwidth selection are the Rule-of-thumb widely referred to as the Silverman rule-of-thumb, and the Least Squares Cross-Validation ($LSCV$) method. Silverman's rule-of-thumb is used when the Gaussian kernel function is chosen. It is based on the asymptotic mean integrated square error, AMISE. [40] in 2018, obtain the values of the smoothing or bandwidth parameter h as, $h = 1.06 \centerdot \widehat{\sigma} \centerdot n^{-\frac{1}{5}}$, where $\widehat{\sigma}$ is the standard deviation of a sample and n is the sample size and applied interquartile range IQR and proposed a slightly reduced value of the bandwidth parameter: $h = 1.09 \centerdot min(\widehat{\sigma}, \frac{IQR}{1.34}) \centerdot n^{-\frac{1}{5}}$. On the other hand, the least squares cross-validation method can be used in any kernel function, it uses the integrated square error, ISE. $LSCV$ is also referred to as unbiased cross-validation with the form:

$$LSCV(h) = \frac{1}{n^2}\sum_{i \cdot j}\int_{-\infty}^{+\infty} K(x, x_i)K(x, x-j)dx - \frac{2}{n(n-1)}\sum_{i}\sum_{j \neq i} K(x_i, x_j)$$

.

In the study conducted by [1] in 2015, an evaluation and comparison were performed on the Gaussian Mixture Model (GMM) and Gaussian Kernel Density Estimator algorithms for detecting anomalies in annotated datasets from the Maritime domain. Both algorithms have a large number of false detections as well as anomalies that are not detected. In 2018, [26], presented an algorithm for clustering based on univariate kernel density estimation called ClusterKDE. It is an iterative technique in which a new cluster is obtained by minimizing a smooth kernel function or the bandwidth parameter at each stage. Although they employed a univariate Gaussian kernel, any bandwidth parameter was utilized in their applications. The proposed technique has the advantage of not requiring the number of clusters to be determined beforehand. In a 2019 study by [46], a novel algorithm was introduced to adaptively determine parameters in the DBSCAN method. The algorithm is structured into four components: disk generation, density information estimation within disks, selection of high-quality disks, and parameter determination through Gaussian kernel density estimation. The experimental findings indicate that the proposed algorithm enhances the DBSCAN's precision. In a 2020 study headed by [23], the focus was on detecting abnormal movement speeds using a coastal surveillance radar. They combined the KDE with a Gaussian kernel function and Silverman's rule-of-thumb as the bandwidth and DBSCAN algorithm to identify the usual movement speeds. The test results indicating a false alarm rate (FAR) equal to zero imply the effectiveness of the approach in accurately identifying usual movement speeds. [27] in 2021, explore the application of the Epanechnikov kernel for estimating the Probability Density Function (PDF) in equalization and blind source separation tasks. The authors investigated the Epanechnikov kernel features, they compared its performance to the common Gaussian (Normal) kernel on the estimation of a standard normal distribution, and the results shall be measured in terms of *Mean Integrated Squared Error (MISE)*. $MISE = E[\int(\hat{f}_X(z) - f_X(z))^2 dz$, where $\hat{f}_X(z)$ is the estimated pdf attained considering the samples of the X random variable. In their experiment, the kernel size varies from 0.05 to 2 in steps of 0.05, and 50 to 500 number of samples, *MISE* values considering the optimal kernel size for the Epanechnikov and Gaussian kernels resulted that Epanechnikov kernel can obtain lower *MISE* levels. The *MISE* performance superiority of the Epanechnikov kernel was made clear in their experiment, it was also noted that for all cases the Gaussian kernel *MISE* error is considerably higher than that of the Epanechnikov kernel. Thus, they utilized the Epanechnikov kernel in their study and demonstrated that using the Epanechnikov kernel for PDF estimation can improve the performance of equalization and blind source separation tasks. [33] in 2023, investigates functions and kernels that can be used to create a *"smoother"* local average of a function. The Fourier analysis is used to find a constant for each kernel and then uses compactness to find a kernel that minimizes this constant. The minimizing kernel is found to be remarkably close to the Epanechnikov kernel in statistics. They consider functions $f : \mathbb{Z} \to \mathbb{R}$ and kernels $u : \{-n, \cdots, n\} \to \mathbb{R}$ normalized by $\sum_{l=-n}^{n} u(l) = 1$, that makes the convolution $u * f$ a "smoother" locale average of a function f. They identified which choice of kernels u that most effectively smooths the second derivative in the following sense and there is a section in their study that they prove a Theorem offering which kernel is closely optimal and easy to implement in practice. The optimal kernel u_n has a complicated equation, but it resembles a parabola, and they discovered that selecting weights by sampling from a parabola works exceptionally well for smoothing the Laplacian of a given function. This choice of weights provides the discrete normalized Epanechnikov kernel $E_n : \{-n, \cdots, n\} \to \mathbb{R}$ defined by: $E_n(k) = \frac{3}{n(4n^2-1}(n^2 - k^2)$. A theorem from their analysis demonstrates that the Epanechnikov kernel is less than 2% worse than ideal in smoothing the Laplacian, providing another incentive to utilize it in practice. [47] in 2023, studied which kernel function to select for the Model Linear Regression (MLR) from two perspectives: first, "Which kernel achieves small error?" and second, "Which kernel is computationally efficient?". The first perspective was investigated and the *asymptotic mean squared error (AMSE)* was adopted as a criterion of an estimation error. Accordingly, the *AMSE* of $\hat{\theta}_n$ is given as a sum of the squared asymptotic bias and the asymptotic variance, as $E[\|\hat{\theta}_n - \tilde{\theta}\|^2] \approx \frac{h_n^4 U^2 \|\tilde{A}^{-1}\tilde{b}\|^2}{4} + \frac{V tr(\tilde{A}^{-1}\tilde{C}\tilde{A}^{-1})}{nh_n^3}$. The result of the first investigation is that the optimal among the non-negative kernels that minimize the *AMSE* is the Biweight kernel function. The second investigation is the perspective "Which kernel is computationally efficient?" they considered and used the *iteratively reweighted least-squares algorithm (IRLS)* for the *MLR* and showed that the *IRLS* guarantee convergence for a wide class of kernels. They then prove a theorem that states the *IRLS* with an Epanechnikov kernel converges in a finite iteration process. Empirical simulation studies have confirmed that using a biweight kernel offers good estimation accuracy, while the Epanechnikov kernel is noted for its computational efficiency.

3 METHODOLOGY

Epanechnikov Kernel Density

Let X_1, X_2, …,X_n in the set of $\mathbb{R}$ be a univariate random sample from a distribution with density f we wish to estimate. The kernel estimator with kernel K as the smooth function was defined by Silverman (2018) as,

$$\widehat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right) \tag{1}$$

where h is the bandwidth or the smoothing parameter and n is the sample size. The Epanechnikov kernel function is defined as:

$$K(u) = \left\{ \begin{array}{ll} \frac{3}{4}(1-u^2), & \text{if } \mid u \mid \leq 1, \\ 0, & otherwise. \end{array} \right\} \tag{2}$$

where $u = \frac{x-X_i}{h}$. The Epanechnikov kernel function is selected because it possesses the lowest mean square error (MSE) [42], [6], [15].

Bandwidth Parameter

In a work published in 2019, Rajan and colleagues [32] emphasized the Grid SearchCV, a method that systematically constructs and assesses a model for every combination of algorithm parameters specified in a grid. GridSearchCV is employed in the research paper by Veeralagan and Priya in 2022 [41] due to its strength and its exhaustive search. GridSearchCV is the process of optimizing hyperparameters for a given model. The GridSearchCV is a function in Scikit-learn (or SK-learn) a Python module that provides a collection of advanced machine learning algorithms for handling medium-scale supervised and unsupervised tasks. [30].

In the context of Kernel Density Estimation, the GridSearchCV is employed because it can systematically explore different bandwidth values, and the exhaustive search across possible bandwidths ensures that the best-performing parameter is chosen. GridSearchCV will fit the KDE model for each bandwidth value and then use cross-validation to estimate the model's performance. The optimal bandwidth is determined by selecting the bandwidth value that produces the best cross-validation performance.

Clustering Evaluation Metric

The Silhouette Index, developed by Peter J. Rousseeuw in 1987 [34], is a method used to evaluate the quality of clusters for partitioning techniques. It is a measure of how similar an object is to its cluster (tightness) in comparison to other clusters (separation). This silhouette identifies which objects are well-contained within their cluster and which are spread out between clusters. Silhouette index ranges from -1 to 1, the range of $0.71 - 1.00$ signifies excellent split of clusters, the range of $0.51 - 0.70$ indicates a reasonable split of clusters, the range of $0.26 - 0.5$ suggests weak split of clusters and below this range implies bad split of clusters [25]. However, it has a limitation: it prefers spherical cluster formations and similar sizes, which may not always be the case in practical scenarios where clusters can have different shapes and sizes. Lengyel and Botta-Dukat in 2019 [19] developed a generalization of the Silhouette width by applying a generalized mean, allowing them to create a flexible formula that adjusts the sensitivity of the index between connectedness and compactness, enabling higher values for non-spherical clusters. It has a parameter p that determines the importance of connectedness and compactness.

Let W be a sample of positive real numbers $w_1, w_2, \ldots, w_n$ and p an element of affinely extended real numbers. The generalized mean of degree p is as follows:

$$M^p(w_1, w_2, \ldots, w_n) = \left(\frac{1}{n} \sum_{k=1}^{n} w_k^p \right)^{\frac{1}{p}} \tag{3}$$

For $p = 0$ and $p = -\infty$ the following exceptions are to be made:

$$M^0(w_1, w_2, \ldots, w_n) = \lim_{p \to 0} M^p(w_1, \ldots, w_n) = \left(\prod_{k=1}^{n} W_k \right)^{\frac{1}{n}} \tag{4}$$

$$M^{-\infty}(w_1, w_2, \ldots, w_n) = \lim_{p \to -\infty} M^p(w_1, \ldots, w_n) = min(w_1, \ldots, w_n) \tag{5}$$

$$M^{\infty}(w_1, w_2, \ldots, w_n) = \lim_{p \to \infty} M^p(w_1, \ldots, w_n) = max(w_1, \ldots, w_n) \tag{6}$$

The generalized mean incorporates the values of familiar summary statistics outlined in Table 1.

Table 1: Special cases of the generalized mean

p	Descriptive Statistic
$-\infty$	Minimum
-1	Harmonic Mean
0	Geometric Mean
1	Arithmetic Mean
2	Quadratic Mean
∞	Maximum

The original Silhouette width is a specific instance where average distances within and between groups are computed with $p = 1$. By setting the value of parameter p in a lower value, greater significance is assigned to objects in close proximity, diminishing the influence of distant neighbors, including outliers. Conversely, when p is greater than 1, greater emphasis is placed on the compactness criterion.

Main Algorithm of DBSCAN-KDE

Algorithm 1 DBSCAN with Kernel Density Estimation (KDE)

```
Input: Database D with n spatial attributes .
 1: Density Estimation with KDE:
 2:   Define Kernel Function:
 3:     Use the Epanechnikov as the kernel function.
 4:   Set the bandwidth $h$ parameter using GridSearchCV.
 5: Compute the Kernel Density Estimate:
 6:   for each data point $x$ in $D$ do
 7:       Calculate the kernel density estimate $\hat{f}(x)$ using the general formula:
              $$\widehat{f}(x) = \frac{1}{nh}\sum_{i=1}^{n} K\big(\frac{x - X_i}{h}\big)$$
 8:       Compute the kernel function $K(u)$.
 9:       Sum the kernel function values for each data point $x_i$ in $D$.
10: Plot the kernel density estimates
11: Define Density Threshold:
12:   After KDE, set a density threshold based on the estimated densities.
13: Identify Cluster Centers:
14:   Using the density threshold, find points with a density higher than the threshold.
15: Set Parameters:
16:   Set the Epsilon parameter ($\epsilon$).
17:   Set $MinPts$, the minimum number of neighbors a data point must have within the $\epsilon$
    parameter.
18: Apply DBSCAN Algorithm:
19:   Apply DBSCAN on the identified cluster centers.
20:   Use parameters $\epsilon$ and $MinPts$ for defining cluster centers as core, border, and noise
    points.
21: Point labels:
22:   for each data point $p$ in $D$ do
23:       find all data points within the $\epsilon$.
24:       if $p$ has at least $MinPts$ within $\epsilon$ then $p$ is a core point.
25:       if $p$ has less than $MinPts$ within $\epsilon$ then $p$ is a border point.
26:       otherwise $p$ is a noise point.
27: Cluster Expansion:
28:   for each core point $c$ if not yet assigned to a cluster do
29:       create a new cluster $C$ and add $c$ to $C$
30:       if neighboring point $q$ of $c$ is also a core point
31:       add it to $C$
32:       if $q$ is a border point but is within $\epsilon$ distance of $c$
33:       add it to $C$
34: Border Points:
35:   these points are added to $C$ but are not used to expand the cluster any further.
36: Output: Collection of clusters and some noise points.
```

Algorithm 1 outlines the steps to follow in executing the DBSCAN with KDE. Initially, the algorithm requires an input of a dataset D with coordinates. Then, a KDE method will serve as a pre-processing step in the DBSCAN process. This method is primarily initiated in each data point x in D, the Epanechnikov kernel function is then defined and the GridSearchCV is used for selecting the bandwidth h parameter. The kernel density estimates for each data point x are calculated by summing up the kernel function $K(u)$ values (see eq'n. 2) evaluated at the distance between the data point and all other points which are then normalized by the bandwidth h parameter and the total number of data points n (see eq'n. 1), the resulting estimate represents the probability density at the given data point. The method iterates through all data points in D, calculating the kernel density estimates for each x. Finally, the list of the kernel density estimates for each data point is then plotted by the algorithm, displaying regions of high and low density. The density threshold (dt) value is set based on the estimated kernel densities of each point to be produced by the KDE method. It functions as a cut-off value to distinguish high-density areas. The dt parameter will filter the data points and data points with density estimates that exceed dt value will represent potential cluster centers, indicating areas of high seismic activity.

In the DBSCAN process, the ϵ and the $MinPts$ are then defined by the user. DBSCAN is then applied to the identified cluster centers, and each cluster center is labeled as *core*, *border*, and *noise* points or an *outlier*. This method is iterated until all cluster centers are labeled. The algorithm proceeds to construct a cluster by picking an arbitrary point y. If y is a *core* point and has not been assigned to a cluster, a new cluster C is initiated, and y is added to C. Then the algorithm identifies all density-reachable points from y with respect to the ϵ and $MinPts$ parameters and adds it to C. If y is a *border* point, no points are density-reachable from y, then y is added in C. The algorithm then moves on to the next point q of the database. Through this iterative process, the algorithm expands clusters around *core* points, progressively adding neighboring points until all reachable points within the ϵ neighborhood are included.

The result of the process is a collection of clusters, where each cluster stands for a different seismic event. In assessing the clustering stability produced by the improved algorithm, the Silhouette width using generalized mean will be used to ensure the reliability and reproducibility of the clustering solution. Furthermore, the performance of the improved DBSCAN-KDE techniques will be compared against the traditional DBSCAN algorithm.

4 Results and Discussions

This section is structured into two sub-sections: a) Simulation results of the traditional DBSCAN algorithm and the DBSCAN-KDE algorithm over 100 repetitions; and b) the outcome of applying DBSCAN-KDE in the recorded earthquake data of the Philippines from the year 1975 to December 5, 2023, to discover significant seismic events.

Simulation Results Comparing Traditional DBSCAN and DBSCAN-KDE algorithms

A spatial distribution of 13 geographical clusters is simulated where each cluster represents a different population center. These clusters are centered at specific coordinates: $(40, -60)$, $(30, -120)$, $(42, -100)$, $(45, -125)$, $(40, -81)$, $(25, -65)$, $(23, -90)$, $(19, -107)$, $(55, -74)$, $(56, -96)$, $(7, -88)$, $(60, -112)$, $(10, -121)$ and the standard deviation SD within each cluster indicates how spread out the points are around each center.

The experiment aimed to evaluate four scenarios for all 13 clusters: (a) S1 consists of the same values of $SD(\sigma)$ ranging from 1.0 to 3.0 with a density threshold (dt) set to 0, (b) S2 consists of different values of SD ranging from 1.0 to 3.0 with a density threshold (dt) set to 0, (c) S3 consists of the same values of SD ranging from 1.0 to 3.0 with a density threshold (dt) set to 0.0005, (d) S4 consists of different values of SD ranging from 1.0 to 3.0, with a density threshold (dt) set to 0.0005.

The first two scenarios (S1, S2) are intended to test the performance of both algorithms when the dt is at 0, while the other two scenarios (S3, S4) are tested when the dt is at 0.0005. This design aimed to evaluate the execution of both algorithms under varying conditions. The Silhouette Width (SW) value with parameter $p = 2$ is the average SW computed across 100 replications with $n = 22,800$ data points. The parameter $p = 2$ is used to put more importance to objects in close proximity (compactness), consider the nonspherical shapes of clusters [19] in the empirical application, and produce consistent results throughout the experiment. The value used for the $MinPts$ parameter is equivalent to $ln(n)$ where n is the size of the database [2].

Scenario 1 (S1): Comparison of the DBSCAN and the DBSCAN-KDE algorithms with Equal Standard Deviation (SD) within each cluster and a $dt = 0$

Below is a table that presents the summary results of the performance of DBSCAN and the DBSCAN-KDE algorithms when dt is set to 0.

Table 2: Results of DBSCAN and DBSCAN-KDE Application on Synthetically Generated Data Comprising *13 Unequal Clusters Sizes* each with equal *Standard Deviations (SD)* and setting the Density Threshold (dt)=0 where $n = 22,800$.

Cases	DBSCAN			DBSCAN-KDE		
	MinPts	Epsilon	Silhouette Width	MinPts	Epsilon	Silhouette Width
Case 1: SD = 1.0	10	0.4	0.89437	10	0.4	0.89437
Case 2: SD = 1.3	10	0.7	0.85821	10	0.7	0.85821
Case 3: SD = 1.6	10	0.7	0.82804	10	0.7	0.82804
Case 4: SD = 1.9	10	0.8	0.79591	10	0.8	0.79591
Case 5: SD = 2.2	10	0.9	0.76377	10	0.9	0.76377
Case 6: SD = 2.5	10	1.0	0.73146	10	1.0	0.73146
Case 7: SD = 2.8	10	1.1	0.67465	10	1.1	0.67465
Case 8: SD = 3.0	10	1.1	0.55036	10	1.1	0.55036

Table 2 illustrates the effectiveness of both algorithms in capturing the predetermined number of clusters. The resulting SW shows that, when dt is set to 0 and each case comprises clusters with equal dispersion SD values, the DBSCAN and the DBSCAN-KDE algorithms performed similarly. The SW values in cases 1 through 6 further suggest that the clusters generated by both algorithms imply strong compactness and are separated from other clusters. Conversely, in cases 7 and 8 where there is an increase in the dispersion SD of the data points within each cluster, the SW is smaller. The reason for this decreased value is that it becomes more challenging to identify natural cluster boundaries when a large value of dispersion SD is uniformly present in each cluster. It makes data points from adjacent clusters appear to be closer than they are. This can lead to overlapping regions where it becomes challenging for the algorithm to decide whether a point belongs to one cluster or another based solely on two parameters: the ϵ and $MinPts$ to construct a cluster. Generally, both algorithms performed effectively under low and moderate levels of dispersion in the data points within each cluster. On the other hand, when cases where each cluster has a uniformly high level of dispersion, both algorithms face the challenge of accurately distinguishing the predetermined clusters.

Scenario 2 (S2): Comparison of the DBSCAN and the DBSCAN-KDE with Unequal Standard Deviation (SD) within each cluster and $dt = 0$

The table below displays the results of the performance of the DBSCAN and DBSCAN-KDE algorithms under varying SD when the dt is set to 0.

Table 3: Results of DBSCAN and DBSCAN-KDE Application on Synthetically Generated Data Comprising *13 Unequal Clusters Sizes* each with Unequal *Standard Deviations (SD)* and setting the Density Threshold (dt)=0 where $n = 22,800$.

Cases	DBSCAN			DBSCAN-KDE		
	MinPts	Epsilon	Silhouette Width	MinPts	Epsilon	Silhouette Width
Case 1 (S2)	10	0.7	0.82746	10	0.7	0.82746
Case 2 (S2)	10	0.8	0.80721	10	0.8	0.80721
Case 3 (S2)	10	0.8	0.79553	10	0.8	0.79553
Case 4 (S2)	10	1.0	0.78016	10	1.0	0.78016
Case 5 (S2)	10	1.0	0.76945	10	1.0	0.76945

Table 3 suggests both the DBSCAN and DBSCAN-KDE algorithms perform comparably when clusters are subjected to varying dispersion SD and the dt is set to 0. This is evident from the SW values. Additionally, it can also be observed that under this scenario both algorithms produced a high value of SW indicating that the objects are well-matched to their cluster and are well-separated from other clusters. This implies that both algorithms were able to determine the predefined number of clusters highlighting their capabilities of handling datasets with varying dispersion.

In scenarios 3 and 4, a density threshold parameter is set according to the estimated density value produced by the KDE.

Scenario 3 (S3): Comparison of the DBSCAN and the DBSCAN-KDE with Equal Standard Deviation (SD) within each cluster and $dt = 0.0005$

Table 4: Results of DBSCAN and DBSCAN-KDE Application on Synthetically Generated Data Comprising *13 Unequal Clusters Sizes* each with equal *Standard Deviations (SD)* and setting the Density Threshold (dt)=0.0005 where $n = 22,800$.

Cases	DBSCAN			DBSCAN-KDE		
	MinPts	Epsilon	Silhouette Width	MinPts	Epsilon	Silhouette Width
Case 1: SD = 1.0	10	0.4	0.89437	10	0.4	0.89457
Case 2: SD = 1.3	10	0.7	0.85821	10	0.7	0.86621
Case 3: SD = 1.6	10	0.7	0.82804	10	0.7	0.84240
Case 4: SD = 1.9	10	0.8	0.79591	10	0.7	0.82131
Case 5: SD = 2.2	10	0.9	0.76377	10	0.7	0.80294
Case 6: SD = 2.5	10	1.0	0.73146	10	0.7	0.78784
Case 7: SD = 2.8	10	1.1	0.67465	10	0.7	0.77596
Case 8: SD = 3.0	10	1.1	0.55036	10	0.7	0.76988

Table 4 displays the results of both algorithms under the uniform condition of SD and the dt parameter set to 0.0005. It further shows that the setting of the dt parameter influences the DBSCAN-KDE algorithm's ability to detect the predefined clusters as observed in the SW value. Though both algorithms produced high values it is noticeable that there is a slight improvement in the performance of the DBSCAN-KDE algorithm particularly in cases 7 and 8. This discernible improvement demonstrates the effectiveness of incorporating KDE as a pre-processing step. As a result, the DBSCAN-KDE approach improved clustering results by maintaining high values of SW despite the high dispersion of data points within clusters.

Scenario 4 (S4): Comparison of the DBSCAN and the DBSCAN-KDE with Unequal Standard Deviation (SD) within each cluster and $dt = 0.0005$

Table 5: Results of DBSCAN and DBSCAN-KDE Application on Synthetically Generated Data Comprising *13 Unequal Clusters Sizes* each with Unequal *Standard Deviations (SD)* and setting the Density Threshold (dt)=0.0005 where $n = 22,800$.

Cases	DBSCAN			DBSCAN-KDE		
	MinPts	Epsilon	Silhouette Width	MinPts	Epsilon	Silhouette Width
Case 1 (S2)	10	0.7	0.82746	10	0.7	0.84377
Case 2 (S2)	10	0.8	0.80721	10	0.7	0.83136
Case 3 (S2)	10	0.8	0.79553	10	0.7	0.82246
Case 4 (S2)	10	1.0	0.78016	10	0.8	0.81606
Case 5 (S2)	10	1.0	0.76945	10	0.8	0.81304

Table 5 summarizes the performance of the DBSCAN and DBSCAN-KDE algorithms under varying SD when the dt is set to 0.0005. Based on the presented outcomes, it can be observed that both algorithms show commendable performance as reflected by their respective SW values. However, the DBSCAN-KDE algorithm consistently achieves higher SW values than the DBSCAN algorithm. This discrepancy becomes particularly noticeable across a range of SD values, demonstrating that the DBSCAN-KDE approach is adept at handling datasets with varied degrees of dispersion. Adding KDE as a pre-processing step effectively enhanced the DBSCAN-KDE algorithm's capacity to capture the true number of clusters.

DBSCAN and the DBSCAN-KDE algorithms applied to Earthquake Dataset

Earthquakes are catastrophic natural events that can result in substantial harm to infrastructure, loss of life, and disruptions to socioeconomic systems. It has also been occurring at an unprecedented rate in recent years, and many researchers have been studying their characteristics. According to the Philippine Atmospheric, Geophysical, and Astronomical Services Administration (PAG-ASA), the Philippine Area of Responsibility (PAR) refers to the most confined monitoring zone, with its boundary closest to the Philippine Islands. Because of its geographical location and physical environment, the Philippines is vulnerable to a variety of natural hazards. In this research, the efficiency of the proposed clustering algorithm DBSCAN with kernel density estimation is tested in producing meaningful clusters using the spatial locations (latitude and longitude) of the earthquake occurrences as key features to understand the geographic patterns. The data used comes from the Earthquake Information (dost.gov.ph) of the Philippine Institute of Volcanology and Seismology (PHIVOLCS) and spans the period from the year 1975 to December 5, 2023. The primary focus is on earthquakes that have a magnitude of $M \geq 4$ and/or have the potential to cause damage.

The DBSCAN algorithm

The DBSCAN algorithm was applied to the spatial component of the earthquake data of the Philippines with an ϵ and $MinPts$ value of 0.3 and 50, respectively. The resulting cluster solutions are presented in Figure 1.

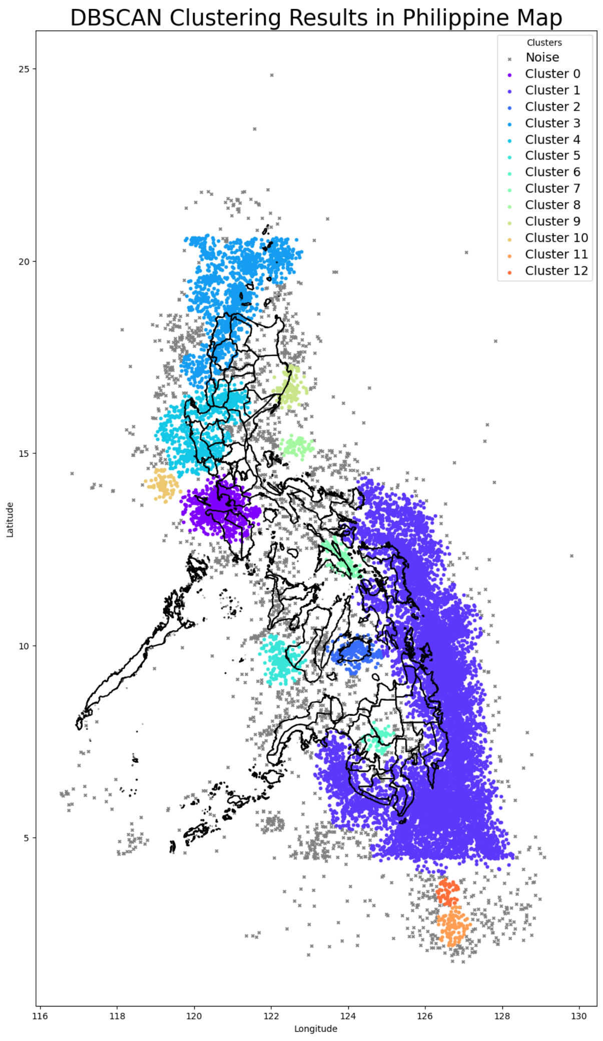


Figure 1: Earthquake Plot of the Philippines in the DBSCAN Algorithm

In the figure presented above, the DBSCAN algorithm identified 13 clusters and some *noise* points. It can be observed that the algorithm displays its ability to detect clusters of arbitrary shapes; however, the obtained SW of 0.46262 with $p = 2$ suggests a lower value. This value is below the midpoint of the SW range, indicating weak separation between the clusters and the background noise.

The DBSCAN-KDE algorithm

Kernel density estimation is primarily applied in the spatial component of the earthquake data of the Philippines using Epanechnikov as a kernel function and the GridSearchCV method for bandwidth selection. A smooth density distribution map across the dataset has resulted from this process. Taking into account the estimated density value, the results in KDE highlight the regions with higher and lower seismic activity. Regions with higher density indicate potential earthquake hotspots.

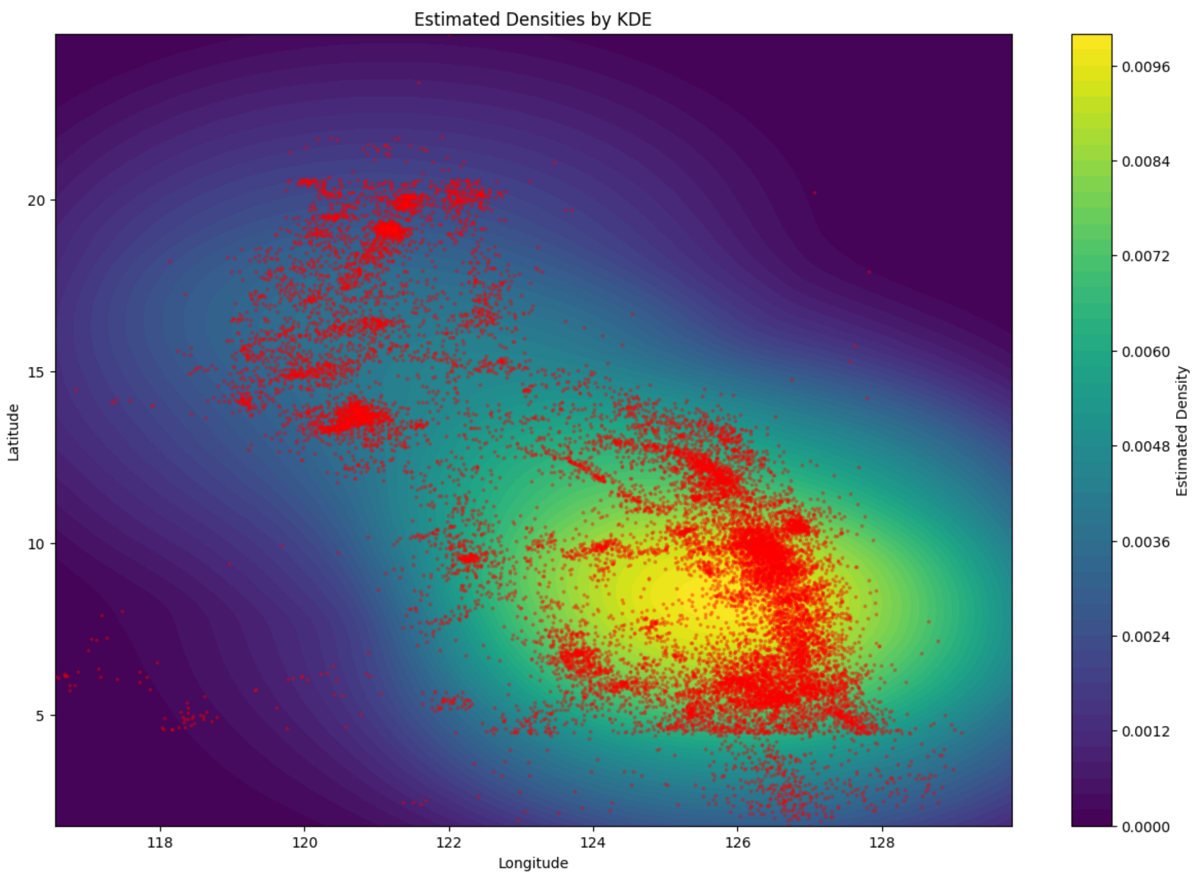
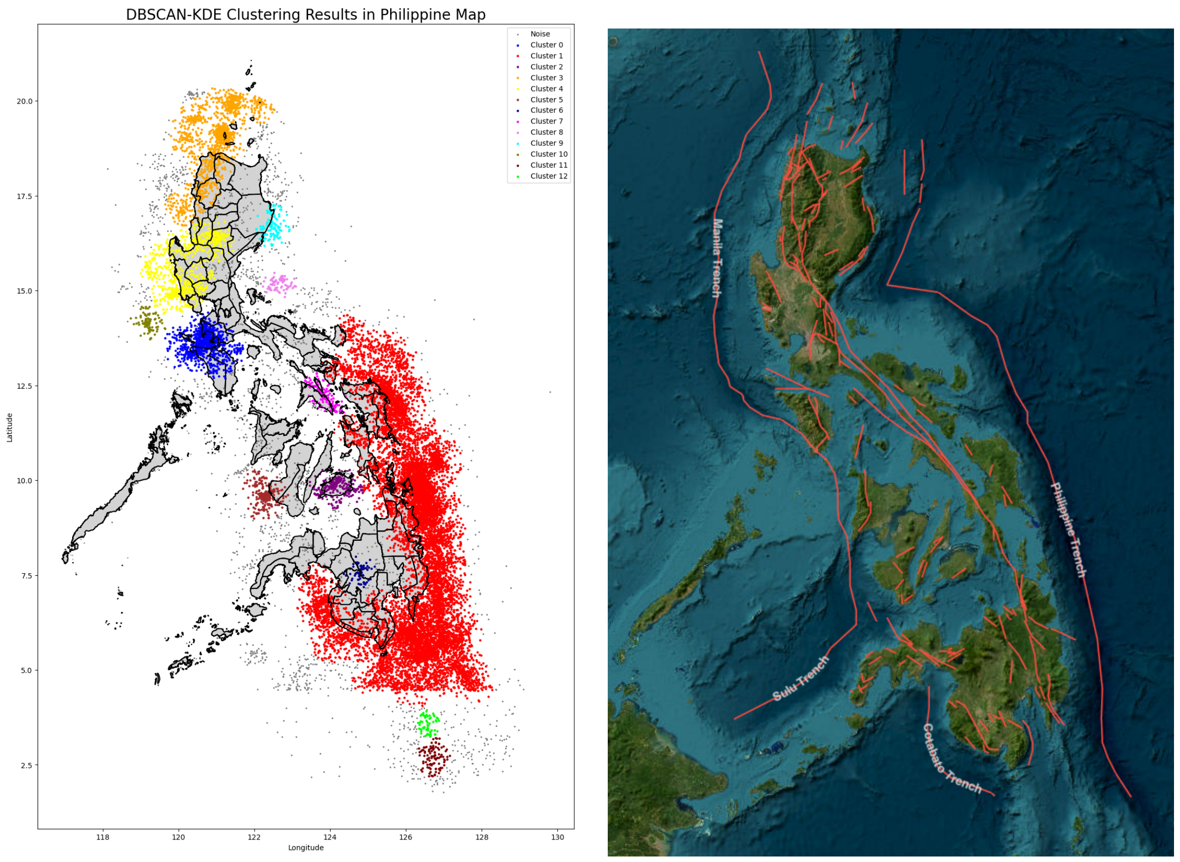


Figure 2: Earthquake Plot of the Philippines with Estimated Density

A density threshold value of 0.0024 was established to determine the regions of significant seismic activity from the background noise. This threshold value is based on the estimated density and was instituted to distinguish areas with frequent occurrences of earthquakes, by using this the cluster centers were defined as points with densities exceeding the threshold. These remaining centers represented potential epicenters of earthquake clusters and are then fed into the DBSCAN algorithm for final cluster identification.

The resulting clusters from the DBSCAN-KDE algorithm with an ϵ and $MinPts$ value of 0.3 and 50, respectively, were verified by comparing them to the map of the active faults in the Philippines, represented by red lines.



(a) (b)

Figure 3: (a) DBSCAN-KDE Clustering of Earthquake Data Points and (b) Active Faults in the Philippines.

Figure 3 shows (a) the results of the DBSCAN-KDE clustering and (b) the active faults in the Philippines. This side-by-side visualization allows for a clear comparison of the identified clusters and the actual locations of the seismic activity. A close alignment between the clusters generated by DBSCAN-KDE and the known active faults in the Philippines can be observed. It can be seen that regions in the Southeastern part of the Philippines are where earthquakes frequently occur, followed by regions in the northern part of the Philippines. This visual validation step of the clustering results of the DBSCAN-KDE technique is crucial because it assesses how well the DBSCAN-KDE algorithm performs in real-world applications.

In addition, the bandwidth value obtained is 5.9566 with Silhouette width, computed using the generalized mean with parameter $p = 2$, for the resulting clustering solutions of the DBSCAN-KDE algorithm, is found to be 0.52846. In this context, a Silhouette width value above 0.5 suggests a reasonable split of clusters. In contrast to the DBSCAN algorithm, since the DBSCAN-KDE approach has a dt parameter that filters data points that are to be part of the clustering solutions, it can be assumed that those data points that are more likely noise points will not be included in the DBSCAN process, hence the reason why DBSCAN-KDE approach obtained a high SD even though both algorithms produced the same clustering solutions. Nonetheless, adjusting the algorithm's parameters or exploring alternative approaches could enhance the separation and cohesion of the identified clusters, leading to more meaningful insights from the data.

5 Summary And Recommendation

Summary

In both the simulation and empirical study, three main points were observed. First, the DBSCAN algorithm is more parameter-sensitive than the DBSCAN-KDE algorithm, especially if the dataset contains greater dispersion. Secondly, when the density threshold (dt) parameter is selected accordingly, the DBSCAN-KDE approach consistently beats the DBSCAN algorithm and is not particularly parameter-sensitive. Lastly, the integration of the kernel density estimation method and dt parameter helps to enhance the performance of the DBSCAN algorithm.

In the simulation study, specific observations were made: a) Table 2 presents the summary of the results obtained in Scenario 1 (S1) where a comparison of the DBSCAN and DBSCAN-KDE algorithms with equal SD within each cluster and dt set to 0. Examining these results both algorithms performed identically. However, when SD is equal to 2.8 and 3.0, both algorithms gave a lower value of average SW across 100 iterations; b) When dt is set to 0 with SD equal to 2.8 and 3.0 DBSCAN and DBSCAN-KDE failed to distinguish the predetermined number of clusters correctly; c) Table 3 summarizes the results of DBSCAN and DBSCAN-KDE algorithms in Scenario 2 (S2) where both algorithms are compared under varying SD within each cluster and dt set to 0. Based on the acquired results both algorithms continuously performed similarly; and d) Table 4 and Table 5 give the summary results of Scenario 3 (S3) and Scenario 4 (S4), respectively. It can be observed in the results that the benefits of integrating the kernel density estimation method in the DBSCAN algorithm with dt parameter set to 0.0005 is satisfactory. Generally, during the simulation, DBSCAN-KDE consistently gives higher SW values under uniform and varying SD among data points within the cluster over 100 iterations.

In conclusion, the comparative analysis highlights the major advantages of employing the DBSCAN-KDE algorithm over the traditional DBSCAN algorithm, particularly in scenarios where data dispersion varies greatly. The incorporation of KDE as a pre-processing approach additionally enhances the algorithm's performance and also broadens its application to a wider range of datasets, making it a more flexible and useful tool for cluster analysis. Additionally, the DBSCAN-KDE algorithm was applied to real-world datasets in an empirical investigation, resulting in clustering solutions that suggest a reasonable split of clusters, and a high SW value was obtained than the traditional DBSCAN algorithm.

Recommendations

Interestingly, it should also be noted that although the DBSCAN-KDE algorithm demonstrates good results in this particular case, the optimal density threshold, in any case, will always depend on the dataset and the specific problem at hand. As a result, it should be selected during in-depth analyses and comparisons. Here are a few potential areas of research: (a) developing an approach for determining the optimal value of the density threshold can be compelling to explore; (b) possible enhancements including methodically choosing the optimal values for the ϵ and $MinPts$ parameters; (c) adding new features or data types. For instance, additional data related to geography or earthquake epicenters could enhance the algorithm resulting in even more precise clustering results; (d) using different kernels or bandwidths; and (e) comparing the performance of the DBSCAN-KDE algorithm with other clustering algorithms. These areas could provide a more comprehensive understanding of the strengths and weaknesses of the DBSCAN-KDE algorithm in different contexts.

Acknowledgements

The authors would like to express their heartfelt gratitude to everyone who contributed to the successful publication of this journal article, particularly the anonymous referees, for their invaluable insights, constructive feedback, and meticulous evaluation of our manuscript. Their expertise and dedication significantly improved the quality and credibility of this work.

References

[1] Anneken, M., Fischer, Y., & Beyerer, J. (2015). *Evaluation and comparison of anomaly detection algorithms in annotated datasets from the maritime domain.* In 2015 SAI Intelligent Systems Conference (IntelliSys). doi:10.1109/IntelliSys.2015.7361141

[2] Birant, D., & Kut, A. (2007). *ST-DBSCAN: An algorithm for clustering spatial–temporal data.* Data and Knowledge Engineering, 60(1), 208–221. doi:10.1016/j.datak.2006.01.013

[3] Bruce E. Hansen, (2009). *Lecture Notes on Nonparametrics.* Univeristy of Wisconsin

[4] Cai, X., Wu, Z., & Cheng, J. (2013). *Using kernel density estimation to assess the spatial pattern of road density and its impact on landscape fragmentation.* International Journal of Geographical Information Science, 27(2), 222–230. doi:10.1080/13658816.2012.663918

[5] Cesca, S., Şen, A. T., & Dahm, T. (2013). *Seismicity monitoring by cluster analysis of moment tensors.* Geophysical Journal International, 196(3), 1813–1826. doi:10.1093/gji/ggt492

[6] Chatfield, C. (2000). *Time-Series Forecasting.* In Chapman and Hall/CRC eBooks. doi:10.1201/9781420036206

[7] Chen, Y. (2017). *A tutorial on kernel density estimation and recent advances.* arXiv. Retrieved from doi:10.48550/arxiv.1704.03924

[8] Cheng, T. (2017). *An Improved DBSCAN Clustering Algorithm for Multi-Density Datasets.* In Proceedings of the 2nd International Conference on Intelligent Information Processing (pp. 1–5). Association for Computing Machinery. doi:10.1145/3144789.3144808

[9] Elsharkawi, M. (2009). *Algorithm for Spatial Clustering with Obstacles.* ArXiv. doi:10.48550/arXiv.0909.4412

[10] Ester, M., Kriegel, H., Sander, J., & Xu, X. (1996). *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise.* Knowledge Discovery and Data Mining. CorpusID:355163

[11] Gaonkar, M.N., & Sawant, K. (2013). *AutoDBSCAN: DBSCAN Clustering of Large Dataset.* academia.edu/26525668

[12] Gaonkar, M.N., & Sawant, K. (2013). *AutoEpsDBSCAN: DBSCAN with Eps Automatic for Large Dataset.* academia.edu/26525674

[13] Gholizadeh, N., Saadatfar, H., & Hanafi, N. (2020). *K-DBSCAN: An improved DBSCAN algorithm for big data.* Journal of Supercomputing, 77, 6214–6235. doi:10.1007/s11227-020-03524-3

[14] Great Learning Team. (2023, May 30). *An Introduction to GridSearchCV — What is Grid Search — Great Learning*. Great Learning Blog: Free Resources What Matters to Shape Your Career!. blog/gridsearchcv/

[15] Gyamerah, S.A., Ngare, P., & Ikpe, D. (2019). *Crop yield probability density forecasting via quantile random forest and Epanechnikov Kernel function*. ArXiv, abs/1904.10959. doi:10.48550/ARXIV.1904.1095

[16] Hinneburg, A., & Gabriel, H. (2007). *DENCLUE 2.0: Fast clustering based on kernel density estimation*. In Lecture notes in computer science (pp. 70–80). doi:10.1007/978-3-540-74825-0_7

[17] Khare, K. (2016). *Integration and Evaluation of Different Kernel Density Estimates in Hierarchical Density-Based Clustering*. ERA: Open Access Publications. doi:10.7939/R3B853V02

[18] Konstantaras, A., Katsifarakis, E., Maravelakis, E., Skounakis, E., Kokkinos, E., & Karapidakis, E. (2012). *Intelligent Spatial-Clustering of seismicity in the vicinity of the Hellenic Seismic Arc*. Earth Science Research, 1(2). doi:10.5539/esr.v1n2p1

[19] Lengyel, A., & Botta-Dukát, Z. (2019). *Silhouette width using generalized mean—A flexible method for assessing clustering efficiency*. Ecology and Evolution, 9(23), 13231–13243. doi:https:10.1002/ece3.5774

[20] Lippiello, E., Marzocchi, W., De Arcangelis, L., & Godano, C. (2012). *Spatial organization of foreshocks as a tool to forecast large earthquakes*. Scientific Reports, 2(1). doi:10.1038/srep00846

[21] Li, S., Liu, Z., & Zhao, H. (2020). *K-Means clustering method based on kernel density estimation to analysis residents travel features: A case study of Chengdu*. Journal of Physics: Conference Series, 1646(1), 012018. doi:10.1088/1742-6596/1646/1/012018

[22] Liu, P., Zhou, D., & Wu, N. (2007). *VDBSCAN: Varied density based spatial clustering of applications with noise*. International Conference on Service Systems and Service Management. doi:10.1109/icsssm.2007.4280175

[23] Loi, N. V., & Nguyen Van, L. (2020). *Abnormal Moving Speed Detection Using Combination of Kernel Density Estimator and DBSCAN for Coastal Surveillance Radars*. In 2020 7th International Conference on Signal Processing and Integrated Networks (SPIN) (pp. 143–47). IEEE. doi:10.1109/SPIN48934.2020.9070885

[24] Malzer, C., & Baum, M. (2020). *A Hybrid Approach To Hierarchical Density-based Cluster Selection*. 2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI). doi:10.1109/mfi49285.2020.9235263

[25] Mamat, A. R., Mohamed, F. S., Mohamed, M. A., Rawi, N. M., & Awang, M. K. (2018). *Silhouette index for determining optimal k-means clustering on images in different color models*. International Journal of Engineering & Technology, 7(2.14), 105. doi:10.14419/ijet.v7i2.14.11464

[26] Matioli, L. C., et al. (2018). *A New Algorithm for Clustering Based on Kernel Density Estimation*. Journal of Applied Statistics, 45(2), 347–66. doi:10.1080/02664763.2016.1277191

[27] Moraes, C. P. A., Fantinato, D. G., & Neves, A. (2021). *Epanechnikov kernel for PDF estimation applied to equalization and blind source separation.* Signal Processing, 189, 108251. doi:10.1016/j.sigpro.2021.108251

[28] Moreira, A., Maribel, Y. S., Santos, & Carneiro, S. (2005). *Density based clustering algorithms, DBSCAN and SNN.*CorpusID:14733163

[29] Ouillon, G., & Sornette, D. (2011). *Segmentation of fault networks determined from spatial clustering of earthquakes.* Journal of Geophysical Research, 116(B2). doi:10.1029/2010jb007752

[30] Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825–2830. jmlr.org/papers/v12/pedregosa11a.html

[31] Ram, A., Jalal, S., Jalal, A. S., & Kumar, M. (2010). *A Density Based Algorithm for Discovering Density Varied Clusters in Large Spatial Databases.* International Journal of Computer Applications, 3(6), 1–4. doi:10.5120/739-1038

[32] Ranjan, G., Verma, A. K., & Radhika, S. (2019). *K-Nearest Neighbors and Grid Search CV Based Real Time Fault Monitoring System for Industries.* In 2019 IEEE 5th International Conference for Convergence in Technology (I2CT) (pp. 1–5). doi:10.1109/i2ct45611.2019.9033691

[33] Richardson, S. (2023). *A sharp Fourier inequality and the Epanechnikov kernel.* arXiv (Cornell University). doi:10.48550/arxiv.2310.09713

[34] Rousseeuw, P. J. (1987). *Silhouettes: A graphical aid to the interpretation and validation of cluster analysis.* Journal of Computational and Applied Mathematics, 20, 53–65. doi:10.1016/0377-0427(87)90125-7

[35] Rosenblatt, M. (1956). *Remarks on some nonparametric estimates of a density function.* Annals of Mathematical Statistics, 27(3), 832–837. doi:10.1214/aoms/1177728190

[36] Samiuddin, M., & El-Sayyad, G. M. (1990). *On nonparametric kernel density estimates.* Biometrika, 77(4), 865–874. doi:10.1093/biomet/77.4.865

[37] Schubert, E., Sander, J., Ester, M., Kriegel, H. P., & Xu, X. (2017). *DBSCAN revisited, revisited.* ACM Transactions on Database Systems, 42(3), 1–21. doi:10.1145/3068335

[38] Scott, D. W. (2015). *Multivarite Density Estimation and Non Parametric Methods.*

[39] Shah, H., Napanda, K., & D'mello, L. (2015). *Density Based Clustering Algorithms.* International Journal of Computer Sciences and Engineering, 3(11), 54-57.

[40] Silverman, B. W. (2018). *Density estimation for statistics and data analysis.* In Routledge eBooks. doi:10.1201/9781315140919

[41] Veeralagan, J., & Priya, S. M. (2022). *Hyper tuning using GridSearchCV on machine learning models for prognosticating dementia.* Research Square (Research Square). doi:10.21203/rs.3.rs-2316713/v1

[42] Wand, M., & Jones, M. (1994). *Kernel smoothing.* In Chapman and Hall/CRC eBooks. doi:10.1201/b14876

[43] Wang, X., & Hamilton, H. J. (2003). *DBRS: A Density-Based Spatial Clustering Method with Random Sampling*. In Lecture Notes in Computer Science (pp. 563–575). doi:10.1007/3-540-36175-8_56

[44] Weglarczyk, S. (2018). *Kernel density estimation and its application*. ITM Web of Conferences, 23, 00037. doi:10.1051/itmconf/20182300037

[45] Wu, C. O. (1997). *The effects of kernel choices in density estimation with biased data*. Statistics & Probability Letters, 34(4), 373–383. doi:10.1016/s0167-7152(96)00205-2

[46] Xie, N., Miao, Z., Wang, J., & Zhang, Q. (2019). *Adaptive DBSCAN Algorithm Based on Sample Density Gradient*. Journal of Physics: Conference Series, 1229, 012056. doi:10.1088/1742-6596/1229/1/012056

[47] Yamasaki, R., & Tanaka, T. (2019). *Kernel Selection for Modal Linear Regression: Optimal Kernel and IRLS Algorithm*. 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA). doi:10.1109/ICMLA.2019.00110